Heuristic Optimization of OLAP Queries in Multidimensionally Hierarchically Clustered Databases*

Dimitri Theodoratos
Department of Computer Science
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
dth@cis.njit.edu

Aris Tsois
Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73, Athens, Greece
atsois@dblab.ece.ntua.gr

ABSTRACT

On-Line Analytical Processing (OLAP) is a technology that encompasses applications requiring a multidimensional and hierarchical view of data. OLAP applications often require fast response time to complex grouping/aggregation queries on enormous quantities of data. Commercial relational database management systems use mainly multiple one-dimensional indexes to process OLAP queries that restrict multiple dimensions. However, in many cases, multidimensional access methods outperform one-dimensional indexing methods.

We present an architecture for multidimensional databases that are clustered with respect to multiple hierarchical dimensions. It is based on the star schema and is called CSB star. Then, we focus on heuristically optimizing OLAP queries over this schema using multidimensional access methods. Users can still formulate their queries over a traditional star schema, which are then rewritten by the query processor over the CSB star. We exploit the different clustering features of the CSB star to efficiently process a class of typical OLAP queries. We detect special cases where the construction of an evaluation plan can be simplified and we discuss improvements of our technique.

1. INTRODUCTION

Decision support applications increasingly rely on On-Line Analytical Processing (OLAP) to analyze business related information. OLAP is a technology that encompasses applications requiring a multidimensional view of data. In such a view of data there is a set of measures that are the metrics of interest. The measures contain numeric data. Each of them is uniquely determined by a set of different and often independent dimensions. Dimensions have associated with them hierarchies that specify different aggregation levels of data and hence different granularities of viewing data. Many relational OLAP systems use the star schema [12] to represent the multidimensional data model. A multidimensional database organized as a star consists of a fact table and a table for each dimension. A dimension table comprises attributes for each (aggregation) level of the dimension and other (descriptive) attributes that characterize the different levels of the dimension. The fact table stores attributes for each numeric measure and foreign key attributes to the attribute of the finest granularity level of each dimension.

OLAP applications often require fast response time to complex grouping/aggregation queries on enormous quantities of data. Common techniques to improve query performance are materializing views, and making extensive use of clustering and indexing methods. In multidimensional databases these techniques have to be adapted in order to account for the multiple dimensions. Materializing views is an efficient technique when the way to compute a query using the materialized views is known. However, the general problem of answering and optimizing grouping/aggregation queries using multiple materialized views [4, 3, 20] is complex. Another difficulty of the view materializing technique is the optimal selection of views to materialize [11, 19]. View selection becomes even more complex when the query pattern is not known in advance [13]. Lastly, this technique incurs important additional space requirements and intricate algorithms for incrementally maintaining the materialized views [9].

Commercial relational database management systems use mainly multiple one-dimensional indexes, like compound indexes and bitmap indexes, to process OLAP queries that restrict multiple dimensions. The search key in compound indexes is a concatenation of multiple attributes (where the order of attributes matter). Therefore, they are useful for processing only some of the queries that restrict these attributes. Selecting views

*Research supported by the European Commission under the IST Program project EDITH: European Development of Indexing Techniques for Databases with Multidimensional Hierarchies

and compound indexes for materialization for a given query set pattern is a difficult task [10] and depends on the specific query set pattern. Bitmap indexes (and their variants) [16] are very popular because of their compactness and support of star joins. Nevertheless, in many cases, multidimensional access methods (e.g., R-tree) outperform bit-mapped indexing methods [18].

Contribution In this paper we focus on heuristically optimizing OLAP queries in databases that are clustered with respect to multiple hierarchical dimensions using multidimensional access methods. The main contributions are the following:

- We present a multidimensional database architecture based on the star model (called CSB star). The dimension tables are organized using one-dimensional hierarchical clustering and encoding techniques, while the fact table is organized using a multidimensional access method.
- We show how OLAP queries can be easily expressed by the users over a traditional star schema. The CSB star schema is intended to be a storage option only. The query processor rewrites user queries over the CSB star schema.
- We exploit the clustering features of the CSB star schema to efficiently process a class of typical OLAP queries. The expensive star-join operations needed in a traditional star schema can be essentially implemented as multidimensional range restrictions on the fact table and range restrictions on the dimension tables. Supplementary joins are implemented as merge join operations on sorted tables. Grouping operations are performed on partially sorted relations.
- In this context we detect special cases where joins of fact table tuples with tuples from the dimensions can be avoided, and the grouping of the tuples can be performed only once before all join operations.

This work is done in the context of the European IST project "EDITH". In this project we use a multidimensional access method integrated into the kernel of a database management system [17].

Outline The next section reviews related work. Section 3 introduces the basic concepts of multidimensional hierarchical clustering adopted here. In Section 4, the architecture of the multidimensional database is presented. Section 5 describes the class of queries considered, introduces a number of physical operators, and shows how queries in this class can be heuristically optimized by exploiting the clustering scheme of the multidimensional database architecture. Section 7 contains concluding remarks and directions for further work.

2. RELATED WORK

Conventional query optimizers exploit the knowledge about the group-by clause in a query only by including the grouping columns in the list of interesting orders during join enumeration. The group-by operation can be pushed past one or more joins. This early grouping may reduce the query processing cost by reducing the amount of data participating in the joins. Necessary and sufficient conditions for deciding when this transformation is valid are provided in [22]. A generalization of the early grouping transformation, the coalescing grouping transformations allow us (a) to perform early group-by but require additional group-by subsequently that coalesces multiple groups and (b) to deal with the case where not all the aggregating columns are present in the node of the query evaluation plan where an early group-by operator is placed [2]. Different other cases of early grouping and aggregation are studied and categorized in [23], along with their reverse transformations of lazy grouping and aggregation. These latter transformations postpone the application of a grouping operation until after a join, and may reduce the number of input rows to the group-by, if the join is selective. Both directions of transformation are considered during query optimization. Transformations as well as optimization algorithms for queries with aggregate views and queries containing aggregate nested subqueries are presented in [3]. The proposed pull-up transformation (the equivalent of lazy grouping and aggregation) makes it possible to reorder relations that belong to different query blocks so that these relations can be joined before the group-by operators are applied. The generalized projection operator, an extension of the duplicate eliminating projection operator, captures the semantics of group-by, aggregation, duplicate-eliminating projection and duplicate preserving-projection in a common unifying framework [8]. In this framework query rewriting rules are able to push aggregation operators past selection conditions (and vice-versa). The pull-up transformation does not apply in the evaluation plans that we consider here because the join operations do not reduce the number of tuples of the joining tables. In contrast, a coalescing grouping transformation can be very efficiently exploited: an early grouping can be pushed past all joins. It is worth noting that this is due to the architecture of the CSB star schema that uses hierarchical clustering and encoding techniques, and does not apply to a traditional star join schema. Related works dealing with multidimensional access methods and multidimensional hierarchical clustering are cited in the next section.

3. MULTIDIMENSIONAL HIERARCHICAL CLUSTERING

We present in this section the basic concepts of multidimensional hierarchical clustering and range query processing adopted in the paper.

Multidimensional clustering and the UB-tree. A tuple of a relation in a relational database can be viewed as a point in a multidimensional space where the dimensions are determined by the attributes of the tuple. In this context, the processing of queries can be supported by *multidimensional access methods* [6]. OLAP queries often impose restrictions on multiple attributes (dimensions). Multidimensional access methods are used to cluster data with respect to multiple dimensions. *Multidimensional clustering* can substantially speed up queries that restrict multiple dimensions. The main problem in the design of multidimensional access methods is that there exists no total ordering among the points in the multidimensional space that preserves spatial proximity. One way to heuristically deal with this problem is

to discover a total order that preserves spatial proximity to some extent. This total order is called *space-filling curve*. Then, a one-dimensional access method can be used in combination with the space-filling curve to improve the access of the points in the space. Such a solution to the problem is provided by the *UB-tree* [1].

Range queries on UB-trees and the Tetris algorithm. A query that restricts all attributes (dimensions) to an interval is called *range query*. The multidimensional interval determined by the one-dimensional intervals is called *query box*. In order to answer a range query, [1] presents an algorithm for UB-trees that fetches from the disk only those regions (pages) that properly intersect the query box. The *Tetris algorithm* [15] is a generalization of the multidimensional range query algorithm that efficiently combines sort operations with the evaluation of multi-attribute restrictions. The Tetris algorithm takes as input an attribute of a relation and a query box determined by the restrictions of a query on the relation, and returns the tuples of the relation satisfying the restrictions, ordered on the input attribute. Compared to the access methods of commercial systems for queries in the TPC-D benchmark [21], the Tetris algorithm shows significant speedups, important temporary storage requirement reduction for the sorting process, and multiple times faster production of the first results of a sort operation [15].

Dimension hierarchies. A *dimension hierarchy* D of depth k is a list $L^k, \ldots, L^1$ of k names which are called *dimension or hierarchy levels*. With every level L in D, a non-empty finite set of values $dom(L)$ is associated through the function dom such that $dom(L^i) \cap dom(L^j) = \emptyset,\ i \neq j$. In each dimension, we additionally assume an auxiliary level L^{k+1} whose domain contains the single value *all*. For every two levels $L^i,\ L^{i+1},\ i \in [1,k]$, a function *parent* from $dom(L^i)$ onto $dom(L^{i+1})$ is defined. For every two levels $L^{i+1},\ L^i,\ i \in [1,k]$, a function *children* from $dom(L^{i+1})$ to the power set of $dom(L^i)$ is defined: $children(v)$ is the set of values v' in $dom(L^i)$ such that $parent(v') = v$. Clearly, if $v, v' \in dom(L^i),\ i > 1$, and $v \neq v'$, $children(v) \neq \emptyset$, and $children(v) \cap children(v') = \emptyset$. A value in $dom(L^i),\ i > 1$, represents a set of values in $dom(L^1)$ through the function *children*. Level L^1 (the lowest level in the hierarchy of dimension D) corresponds to the smallest (finest) granularity of viewing data. Level L^{k+1} (the top level in the hierarchy of dimension D) corresponds to the largest granularity of viewing data. A dimension hierarchy defines a hierarchy tree: the nodes of the tree are the values in $\bigcup_{i=1}^{k+1} dom(L^i)$, while the edges are determined by the *parent* function. The leaf nodes of the tree are the values in $dom(L^1)$. The root node is the unique value of $dom(L^{k+1})$. The *path* of a node (value) $n^i \in dom(L^i),\ i \in [1,k]$, is defined to be the concatenation of the nodes $n^k, n^{k-1}, \ldots, n^i$ on the path in a hierarchy tree from the root node to n^i.

Hierarchy clustering and encoding. OLAP queries impose restrictions on different levels of the dimension hierarchies. Hierarchy clustering and encoding [24, 14] in combination with multidimensional access methods can be used to speed up these queries and to optimize the storage usage.

In order to take into account dimension hierarchies in the clustering of data, instead of the values v of the lowest level of a dimension, their path p in the hierarchy tree of the dimension is considered. The concatenated values can be shortened using the following encoding schema which is quite similar to that of [14]. Let v be a value in the domain of level $L^i,\ i \in [1,k]$. Let also v' be the parent of v in the hierarchy tree, V be the cardinality of $children(v')$, and $<_Q$ be the "less than" comparison operator of the query language. We define a one-to-one function $S : children(v') \rightarrow [0, V-1]$ such that: for every $u, u' \in children(v'),\ u <_Q u'$ implies $S(u) < S(u')$. $S(v)$ is called the *surrogate of* v. Note that if $<_Q$ is not defined in the query language for the values in $dom(L^i)$, S is simply defined as a one-to-one function from $children(v')$ onto $[0, V-1]$.

Let v be a value in $\bigcup_{i=1}^{k} dom(L^i)$. Then, the *compound surrogate of* v, $C(v)$, is the path of v where the concatenated values are replaced by their surrogates. One way to compactly store compound surrogates is as fixed length strings of bits: for each level $L^i,\ i \in [1,k]$, in the hierarchy, the maximum surrogate x is defined as $max\{V \mid \text{V is the cardinality of } children(v) \text{ and } v \in dom(L^{i-1})\}$. Then, at least $\lceil \log_2 x \rceil$ bits are reserved in the binary representation of the compound surrogate for the surrogates of level L^i. The total number of bits reserved for a level L^i is called *spread* of L^i.

4. THE CSB STAR SCHEMA

We present in this section the architecture of our multidimensional database. The schema of the multidimensional database is a star with one fact table F and dimension tables $D_1, \ldots, D_n$.

The dimension tables. The schema of a dimension table D_i corresponding to a dimension hierarchy D_i of depth k_i consists of:

(a) A set $\mathbf{H}_i$ of *hierarchy attributes* $\{H_i^1, H_i^2, \ldots, H_i^{k_i}\}$ that correspond one-to-one to the k_i levels of the dimension hierarchy; H_i^1 corresponds to the lowest level in the hierarchy and $H_i^{k_i}$ to the highest.

(b) A set $\mathbf{F}_i$ of *feature attributes* that provide descriptive characterizations of the different levels of the dimension. A feature attribute F_i^j characterizes the hierarchy attribute H_i^j. Feature attributes are optional in the schema of a dimension table.

(c) A *compound surrogate* attribute C_i.

If t is a tuple in table D_i, $t[H_i^j] \in dom(H_i^j),\ j \in [1,k_i]$, and $parent(t[H_i^j]) = t[H_i^{j+1}],\ j \in [1, k_i - 1]$. $t[C_i]$ is the compound surrogate of $t[H_i^1]$.

By the definition of the compound surrogate of a value, it is clear that a point restriction on a hierarchy attribute of a dimension table can be expressed as a single range restriction on the compound attribute of this dimension table.

The compound surrogate attribute C_i is the primary key of the dimension table D_i. By the definition of a dimension hierarchy, H_i^1 is also a key of table D_i, and the following functional dependencies between hierarchy attributes hold on D_i: $H_i^j \rightarrow H_i^{j+1},\ j \in [1, k_i -$

1]. Furthermore, the following functional dependencies between hierarchy and features attributes hold on D_i: $H_i^j \rightarrow F_i^j,\ i \in [1, k_i]$, where F_i^j is a feature attribute characterizing the hierarchy attribute H_i^j.

We associate with a dimension table a primary index P_i on C_i, and a secondary (compound) index I_i on $H_i^{k_i}, \ldots, H_i^1, C_i$. Index I_i is important for computing ranges of values on C_i from point or range restrictions on the different hierarchy attributes of D_i.

Since the tuples of D_i are clustered on C_i, range restrictions on C_i can be computed efficiently. This clustering provides also a grouping of the tuples of the dimension table with respect to any hierarchy attribute. This property is particularly useful for evaluating grouping/aggregation queries as those that are extensively used in OLAP applications.

The fact table. The schema of the fact table F consists of:

(a) The compound surrogate attributes $C_1, \ldots, C_n$, one for each dimension table $D_i,\ i \in [1, n]$.

(b) A set of measure attributes $\mathbf{M} = \{M_1, \ldots, M_k\}$.

The set of attributes $C_1, \ldots, C_n$ is the primary key of F. Each C_i in F is a foreign key and refers to attribute C_i of the dimension table D_i. In general, a fact table contains a huge number of tuples. In a traditional star schema, the lower level hierarchy attributes are stored as foreign keys in the fact table. Instead, by storing the compound surrogate attributes as foreign keys in the fact table, we importantly reduce its size. Table F is organized as a UB-tree on the attributes $C_1, \ldots, C_n$. The schema of our multidimensional database is called *Compound Surrogate Based* star schema (*CSB* star for short).

User View. The clustering scheme of the CSB star schema is intended to be a storage option only, without affecting the formulation of queries by the user. User queries are easily formulated on a simple star schema (called *user star schema*) as this is defined by the view FT for the fact table, and the views $DT_1, \ldots, DT_n$ for the n dimension tables:

```
CREATE VIEW FT AS
SELECT      H_1^1, ..., H_n^1, M
FROM        F, D_1, ..., D_n
WHERE       F.C_1 = D_1.C_1, ..., F.C_n = D_n.C_n
```

```
CREATE VIEW DT_i AS
SELECT      H_i, F_i
FROM        D_i
```

Clearly, the views FT and $DT_1, \ldots, DT_n$ form a typical star schema. User queries formulated on the user schema are rewritten by the query processor over the tables F and $D_1, \ldots, D_n$ by replacing the views by their view definitions. In the following we consider queries that are rewritten over the CSB star schema.

5. OPTIMIZING OLAP QUERIES

We show in this section how to heuristically optimize OLAP queries by exploiting the clustering scheme and the access methods of the multidimensional database architecture.

5.1 The class of queries considered

We consider OLAP queries of the form shown below. Typical cases of OLAP operations can be expressed by this SQL query.

```
SELECT      X, A
FROM        F, D_1, ..., D_n
WHERE       c^J AND c^H AND c^F
GROUP BY    G
HAVING      c^A
ORDER BY    O
```

$\mathbf{X}$ is a set of hierarchy and/or feature attributes (called *projected grouping attributes*). $\mathbf{A}$ is a set of aggregated measures (aggregate functions on measure attributes). Using the catergorization of aggregate functions introduced in [7], we focus on distributive SQL aggregate functions: `min, max, sum, count`. $\mathbf{G}$ is a set of hierarchy and/or feature attributes (called *grouping attributes*). c^J is a conjunction of equi-joining conditions on the compound surrogates. c^H is a conjunction of comparisons involving exclusively hierarchy attributes. c^F is a conjunction of comparisons involving exclusively feature attributes; c^A is a conjunction of comparisons involving exclusively aggregated measures from $\mathbf{A}$. $\mathbf{O}$ is a list of attributes from $\mathbf{X} \cup \mathbf{A}$. The joining conditions are of the form $F.C_i = D_i.C_i$. The comparisons involving an attribute A are of the form $A\ \theta\ c$ where θ is one of the comparison operators $<, \leq, =, \geq, >$, and c is a constant value.

We assume that at least one hierarchy attribute from each dimension is involved in a comparison in the `WHERE` clause of the query. We also assume that if comparisons of the form $H_i^j\ \theta\ c$, where $\theta \in \{<, \leq, \geq, >\}$ appear in the `WHERE` clause of a query, then the *parent* function of the dimension hierarchy D_i is *monotone*. A *parent* function is monotone if for every $v, v' \in dom(H_i^j),\ j = 1, \ldots, k_i$, if $v \leq v'$ then $parent(v) \leq parent(v')$. This assumption guarantees that a range restriction on a hierarchy attribute of a dimension table can be expressed as a single range restriction on the compound surrogate of the table. In this section we consider queries whose hierarchy and feature attribute restrictions can be expressed as a multidimensional range restriction on the compound surrogate attributes of all the dimensions tables (i.e., a single query box).

5.2 Physical operators

In order to construct an OLAP query evaluation plan, we use a number of physical operators that are presented below. Some of them are the traditional relational operators and the others are specific to the organization of the multidimensional database.

By abuse of notation, we view a compound surrogate attribute C_i as a composite attribute consisting of *surrogate attributes* $S_i^{k_i}, \ldots, S_i^1$. If t is a tuple in the dimension table D_i, $t[S_i^j], j \in [1, k_i]$, is the binary representation of the surrogate of $t[H_i^j]$ ($t[S_i^j] = S(t[H_i^j])$). $t[S_i^j]$ is part of $t[C_i]$ and its length in bits is equal to the spread of attribute H_i^j.

We denote by $P_{\mathbf{X}}$ the projection with duplicate retention operator on the set of attributes $\mathbf{X}$, and by $\Pi_{\mathbf{X}}$ the

set-theoretic projection operator (SELECT DISTINCT) on $\mathbf{X}$. σ_c denotes the selection operator with selection condition c. $S[\mathbf{Y}]$ denotes the sorting operator on the list of attributes $\mathbf{Y}$. $\bowtie_{\mathbf{Y}}$ denotes the natural join operator on the list of attributes $\mathbf{Y}$. The natural join operator is applied on tables that are sorted on the list of attributes $\mathbf{Y}$ and is implemented as merge join. $\pi_{\mathbf{X},\mathbf{A}}$ denotes the generalized projection operator [8] (grouping/aggregation operator), where $\mathbf{X}$ is a set of grouping attributes and $\mathbf{A}$ is a set of aggregate functions on measure attributes.

The *Tetris operator*, denoted $T[[l_1, u_1], \ldots, [l_n, u_n]; C_i]$, $i \in [1, n]$, can be applied to the fact table and represents the Tetris algorithm. $[l_j, u_j]$, $j \in [1, n]$, is a range of values for the compound surrogate attribute C_j, and C_i is the compound surrogate attribute on which the resulting table is ordered (refer to Section 3). The schema of the resulting table is that of the fact table.

The *Range operator*, denoted $R[c_i^H; B]$ can be applied to the compound index I_i of a dimension table D_i. c_i^H is a condition of the type described in Subsection 5.1, and involves hierarchy attributes of dimension table D_i. B is a boolean variable that takes values $'1'$ and $'0'$. $R[c_i^H; 0]$ on I_i returns a range of values $[l_i, u_i]$ on C_i such that restricting C_i of D_i in this range is equivalent to applying the restriction c_i^H to D_i. $R[c_i^H; 1]$ on I_i returns, besides $[l_i, u_i]$, a table T. The schema of T is $H_i^{k_i}, \ldots, H_i^1, C_i$ and its content is the set of tuples t over $H_i^{k_i}, \ldots, H_i^1, C_i$ in I_i such that $l_i \leq t[C_i] \leq u_i$.

5.3 Construction of the evaluation plan

We show now how to optimize OLAP queries of the type presented in Subsection 5.1. Our approach is heuristic and is not based on a specific cost model. For ease of presentation we use an example query that is general enough to encompass different optimization cases, and we show how our technique can be applied to produce an evaluation plan.

EXAMPLE 5.1. We consider the following query defined over a four dimensional schema.

SELECT	$F_1^3,\ H_2^2,\ H_3^3,\ F_3^3,\ \mathtt{sum}(M)$
FROM	$F,\ D_1,\ D_2,\ D_3,\ D_4$
WHERE	$F.C_1 = D_1.C_1$ AND $F.C_2 = D_2.C_2$ AND $F.C_3 = D_3.C_3$ AND $F.C_4 = D_4.C_4$ AND c_1^H AND c_2^H AND c_3^H AND c_4^H AND c_3^F
GROUP BY	$F_1^3,\ H_2^2,\ H_3^3,\ F_3^3$
HAVING	c^A
ORDER BY	$H_3^3,\ H_2^2$

□

A query evaluation plan for this query that takes advantage of our multidimensional database architecture is shown in Figure 1. The expensive star-join operations required in a traditional star schema are here essentially implemented by a multidimensional range restriction on the fact table [14]. The presence of the compound surrogate attributes in the fact table allows for an early grouping and aggregation operation. This operation is facilitated by the fact that the selected fact table tuples are retrieved sorted on a compound surrogate attribute. The evaluation plan comprises two kinds of nodes: operation nodes representing operations, and data nodes representing input data, and intermediate and final results. An operation node is depicted by a small circle and is labeled by the operation(s) it represents. Some operations may be pipelined in which case the corresponding temporary results are not actually stored on the disk. The fact table and the dimension tables are depicted by bigger circles, the compound indexes by rectangles, and the ranges of computed compound surrogate attribute values by triangles. Before discussing, in the following, the different steps of the evaluation plan, we provide some definitions.

DEFINITION 5.1. A (hierarchy or feature) attribute is called *restricted attribute* if it is involved in a selection condition (c^F or c^H) in the query.

An attribute is called *imported attribute* if it is a projected grouping hierarchy attribute or a grouping feature attribute in the query. A dimension containing imported attributes is called *joining dimension*. □

An imported attribute needs to be added to selected fact table tuples. A grouping hierarchy attribute that is not projected in the query need not be added to the fact table tuples since the compound surrogate attributes can be used for performing the grouping operation. For each joining dimension, a join operation is needed in order to add the imported attributes of the dimension to selected fact table tuples.

DEFINITION 5.2. A joining dimension D_i is called *candidate last joining dimension* if one of the following two conditions hold:

(a) The first attribute in the list of sorting attributes in the query (the list $\mathbf{O}$ of attributes in the ORDER BY clause of the query) is a hierarchy attribute of D_i.

(b) The first attribute in the list of sorting attributes in the query is not a hierarchy attribute (or there is no such a list), there is a grouping feature attribute in the query, and there is a grouping hierarchy attribute of D_i in the query. □

Selecting the dimension involved in the last join operation to be a candidate last joining dimension simplify the final grouping and sorting operations.

Computing ranges of compound surrogate values The first step in the construction of the query evaluation plan is the application of the Range operator $R[c_i^H; B]$ to the compound index I_i of each dimension. $R[c_i^H; B]$ computes the lower and the upper bound of the range of values $[l_i, u_i]$ of the compound surrogate using the comparisons in c_i^H and the compound index I_i. The range $[l_i, u_i]$ is always provided as input to the Tetris operator. It can also be used for a selection operation on the dimension table D_i. If only hierarchy (and no feature) attributes from a dimension are involved in a query, then their values can be obtained directly from the compound index I_i, without accessing the dimension table D_i. In this case the parameter B of $R[c_i^H; B]$ is set to $'1'$. In general the application of the range operator follows the rules below:

(a) If there is no imported attribute or restricted feature attribute of the dimension in the query then

B is set to $'0'$ and the computed range is used only by the Tetris operator.

(b) If the imported attributes of the dimension are only hierarchy attributes and there is no restricted feature attribute of the dimension in the query then B is set to $'1'$. The computed range is provided to the Tetris algorithm only, while the set of tuples retrieved from I_i are used in a subsequent join operation.

(c) If the imported attributes of the dimension include a feature attribute or there is a restricted feature attribute of the dimension in the query then B is set to $'0'$. The computed range is provided both to the Tetris algorithm and to a selection operation on the corresponding dimension table.

Restricting the dimension tables If the imported attributes of the dimension include a feature attribute or there is a restricted feature attribute of the dimension in the query, the dimension table needs to be accessed. The primary index on the compound surrogate attribute C_i of the dimension is used to retrieve the tuples of the dimension table that fall within the range of values computed by the range operator. Those of these tuples that satisfy the condition c_i^F are retained projected over an appropriate set of attributes. If this computation returns an empty set of tuples, the whole processing of the query is ended since the answer is an empty set. Otherwise, the computed tuples are subsequently joined with tuples derived from the fact table. The set $\mathbf{S}$ of attributes of D_i to be projected are determined as follows:

(a) $\mathbf{S}$ includes all the imported attributes of D_i, and

(b) From the surrogate attributes $S_i^{k_i}, \ldots, S_i^1$ of the compound surrogate C_i, $\mathbf{S}$ includes the surrogate attributes $S_i^{k_i}, \ldots, S_i^j$, where j is the minimal level of the imported attributes of D_i.

If the minimal level j is high enough in the dimension hierarchy this duplicate elimination projection is expected to significantly reduce the number of selected tuples.

It is important to note that the tuples from the dimension table are retrieved sorted on the compound surrogate attribute C_i. Since C_i is the concatenation of the surrogate attributes $S_i^{k_i}, \ldots, S_i^1$, these tuples are also sorted with respect to the list of surrogate attributes $S_i^{k_i}, \ldots, S_i^j$. As a consequence, the elimination of duplicates required for the projection operation can be performed without extra cost. Furthermore, the sort order of the output tuples can be exploited in a subsequent merge join operation with tuples derived from the fact table.

Multidimensional range selection and sorting The Tetris operator takes the ranges of values on the compound surrogate attribute computed in the first step, and retrieves from the fact table the qualifying tuples sorted on a compound surrogate attribute C_i. In general, the choice of the sorting attribute C_i does not affect the performance of the Tetris operator. We assume that the cache memory requirements of the Tetris algorithm are satisfied by the available main memory [15]. The sorting attribute C_i for the Tetris algorithm is chosen from a dimension according to the following rules:

(a) C_i is chosen from a joining dimension that is not a candidate last joining dimension.

(b) If C_i cannot be chosen by the rule (a), it is chosen from a joining dimension.

(c) If C_i cannot be chosen by rules (a) or (b), it is chosen from a dimension that has a hierarchy attribute involved as a grouping attribute in the query.

(d) If C_i cannot be chosen by the previous rules, it is chosen arbitrarily.

Processing of the selected fact table tuples The presence of the compound surrogate attributes $C_1, \ldots, C_n$ in the fact table, allows for an early grouping and aggregation of the fact table tuples resulting by the Tetris algorithm [5]. This operation can be performed efficiently due to the fact that the tuples resulting by the Tetris operation are already sorted (and thus grouped) with respect to one compound surrogate attribute C_i, $i \in [1, n]$. The set of attributes on which the grouping is performed comprises the surrogate attributes $S_i^{k_i}, \ldots, S_i^j$ from each dimension D_i that contains a grouping attribute in the query. j is the minimal level of the (hierarchy and feature) grouping attributes of D_i in the query. Usually, in OLAP applications, a fact table contains a huge number of tuples which are grouped to produce a small number of aggregated results. Therefore, this early grouping operation is expected to drastically reduce the number of fact table tuples at an early stage of their processing.

If there is no grouping feature attribute in the query then the final grouping and aggregation operation that will be presented in the next step is not needed. The reason is that the compound surrogate attributes have already be used instead, for grouping on hierarchy attributes. In this case the selection operation on the aggregated measures (σ_{c^A}) can follow immediately after the early grouping operation to further reduce the number of aggregated fact table tuples that are left to be processed.

If there is at least one joining dimension, one of them has provided its compound surrogate attribute as a sorting attribute to the Tetris operator. In this case, the aggregated tuples are equi-joined on the common attributes with the selected tuples of this dimension. Since both sets of tuples are sorted on their common attributes, a merge join algorithm can be efficiently applied. Each aggregated tuple joins with exactly one tuple from the dimension table. This operation does not alter the number of tuples resulting by the grouping/aggregation operation. It does add to these tuples the imported attributes of the joining dimension. Grouping hierarchy attributes of the dimension table that are not projected grouping attributes in the query are not needed since the fact table tuples have already been grouped using the surrogate attributes.

A similar operation is performed on the resulting tuples for each other joining dimension. This operation has to be preceded by a sort operation with respect to the joining attributes, and by a project operation that eliminates the attributes not needed in subsequent operations. The sort operation has to be performed only on the tuples resulting from the fact table since the di-

mension tuples are already sorted with respect to the joining attributes. The order of the join operations does not significantly matter. The only rule that has to be respected is to choose a candidate last joining dimension for the final join operation. This way the sort order of the resulting tuples can be exploited in the subsequent operations.

Final grouping/aggregation and sorting In the last step of the evaluation plan the tuples resulting by the last join operation are grouped and aggregated. The aggregated tuples that satisfy the condition c^A on aggregated measures are retained projected over the attributes required in the output. As mentioned previously, these operations can be avoided when there is no grouping feature attribute in the query. If the output tuples are required sorted, a final sorting operation is also performed. These operations exploit the sort order of the tuples resulting from the previous step.

6. CONCLUSION

OLAP applications view data structured in multiple hierarchically organized dimensions. Complex grouping/aggregation queries for OLAP applications process enormous quantities of data and require fast response time. Recent research suggests that multidimensional access methods outperform one-dimensional indexing techniques.

We have presented the CSB star, an architecture for a multidimensional database that is based on the star schema. This architecture uses one-dimensional hierarchical clustering and encoding techniques to organize the dimension tables and multidimensional access methods to organize the fact table. Users can express their queries over a traditional star schema, which are then rewritten by the query processor over a CSB star schema. We have shown how the features of this schema allow the heuristic optimization of a class of typical OLAP queries: expensive star-join operations are essentially reduced to multidimensional and one-dimensional range restrictions, supplementary joins are implemented as merge join operation on sorted tables, and grouping operations are performed on partially sorted data. We have detected special cases where supplementary joins are avoided, and a grouping operation can be pushed past all join operations.

An interesting extension of the present work concerns considering a larger class of queries. In particular relaxing a number of the restrictions adopted here can result in queries that determine multiple query boxes on the compound surrogate attributes. Our results apply to this case too by considering the different query boxes separately. However, a cost based optimization technique that considers different groupings of these query boxes is expected to provide further improvements.

7. REFERENCES

[1] R. Bayer. The Universal B-Tree for Multidimensional Indexing: General Concepts. In *Proc. of the Intl. Conf. on World-Wide Computing and its Applications, Springer, LNCS 1274*, pages 98–112, 1997.

[2] S. Chaudhuri and K. Shim. Including Group-By in Query Optimization. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, pages 354–366, 1994.

[3] S. Chaudhuri and K. Shim. Optimizing Queries with Aggregate Views. In *Proc. of the 5th Intl. Conf. on Extending Database Technology*, pages 167–182, 1996.

[4] S. Dar, H. V. Jagadish, A. Y. Levy, and D. Srivastava. Answering SQL Queries with Aggregation using Views. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases*, pages 318–329, 1996.

[5] K. Elhardt. EDITH query processing. Technical report, Transaction Software, Jan 2001.

[6] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2), 1997.

[7] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In *Proc. of the 12th ICDE*, 1996.

[8] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-Query Processing in Data Warehousing Environments. In *Proc. of the 21st Intl. Conf. on Very Large Data Bases*, pages 358–369, 1995.

[9] A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques and applications. *Data Engineering*, 18(2):3–18, 1995.

[10] H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index Selection for OLAP. In *Proc. of the 13th Intl. Conf. on Data Engineering*, pages 208–219, 1997.

[11] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. In *Proc. of the ACM SIGMOD Intl. Conf.*, 1996.

[12] R. Kimball. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996.

[13] Y. Kotidis and N. Roussopoulos. DynaMat: A Dynamic View Management System for Data Warehouses. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 371–382, 1999.

[14] V. Markl, F. Ramsak, and R. Bayer. Improving OLAP Performance by Multidimensional Hierarchical Clustering. In *Proc. of the Intl. Database Engineering and Applications Symposium*, pages 165–177, 1999.

[15] V. Markl, M. Zirkel, and R. Bayer. Processing Operations with Restrictions in RDBMS without External Sorting: The Tetris Algorithm. In *Proc. of the 15th Intl. Conf. on Data Engineering*, pages 562–571, 1999.

[16] P. O'Neil and D. Quass. Improved Query Performance with Variant Indexes. In *Proc. of the ACM SIGMOD Intl. Conf.*, pages 38–49, 1997.

[17] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, and R. Bayer. Intergating the UB-Tree into a Database System Kernel. In *Proc. of the 26st Intl. Conf. on Very Large Data Bases*, pages 263–272, 2000.

[18] S. Sarawagi. Indexing OLAP Data. *Data Engineering*, 20(1):36–43, 1997.

[19] D. Theodoratos and T. Sellis. Data Warehouse Configuration. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, pages 126–135, 1997.

[20] D. Theodoratos and T. Sellis. Answering Queries on Cubes Using Other Cubes. In *Proc. of the Intl. Conference on Scientific and Statistical Databases*, pages 109–122, 2000.

[21] TPC benchmark D. Technical report, Transaction Processing Performance council, 1997.

[22] W. Yan and P.-Å. Larson. Performing Group-By before Join. In *Proc. of the 10th Intl. Conf. on Data Engineering*, pages 89–100, 1994.

[23] W. Yan and P.-Å. Larson. Eager Aggregation and Lazy Aggregation. In *Proc. of the 21st Intl. Conf. on Very Large Data Bases*, pages 1–13, 1995.

[24] C. Zou, B. Salzberg, and R. Ladin. Back to the future: Dynamic hierarchical clustering. In *Proc. of the 14th Intl. Conf. on Data Engineering*, pages 578–587, 1998.

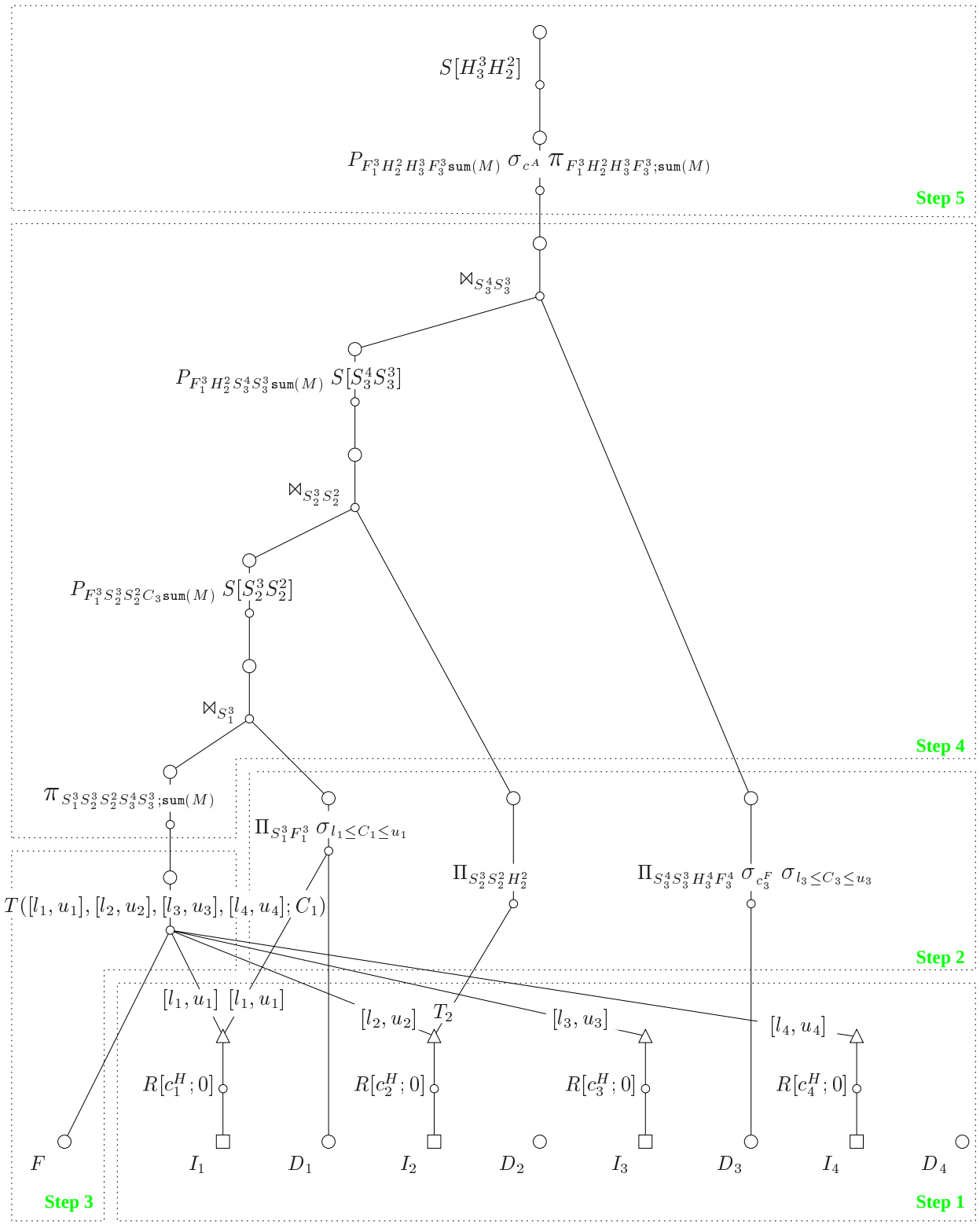

Figure 1: A query evaluation plan