Dieses Dokument ist eine Zweitveröffentlichung (Postprint) /

This is a self-archiving document (accepted version):

J. Albrecht, H. Günzel, W. Lehner

Set-Derivability of Multidimensional Aggregates

Erstveröffentlichung in / First published in:

Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast, 30.08. - 01.09.1999. Springer, S. 133-142. ISBN 978-3-540-48298-7.

DOI: https://doi.org/10.1007/3-540-48298-9_15

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-828917

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

Set-Derivability of Multidimensional Aggregates

J. Albrecht, H. Günzel, W. Lehner[1]
Department of Database Systems, University of Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
{jalbrecht, guenzel, lehner}@informatik.uni-erlangen.de

Abstract. A common optimization technique in data warehouse environments is the use of materialized aggregates. Aggregate processing becomes complex, if partitions of aggregates or queries are materialized and reused later. Most problematic are the implication problems regarding the restriction predicates. We show that in the presence of hierarchies in a multidimensional environment an efficient algorithm can be given to construct - or to derive - an aggregate from one or more overlapping materialized aggregate partitions (*set-derivability*).

1 Introduction

In the last few years data warehousing has emerged from a mere buzzword to a fundamental database technology. Today, almost every major company is deploying an integrated, historic database, the data warehouse, as a basis for multidimensional decision support queries. The purpose is to provide business analysts and managers with online analytical processing (OLAP). Besides the use of big parallel database servers, a common optimization technique is to precompute aggregates, i.e. to use summary tables or materialized views (e.g. [3], [8], [9], [15]). Most of the presented algorithms base on the assumption that during the data warehousing loading process a pre-determined set of aggregates is materialized and used during the analysis phase. But there is also a great performance potential in the dynamic reutilization of cached query results ([1], [5]).

However, today the transparent reuse of aggregates is based on limited cases of query containment, i.e. the query must be contained in *one* certain aggregate. Since the implication problem for query restrictions containing the six comparison operators as well as disjunctions and conjunctions is solvable NP-hard [13], algorithms like [9] as well as commercial products (e.g. [3]) are based on aggregate views defined without restrictions to circumvent this problem. Using this approach, the definition and reuse of *aggregate partitions* for hot spots, like the current month or the most important product group, and the reuse of queries are impossible.

In many cases this is too restrictive. Consider the query *"Give me the total sales for the video product families by region in Germany"* and the tabular result illustrated in figure 1. The materialized query represents a partition of

Sum(Sales)	Camcorder	HomeVCR
G-East	12	37
G-West	22	32

Fig. 1. A partition of an aggregated data cube.

[1] Current address: IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, U.S.A.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

an aggregated data cube. If there were two redundant aggregates in the database, one containing the sales for camcorders and the other one containing the sales for home VCRs, then the query could be computed by the union of these aggregates. The goal of this article is to provide a constructive solution for this problem.

In the presence of a set of materialized aggregate partitions, a multidimensional query optimizer has to determine under which circumstances and *how* a query can be computed from these aggregates (figure 2). The basis of our approach are multidimensional objects which were initially presented in [10]. Multidimensional objects provide the information to a multidimensional query optimizer for the transparent reuse of materialized aggregate partitions. Their definition includes semantic information about it genesis, i.e. the applied aggregation and the selection predicates. Thus, a certain class of aggregation queries can be directly translated into multidimensional objects. Queries involving composite aggregations can at least utilize multidimensional objects based on the component aggregates.

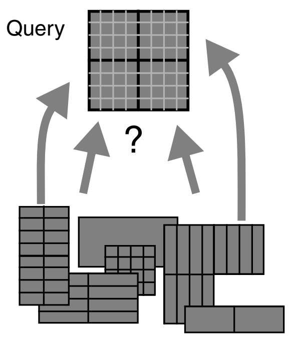


Fig. 2. Derivability problem: Can the query be computed from the set of multidimensional objects?

Structure of the Paper. The next section covers related work. Basis for the determination of derivability are the dimensional data structures presented in section 3. Section 4 introduces multidimensional objects and some basic operators. The derivability of multidimensional objects with a focus on the solution to implication problems in the presence of hierarchies is covered in section 5. The article closes with a short summary.

2 Related Work

The general idea of precomputing summary data appeared already in [4]. In the last few years it became very popular with the emergence of data warehousing and OLAP and the resulting need for an efficient and mostly read-only access to aggregates in a multidimensional context. Several articles deal with the selection and use of materialized views (e.g. [5], [8],[9]; see overview in [15]). In contrast to our approach, these articles are not able to construct a new query from a set of materialized queries but are limited to certain cases of query containment.

Summarizability and derivability are terms describing under which circumstances summary data can be derived from other summary data. [4] investigates conditions under which already aggregated cells might be further aggregated. Aggregation functions are classified as additive and computed. These notions correlate to the distinction of distributive and algebraic functions in [7]. The question under which circumstance a query is derivable from one or more other queries has been studied for a long time ([6], [13]). For summary data, disjointness and completeness are fundamental [4]. Another semantic condition, type compatibility, was identified by [11].

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

3 Dimensional Data Structures

The notion of a dimension provides a lot of semantic information especially about the hierarchical relationships between its elements like product groups or geographic regions. This information is heavily used for both aggregate queries and selections, and it provides the basis for the definition of multidimensional objects.

Definition 1: A *dimensional schema* is a partially ordered set of dimensional attributes ($\mathcal{D} \cup \{\text{Total}_D\}; \to$) where $\mathcal{D} = \{D_1, ..., D_n\}$. $\text{Total}_{\mathcal{D}}$ is a generic element which is maximal with respect to "$\to$", i.e. $D_i \to \text{Total}_{\mathcal{D}}$ for each $D_i \in \mathcal{D}$.
An attribute D_j is called a *direct parent* of D_i, denoted as $D_i \overset{\circ}{\to} D_j$, if $D_i \to D_j$ and there is no D_k with $D_i \to D_k \to D_j$.

Figure 3 shows examples for dimensional schemas illustrated as directed acyclic graphs according to the partial order "$\to$" which denotes a functional dependency, i.e. a 1:n relationship. $\text{Total}_{\mathcal{D}}$ is generic in the sense that it is not modeled explicitly.

Definition 2: The instances $c \in \text{dom}(D_i)$ of some dimensional attribute $D_i \in \mathcal{D}$ are called *classification objects* or *classes* of D_i. D_i is called the level of c.
Moreover, $\text{dom}(\text{Total}_{\mathcal{D}}) := \{\text{'ALL'}\}$.
An *instance of a dimension* $\mathcal{D}$ is the set of all classes $c \in \cup_i \text{dom}(D_i)$.

A hierarchy can be specified by a *categorization*, i.e. a path to Total in a dimension. By defining dom(Total):={'ALL'} it is guaranteed that all classification hierarchies are trees having "ALL" as the single root node. A sample classification hierarchy for the categorization Article→Family→Group→Area→Total is shown in figure 4. The edges in such a tree can also be seen as a mapping from the descendents to the ancestors.

Definition 3: Let $D_i, D_j \in \mathcal{D}$ such that $D_i \to D_j$. A class $a \in \text{dom}(D_j)$ is called *ancestor* of class $b \in \text{dom}(D_i)$, denoted as ancestor(a,b), if and only if a maps to b according to the functional dependency $D_i \to D_j$. In this case b is called a *descendant* of a, i.e. descendant(b,a)⇔ancestor(a,b).
The domain of a class a with respect to the dimensional attribute D_i is defined as the set of it descendents, i.e. $\text{dom}(a \mid D_i) = \{b \in \text{dom}(D_i): \text{descendant}(b,a)\}$.

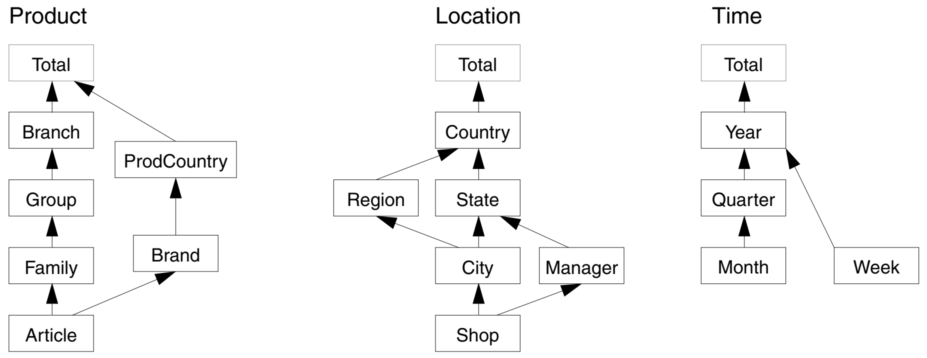


Fig. 3. Illustration the of dimensional schemas for the product, location and time dimension as directed acyclic graphs.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

4 Multidimensional Data Structures

The following definition of multidimensional objects is an extension of the work presented in [10]. In contrast to other multidimensional data structures (see [12]), multidimensional objects contain additional information besides the measures and aggregation level (granularity) which is necessary to check the derivability. One thing is that instead of treating measures as simple attributes the information about the aggregation operation which was applied to the measure is made part of their definition.

Definition 4: Let Ω be a set of additive aggregation functions. A *measure* is tuple M = (N, O), where N is a *name* for the corresponding fact and $O \in \Omega \cup \{NONE, COMPOSITE\}$ is the *operation type* applied to that specific fact.

We assume that a measure M has a numerical domain $dom(M) \in \{\mathbb{R}, \mathbb{N}, \mathbb{Q}, \mathbb{Z}\}$ and $\Omega = \{SUM, COUNT, MIN, MAX\}$. Only additive operations (in the sense of [4]) are explicitly represented. Other operations are subsumed by the operation type COMPOSITE, i.e. those measures can not be used for the automatic derivation of higher aggregates. However, for many composite operations, like AVG, one can extend our concept by implicitly storing SUM and COUNT. The value NONE states that a measure is not aggregated.

Definition 5: A multidimensional object over the dimensions $\mathcal{D}_1,...,\mathcal{D}_d$ is a triple $\mathcal{M}$ = [M, G, S] where

- $M = (M_1,...,M_m) = ((N_1,O_1), ..., (N_m,O_m))$ is a set of measures[2]
- $G = (G_1,...,G_n)$ is the granularity specification consisting of a set of dimensional attributes, i.e. $G_i \in \mathcal{D}_1 \cup ... \cup \mathcal{D}_d$ such that for each G_i, G_j: $G_i \not\rightarrow G_j$
- S is logical predicate denoting the scope.

The scope is a restriction predicate describing *which* data cells have been aggregated in this particular (sub-) cube. It may include any propositional logic expression involving the granularity attributes of $\mathcal{M}$ and any dimensional attribute that is functionally dependent on some $G_i \in G$. For example, the multidimensional object in figure 4 is[3]

[(Sales, SUM),(P.Family, L.City, T.Month),(P.Area='Brown Goods'^L.Country='Germany')]

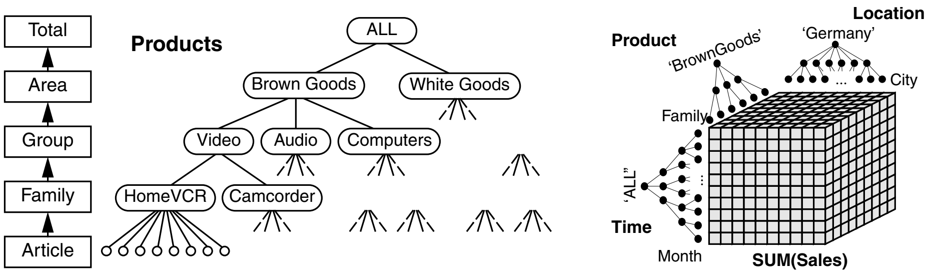


Fig. 4. A classification hierarchy for the product dimension and a multidimensional object.

[2] The definition of M and G as tuples is only for the sake of simplicity; the order of the elements does not matter. Therefore, we will also apply the set operators like $\in$, $\cup$, $\cap$, = to M and G.

[3] In the following examples we will abbreviate the dimension names Product, Location and Time with P, L and T, respectively.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

In the literature on multidimensional data models several operators were defined on multidimensional data cubes [12]. Most important are selections and aggregations. The goal of this section is to define the influence of these operators on multidimensional objects, especially on the measures and the scope. Since the definition of operators is not the topic of this article, we will only shortly mention some operators.

Fundamental are the *metric projection* of the attributes $M'=(M_1',...,M_k')\subseteq M$, written as $\pi_{M'}(\mathcal{M}) = [M', G, S]$, and the *restriction* by a predicate P, defined as $\sigma_P(\mathcal{M}) = [M, G, S \wedge P]$. On set-compatible multidimensional objects $\mathcal{M}$ and $\mathcal{M}'$ (i.e. M=M' and G=G') one can define the common *set-operations* $\mathcal{M} \cup \mathcal{M}' = [M, G, S \vee S']$, $\mathcal{M} \cap \mathcal{M}' = [M, G, S \wedge S']$, and $\mathcal{M} \setminus \mathcal{M}' = [M, G, S \wedge \neg S']$. However, the most important operations on multidimensional objects are aggregations, which are defined first on the measures alone and then on multidimensional objects.

Definition 6: The application of an aggregation function F to a measure M = (N,O) results in a measure F(M)=(N,O') where

- O' = F if O=NONE or if F=O and $O \in \{SUM,MIN,MAX\}$,
- O' = COUNT if O=COUNT and F=SUM
- O' = COMPOSITE otherwise.

A granularity specification $G=(G_1,...,G_n)$ is *finer than or equal to* a granularity specification $G'=(G_1',...,G_k')$, denoted as $G \leq G'$, if and only if for each $G_j' \in G'$ there is a $G_i \in G$ such that $G_i \rightarrow G_j'$. For example (P.Article, L.City)≤(P.Group, L.Region)≤(P.Area).

Definition 7: The *aggregation* of a multidimensional object $\mathcal{M}$ by a family of aggregate functions $\Phi=(F_1,...,F_m)$ to the granularity $G' \geq G$ is defined as:

$$\Phi(G', \mathcal{M}) = [\,(F_1(M_1),...,F_m(M_m)),\ G',\ S\,]$$

For example, if $\mathcal{M}$ = [(Sales, SUM),(P.Family),(P.Group = 'Video')] then
(SUM) ((P.Group), $\mathcal{M}$) = [(Sales, SUM),(P.Group),(P.Group = 'Video')] and
(AVG) ((), $\mathcal{M}$) =[(Sales, COMPOSITE),(P.Group),(P.Group = 'Video')].

5 Derivability in the Presence of Hierarchies

Based on the definitions of the last section, we will now define under which conditions and how a multidimensional object can be computed from a set of materialized MOs. A necessary prerequisite to derive a multidimensional object is that the aggregation level of the original MOs is finer than the granularity of the derived MO. This condition directly corresponds to the relationship of the aggregates in an aggregation lattice [9]. Two further conditions, measure compatibility and reconstructibility, are necessary to define the derivability of multidimensional objects.

Definition 8: A multidimensional object $\mathcal{M}$ = (M, G, S) is *derivable* from a multidimensional object $\mathcal{M}'$ = (M',G',S') if and only if

- for each measure $M_i \in M$ there is $M_j' \in M'$ such that $N_i = N_j'$ and O = O' or O' = NONE
- the granularity specification of $\mathcal{M}'$ is finer than $\mathcal{M}$, i.e. $G' \leq G$
- S is contained in S', i.e. $S \subseteq S'$ (or $S \rightarrow S'$) and S is reconstructible from S'.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

Measure compatibility simply means that for example total sales are derivable from total sales. Most problematic is the third condition, one is that S is contained by S' (considering S and S' as sets of dimensional elements). How to check this condition and the notion of reconstructibility is explained in the next section.

5.1 Scope Normalization

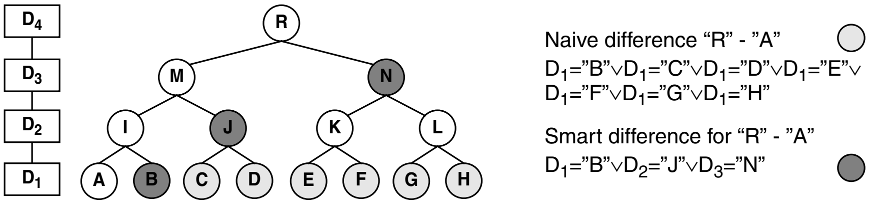
In [13] it is shown that the problem to determine that some predicate S is implied by another predicate S' is NP-complete if the predicates may contain disjunctions and inequalities besides simple comparison operators. In this section we will give an efficient polynomial time algorithm, which solves implication problems in the presence of hierarchies on finite domains even for negations and disjunctions. The algorithm is based on compact scopes for which the determination of scope containment is very simple.

Definition 9: A scope S is *compact* if S is a conjunction of positive terms and there are no two different terms $\mathcal{D}_i.D_j$=c and $\mathcal{D}_i.D_k$=c' with ancestor(c, c').

Thus, the scope ((P.Family='Camcorder'∨P.Family='HomeVCR')^L.Country='Germany') is not compact, but (P.Group='Video'^L.Country='Germany') is.

A compact scope S is *contained* in a scope S' (denoted as S→S' or S⊆S') if and only if for each term $\mathcal{D}_i.D_k$=c' in S' there exists a term $\mathcal{D}_i.D_j$=c in S such that ancestor(c', c). For example, (P.Family='HomeVCR')⊄(P.Group='Video'^L.Country='Germany') but (P.Family='HomeVCR' ^L.Country='Germany')⊆(P.Group='Video').

Not only scope containment, but also all problems of finding the intersections or differences of two compact scopes can be solved simply by determining ancestor/descendant relationships of classes appearing in the conjunctive clauses. Both operations are based on the one-dimensional intersection and the difference of two classes. For the intersection of two classes c∈dom(D_i) and c'∈dom(D_j) holds c∩c'=c if ancestor(c',c) and c∩c'=∅ otherwise. For example in figure 5 M∩B=B and M∩E=∅. Intersections of classes in parallel hierarchies like P.Family='HomeVCR'^P.Brand='Sony' are not resolved but treated as if it were separate dimensions. The difference of two classes can be computed by the algorithm ClassDifference as illustrated in figure 5 (see [2] for the complete algorithm).



D_4
D_3
D_2
D_1
R
M
N
I
J
K
L
A
B
C
D
E
F
G
H
Naive difference "R" - "A"
D_1="B"∨D_1="C"∨D_1="D"∨D_1="E"∨ D_1="F"∨D_1="G"∨D_1="H"
Smart difference for "R" - "A"
D_1="B"∨D_2="J"∨D_3="N"

Fig. 5. Illustration of the algorithm ClassDifference.

Each scope can be transformed into a "minimal" disjunction of mutually disjoint compact scopes, the disjunctive scope normal form (DSNF). Based on the DSNF and the scope difference the scope implication problems for non-compact scopes can be solved in a constructive way. To explain the construction of the DSNF consider the following multidimensional object:

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

[(Sales, SUM),
(P.Group, L.Region),
(P.Group = ‘Video’ ^ L.Region = ‘G-East’) ∨
((L.Region = ‘G-West’ ∨ L.Region = ‘G-East’) ^
P.Group = ‘Video’ ^ P.Family ≠ ‘Camcorder’)]

Sum(Sales)	Camcorder	HomeVCR
G-East	12	37
G-West		32

Intuitively, the scope definition is not minimal, because the terms L.Region=‘G-East’ ∨ L.Region=‘G-West’ can be reduced to L.Country=‘Germany’. In order to find all such terms, it is necessary to translate the predicate into conjunctive normal form where such terms appear in a single disjunctive clause and can be discovered easily. This kind of reduction together with the replacement of negative terms is realized by algorithm 1 (see [2] for details), which constructs the conjunctive scope normal form (CSNF), i.e. a minimal expression of the scope in CNF. The resulting scope for the example above is

(P.Group = ‘Video’ ∨ L.Country = ‘Germany’) ∧ (P.Group = ‘Video’) ∧
(L.Country = ‘Germany’) ∧ (L.Region = ‘G-East’ ∨ P.Family = ‘HomeVCR’)

The translation from CSNF into DSNF is analogous to translating CNF into DNF. This implies that all positive terms remain positive. The example yields the following DNF:

(P.Group = ‘Video’ ∧ L.Country = ‘Germany’ ∧ L.Region = ‘G-East’) ∨
(P.Group = ‘Video’ ∧ P.Family = ‘HomeVCR’ ∧ L.Country = ‘Germany’)

To make the clauses compact, for each class it must now be checked if an ancestor is also in the same clause. If so, the ancestor is removed. This leads to

(P.Group = ‘Video’ ^ L.Region = ‘G-East’) ∨ (P.Family = ‘HomeVCR’ ^ L.Country = ‘Germany’)

Algorithm: ConjunctiveScopeNormalization

```
Input:    Scope of a MO over dimensions $\mathcal{D}_1,...,\mathcal{D}_n$
          in conjunctive normal form $S = S_1 \wedge ... \wedge S_k$
Output:   Scope S in conjunctive scope normal form
Begin
  Foreach $S_i$
    replace all negative terms $\mathcal{D}_j.D_k \neq c$ by
      ClassDifference($\mathcal{D}_j$.Total="ALL", $\mathcal{D}_j.D_k \neq c$);

    Foreach term $\mathcal{D}_j.D_k = c$
      If (c’ with Ancestor(c’, c) is also contained)
        remove $\mathcal{D}_j.D_k \neq c$

    Foreach term $\mathcal{D}_j.D_k = c$
      let $D_p$ represent a direct parent level,
        i.e. $\mathcal{D}_j.D_k \overset{\circ}{\rightarrow} D_p$;
      p = GetAncestor(child | $D_p$);
      If (all elements of dom(p | $D_k$) c are in $S_i$)
        replace c and all siblings by p;

  End Foreach

  Return $S = S_1 \wedge ... \wedge S_n$;
End
```

Algorithm 1: ConjunctiveScopeNormalization transforms a scope from conjunctive normal form to conjunctive scope normal form.

Algorithm: PatchWork

```
Input:    A compact scope SC and
          a scope in DSNF $S = SC_1 \cup ... \cup SC_n$
Output:   TRUE if $S \rightarrow SC$, FALSE otherwise
Begin
  remainder = {SC};
  solution = $\emptyset$;

  While remainder $\neq \emptyset$ Do
    Foreach R $\in$ remainder
      found = false;
      For i = 1 To n
        // check if this part of the remainder is
        // intersected by a compact scope in S
        If Intersection(R, $SC_i$)$\neq \emptyset$ Then
          remainder = remainder \ {R} $\cup$
                         ScopeDifference(R, $SC_i$);
          solution = solution$\cup$Intersection(R,$SC_i$);
          found = TRUE;
          Break;
        End If
      End For
      If (Not found)
        Return $\emptyset$;
    End Foreach
  End While
  Return solution;
End
```

Algorithm 2: PatchWork constructs a solution in DSNF how to compute the compact scope SC from $SC_1,...,SC_n$.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

By using the ScopeDifference it is now possible to make the clauses mutually disjoint. Therefore, a DSNF representation of the scope is

(P.Group = 'Video' ^ L.Region = 'G-East') $\vee$ (P.Family = 'HomeVCR' ^ L.Region = 'G-West')

Based on the DSNF, the problem if a scope $S_1=SC_{11}\vee...\vee SC_{1n}$ is contained in a scope $S_2=SC_{21}\vee...\vee SC_{2m}$ can be solved constructively by algorithm 2. For each conjunctive term SC_{1i} the following steps are executed. The remainder is a set of "patches", i.e. "compact" fragments which still must be covered. For each remaining patch R it must be checked if $R\cap SC_{2i}\neq\emptyset$ for some i (line 11). If so, the intersection is removed from the remainder and added to the solution (lines 12-14). If for some remaining patch no intersecting clause from S_2 is found, then $S_1 \not\subseteq S_2$. Thus, the solution itself is a set of mutually disjoint compact scopes (patches) and can be seen as an instruction *how* to compute SC_{11} from SC_2.

However, this construction does not work in all cases because a multidimensional object still may not be *reconstructible* from the other one. Consider the query: *"Give me the total sales of all video and audio products per region"* expressed by the multidimensional object

$\mathcal{M}$ = [(Sales, SUM), (L.Region), (P.Group='Video'$\vee$P.Group='Audio')].

If there was an aggregate with no restrictions (equivalent to Total="ALL" in all dimensions)

$\mathcal{M}'$ = [(Sales, SUM), (P.Area, L.Region), ()],

then question is, if $\mathcal{M}$ is derivable from $\mathcal{M}'$. It turns out that, although $G\leq G'$ and $S\subseteq S'$, $\mathcal{M}'$ is not reconstructible from $\mathcal{M}$ because the two patches with P.Group='Video' and P.Group='Audio' can not be addressed in $\mathcal{M}'$. The reason is that $\mathcal{M}'$ has already a higher granularity (P.Area) than the attributes in the patch clauses (P.Group). This must also be checked (see definition 8). However, in case S=S' it would work anyway.

5.2 Set-Derivability

A set of multidimensional objects { [M, G, SC_1],...,[M, G, SC_n] } can also be seen as a MO $\mathcal{M}$=[M, G, $SC_1\vee...\vee SC_n$] and the other way around. Since it is easy to aggregate MOs at a finer granularity $G'\leq G$ (definitions and 8) to G, one can easily extend algorithm 2 to construct one multidimensional object $\mathcal{M}_Q$ from a set of multidimensional objects $\mathcal{M}_1,...,\mathcal{M}_n$ at granularities $G'\leq G$ (figure 6). Thus, $\mathcal{M}_Q$ is *set-derivable* from $\mathcal{M}_1,...,\mathcal{M}_n$ if algorithm 3 yields a non-empty solution. Set-derivability in conjunction with a cost-based selection can serve as a basis to compute one query from a set of previously materialized queries. For an illustration consider the following multidimensional objects in DSNF:

$\mathcal{M}_1$ = [(Sales, SUM), (P.Family, L.Region), (P.Group = 'Video' ^ L.Country='Germany')]
$\mathcal{M}_2$ = [(Sales, SUM), (P.Group, L.City), ((P.Area = 'Brown Goods' ^ L.Region='G-West')]
$\mathcal{M}_3$ = [(Sales, SUM), (P.Article, L.City), ((P.Group = 'Audio' ^ L.Country = 'Germany') $\vee$
(P.Group = 'Video' ^ L.Region='G-West')]

The use of algorithm 2 in the context of set-derivability can be used to derive the query

$\mathcal{M}_Q$ = [(Sales, SUM), (P.Goup, L.Region), ((P.Area = 'Brown Goods' ^ L.Country='Germany')]

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

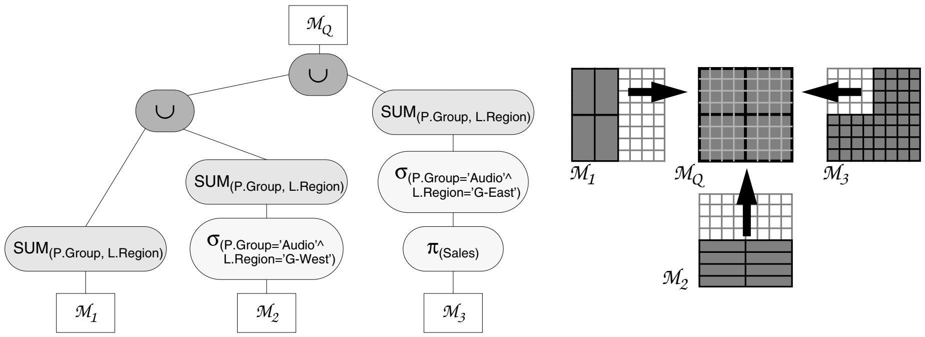


Fig. 6. Patch-working. The requested MO $\mathcal{M}_Q$ can be constructed from $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$.

from $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ by the query execution plan depicted in figure 6. Such an approach overcomes the limitations of query containment for the reuse of cached aggregates and bears a high performance potential [1].

6 Summary and Future Work

The determination of the derivability of multidimensional aggregates is an essential task for a multidimensional query optimizer. In this article we presented multidimensional objects as an enriched data structure which helps to accomplish this task. We have shown that the inclusion of the semantics of aggregations on the measures in an extended multidimensional algebra allows much more flexibility for the selection of aggregates by the query optimizer than is possible today. For the derivability of multidimensional objects three conditions have to be checked: measure and granularity compatibility as well as scope containment. An efficient algorithm was given which solves the scope implication problem in the presence of hierarchies in a constructive way. The potential of the approach has already been proved for a certain class of multidimensional objects. Experimental results are given in [1].

Future research aims at an extension of the presented concept on a more complete multidimensional algebra, including other aggregation functions and also binary operations. Another idea is to include comparisons on the aggregated measure attributes in the scope restriction, a problem that has already been investigated in [14]. Our strategic goal is to supply the query optimizer with sufficient knowledge to solve problems of the following kind: *"Given a formula* Turnover=Sales*Price *and an aggregated sales data cube, under which circumstances is it possible to use this aggregated sales cube to derive an aggregated turnover cube?"* There are many possibilities under which the information about computed measures can be used to utilize materialized multidimensional objects for the actual computation. In several relevant cases binary operations do not change the operation type of the resulting measure, for example, (Stock,SUM)=(StockReceipt,SUM)-(Sales, SUM). In such cases the total Stock can be computed from the total StockReceipt minus the total Sales.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: First International Conference. Kohala Coast 1999", S. 133-142, ISBN 978-3-540-48298-7
https://doi.org/10.1007/3-540-48298-9_15

References

1. Albrecht, J.; Bauer, A.; Deyerling, O.; Günzel, H.; Hümmer, W.; Lehner, W.; Schlesinger, L.: Management of multidimensional Aggregates for efficient Online Analytical Processing, in: *International Database Engineering and Applications Symposium* (IDEAS'99, Montreal, Canada, August 1-3), 1999
2. Albrecht, J.; Günzel, H.; Lehner, W.: Set-Derivability of Multidimensional Aggregates (long version), Technical Report, University of Erlangen, 1999 (http://www6.informatik.uni-erlangen.de/publications)
3. Bello, R.; Dias, K.; Downing, A.;Feenan, J.; Norcott, W.; Sun, H.; Witkowski, A.;Ziauddin, M.: Materialized Views in Oracle, in: *Proceedings of 24th International Conference on Very Large Data Bases* (VLDB'98, New York, USA, August 24-27), 1998,
4. Chen, M. C.; McNamee, L.; Melkanoff, M.: A Model of Summary Data and its Applications in Statistical Databases, in: *Proceedings of the 4th International Working Conference on Statistical and Scientific Database Management* (4SSDBM, Rome, Italy, June 21-23), 1988
5. Deshpande, P.M.; Ramasamy, K.; Shukla, A.; Naughton, J.F.: Caching Multidimensional Queries Using Chunks, in: *Proceedings of the 27th International Conference on Management of Data* (SIGMOD'98, Seattle (WA), USA, June 2-4), 1998
6. Finkelstein, S.: Common Subexpression Analysis in Database Applications, in: *Proceedings of the 11th International Conference on Management of Data* (SIGMOD'82, Orlando (FL), June 2-4), 1982
7. Gray, J.; Bosworth, A.; Layman, A.; Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, in: *Proceedings of the 12th International Conference on Data Engineering* (ICDE'96, New Orleans (LA), USA, Feb. 26-March 1), 1996
8. Gupta, A.; Harinarayan, V.; Quass, D.: Aggregate-Query Processing in Data Warehousing Environments, in: *Proceedings of the 21th International Conference on Very Large Data Bases* (VLDB'95, Zurich, Schwitzerland, Sept. 11-15), 1995, pp. 358-369
9. Harinarayan, V.; Rajaraman, A.; Ullman, J.D.: Implementing Data Cubes Efficiently, in: *Proceedings of the 25th International Conference on Management of Data*, (SIGMOD'96, Montreal, Quebec, Canada, June 4-6), 1996
10. Lehner, W.: Modeling Large Scale OLAP Scenarios, in: *6th International Conference on Extending Database Technology* (EDBT'98, Valencia, Spain, March 23-27), 1998
11. Lenz, H; Shoshani, A.: Summarizability in OLAP and Statistical Databases, in: *9th International Conferenc on Statistical and Scientfic Databases*, (SSDB'97, Olympia, Washington, Aug. 11-13), 1997
12. Sapia, C.; Blaschka, M.; Höfling, G.; Dinter, B.: Finding Your Way through Multidimensional Data Models, in: *9th International Workshop on Database and Expert Systems Applications* (DEXA'98 Workshop, Vienna, Austria, August 24-28), 1998
13. Sun, X.-H.; Kamel, N.; Ni, L.M.: Solving Implication Problems in Database Applications, in: *Proceedings of the 18th International Conference on Management of Data* (SIGMOD'89, Portland (OR), USA, May 31-June 2), 1989
14. Ross, K.; Srivastava, D.; Stuckey, P.; Sudarshan, S.: Foundations of Aggregation Constraints, in: *Theoretical Computer Science*, Volume 193, Numbers 1-2, Feb. 28, 1998
15. Theodoratos, D.; Sellis, T.: Data Warehouse Configuration, in: *Proceedings of the 23rd International Conference on Very Large Data Bases* (VLDB'97, Athens, Greece, Aug. 25-29), 1997

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden