Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Information Sciences 152 (2003) 121–138

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

An aggregation algorithm using a multidimensional file in multidimensional OLAP

Young-Koo Lee [a,*], Kyu-Young Whang [a], Yang-Sae Moon [a], Il-Yeol Song [b]

[a] *Department of Computer Science and Advanced Information Technology Research Center (AITrc), Korea Advanced Institute of Science and Technology (KAIST), 373-1, Kusong-Dong Yusong-Gu, Taejon 305-701, South Korea*

[b] *College of Information Science and Technology, Drexel University, Philadelphia, PA 19104, USA*

Received 30 July 2001; accepted 8 October 2001

Communicated by Ahmed Almargarmid

Abstract

Aggregation is an operation that plays a key role in multidimensional OLAP (MOLAP). Existing aggregation methods in MOLAP have been proposed for file structures such as multidimensional arrays. These file structures are suitable for data with uniform distributions, but do not work well with skewed distributions. In this paper, we consider an aggregation method that uses dynamic multidimensional files adapting to skewed distributions. In these multidimensional files, the sizes of page regions vary according to the data density in these regions, and the pages that belong to a larger region are accessed multiple times while computing aggregations. To solve this problem, we first present an aggregation computation model that uses the new notions of *disjoint-inclusive partition* and *induced space filling curves*. Based on this model, we then present a dynamic aggregation algorithm. Using these notions, the algorithm allows us to maximize the effectiveness of the buffer—we control the page access order in such a way that a page being accessed can reside in the buffer until the next access. We have conducted experiments to show the effectiveness of our approach. Experimental results for a real data set show that the algorithm reduces the number of disk accesses by

* Corresponding author. Fax: +82-42-869-3510.

E-mail addresses: yklee@mozart.kaist.ac.kr (Y.-K. Lee), kywhang@mozart.kaist.ac.kr (K.-Y. Whang), ysmoon@mozart.kaist.ac.kr (Y.-S. Moon), song@drexel.edu (I.-Y. Song).

0020-0255/03/$ - see front matter
doi:10.1016/S0020-0255(03)00077-X

up to 5.09 times compared with a naive algorithm. The results further show that the algorithm achieves a near optimal performance (i.e., normalized I/O = 1.01) with the total main memory (needed for the buffer and the result table) less than 1.0% of the database size. We believe our work also provides an excellent formal basis for investigating further issues in computing aggregations in MOLAP.

1. Introduction

On-line analytical processing (OLAP) is a database application that allows users to easily analyze large volumes of data in order to extract the information necessary for decision-making [4]. OLAP queries make heavy use of aggregation for summarizing data since summarized trends derived from the records are more useful for decision-making rather than individual records themselves. Since computing aggregation is very expensive, good aggregation algorithms are crucial for achieving performance in OLAP systems [1,8,11,19].

OLAP is based on a multidimensional data model that employs multidimensional arrays for modeling data [4]. The multidimensional data model consists of *measures* and *dimensions*: measures are the attributes that are analyzed; dimensions are the attributes that determine the values of the measures. A dimension is mapped to an axis of the multidimensional array. A measure is mapped to a value stored in a cell. This model allows OLAP users to analyze changes in the values of the measures according to changes in the values of the dimensions.

OLAP systems are categorized into two classes according to their storage structures: relational OLAP (ROLAP) and multidimensional OLAP (MOLAP) [4]. ROLAP, built on top of the relational database system, stores OLAP data in tables. In contrast, MOLAP uses multidimensional files that can efficiently store and manage multidimensional data. Recently, as the effectiveness of the multidimensional files on OLAP is recognized, there have been attempts to use them even in ROLAP [8,19].

While aggregation methods for ROLAP have been extensively studied [1,7], the corresponding work for MOLAP has been rare. MOLAP primarily uses static methods that materialize precomputed aggregates [8,16]. However, these methods suffer from storage and periodic update overheads caused by storing all the precomputed aggregate results. To overcome these shortcomings, we need dynamic methods that compute aggregates on the fly when queries are issued.

Dynamic methods for computing aggregates in MOLAP have been limited to a few kinds of multidimensional file structures. Earlier methods use either multidimensional arrays [19] or compressed multidimensional arrays [11]. However, these structures have shortcomings. Multidimensional arrays are inadequate for data with a skewed distribution. Compressed multidimensional

arrays degrade performance for non-aggregate OLAP operators such as range queries by destroying multidimensional clustering.

In this paper, we present a dynamic aggregation method using a multidimensional file that can maintain multidimensional clustering and that adapts to skewed data distributions. We first present an aggregation computation model that employs the new notion of the *disjoint-inclusive partition* for multidimensional files. We will formally define this notion in Section 4.1. We then present a dynamic aggregation algorithm based on this model. We implement the algorithm using the multilevel grid file (MLGF) [17,18], which is a dynamic multidimensional file structure. Experimental results show that performance of our algorithm is excellent compared with a naive algorithm.

The rest of the paper is organized as follows. Section 2 briefly reviews multidimensional files. Section 3 presents the motivation for this research and describes a general method for computing aggregations using multidimensional files. Section 4 proposes our aggregation computation model. Section 5 presents the aggregation algorithm based on our model. Section 6 presents the results of the experiments. Finally, Section 7 concludes the paper.

2. Multidimensional files

We first define some terminology used for multidimensional files [17]. A *file* is a collection of *records*, where a record consists of a list of *attributes*. A subset of these attributes that determines the placement of the records in the file is called the *organizing attributes*. A file has a *multidimensional organization* if it contains more than one organizing attribute. A *domain* of an attribute is a set of values from which an attribute value can be drawn. We define the *domain space* as the Cartesian product of the domains of all the organizing attributes. We call any subset of the domain space a *region*. We call the region allocated to a page P a *page region* and denote it by $\tilde{P}$.

Multidimensional files have the multidimensional clustering property. The property enables efficient multiattribute accesses, which retrieve qualified records using multiple attributes. The multidimensional clustering means that similar records, whose organizing attributes have similar values, are stored in the same page. To support the multidimensional clustering, multidimensional files partition the domain space into regions and store the records in each region on the same page. Thus, the directory represents the state of the domain space partition.

Multidimensional files can be classified into two categories according to the way the boundary value for splitting the region is determined [9]. One uses record-oriented splitting; the other region-oriented splitting. *Record-oriented splitting* divides the region into two subregions so that each subregion has the same number of records. Thus, the boundary value for splitting the region

depends on the record distribution. *Region-oriented splitting* bisects the region regardless of the record distribution. Thus, the boundary values for splitting the region are predetermined independent of the record distribution.

In this paper, we perform experiments using the MLGF [9,17,18], a dynamic multidimensional file structure that uses region-oriented splitting and that adapts well to skewed distributions. The MLGF is a balanced tree and consists of a multilevel directory and data pages. A distinct characteristic of the MLGF is that it uses the *local splitting strategy*, which splits only the region where splitting is required rather than across the entire hyperplane. As a result, in the MLGF, the directory growth is linearly dependent on the number of inserted records regardless of data distributions, data skew, or correlation among the organizing attributes [18]. Thus, the MLGF gracefully adapts to highly skewed and correlated distributions that frequently occur in the OLAP data.

3. Computing aggregates using multidimensional files

In this section, we present our motivation for using multidimensional files in aggregation computation. Section 3.1 defines necessary terminology. Section 3.2 presents a general method for computing aggregates using multidimensional files. Section 3.3 discusses the necessity of buffers in aggregation computation.

3.1. Terminology

Aggregation is an operation that classifies records into groups according to the values of the specified attributes and determines one value per group by applying the given aggregate function [7]. An *aggregate* is the summarized value per group obtained through aggregation. We call the attributes used for grouping records the *grouping attributes*, and the attribute to which the aggregate function is applied the *aggregation attribute*. We define the *grouping domain space* as the Cartesian product of the domains of all the grouping attributes. We call any subset of the grouping domain space a *grouping region*. When we partition the grouping domain space into grouping regions to compute aggregation, we call them *aggregation windows*. We define a *partial aggregation* as aggregation for an aggregation window.

We define the *page grouping region* of a page P, denoted by $R = \prod_G \tilde{P}$, as the projection of the page region $\tilde{P}$ onto the grouping domain space consisting of a set G of grouping attributes. When a page region $\tilde{P}$ and a region Q overlap, we simply say that *the page P and the region Q overlap*. Likewise, when a page grouping region $\prod_G \tilde{P}$ and an aggregation window W overlap, we say that *the page P and the aggregation window W overlap*. We define *aggregation window pages* of an aggregation window W as the pages that overlap with W.

Example 1. Fig. 1 shows an example of computing aggregation in a multidimensional file with three organizing attributes X, Y, and Z. The domain space has been divided into six regions with the records in each region being stored in pages A, B, C, D, E, and F. In the figure, the grouping attributes are X and Y, and the grouping domain space is $X : [0,99] \times Y : [0,99]$. An example of the grouping region is $X : [0,49] \times Y : [0,49]$. The four grouping regions forming a partition of the grouping domain space $X : [0,49] \times Y : [0,49]$, $X : [0,49] \times Y : [50,99]$, $X : [50,99] \times Y : [0,49]$, and $X : [50,99] \times Y : [50,99]$, represented by dashed lines in the figure, are the aggregation windows. Finally, A and E are the aggregation window pages of the aggregation window $X : [0,49] \times Y : [0,49]$.

3.2. A general method for aggregation computation

A simple method to compute aggregation is to use one aggregation window that is equal to the grouping domain space. We first create a result table consisting of entries <values of grouping attributes, aggregate value> in the main memory. We then scan the file and retrieve the records. For each record retrieved, using the values of the grouping attributes, we locate the corresponding entry from the result table, aggregate the value of the aggregation attribute, and store the result in the entry. If there is no corresponding entry, insert a new entry. This method can compute aggregates by scanning a file only once. However, this method would not be effective for a large volume of data because of limited availability of main memory.

An alternative method to compute aggregation is to use multiple aggregation windows that forms a partition of the grouping domain space. The rationale behind this method is that computing aggregates for an aggregation window is independent of those for other aggregation windows. This is because

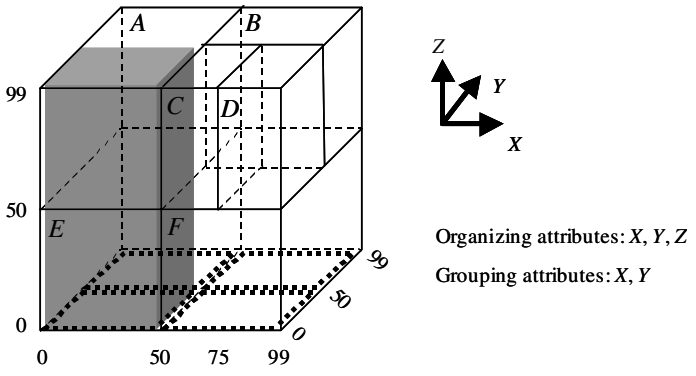

Fig. 1. Computing aggregation in a multidimensional file.

records in different aggregation windows have different values for the grouping attributes forming different groups. If the size of an aggregation window is small, so is the size of the partial aggregation result. Thus, we can choose aggregation windows in such a way that the size of a partial aggregation result can fit in the result table.

The structure of a multidimensional file allows us to efficiently retrieve the records in an aggregation window. This is because multidimensional clustering renders range queries efficient. In contrast, in ROLAP, it is not efficient to retrieve the records that belong to an aggregation window since the table structure in a relational database does not support multidimensional clustering. Therefore, we propose an aggregation algorithm that exploits the clustering characteristic of the multidimensional file.

Fig. 2 shows the algorithm, General_Aggregation, derived from the above principles. Step 1 partitions the grouping domain space into aggregation windows. Step 2 computes the partial aggregation for each aggregation window. Step 2.1 forms the range query for the partial aggregation. In the query, for the grouping attributes, the intervals correspond to the aggregation window; for other organizing attributes, the intervals correspond to the whole domain of each attribute. In Steps 2.2 and 2.2.1, the partial aggregation is computed using the records via the range query. The intermediate result for the partial aggregation is stored in the result table residing in the main memory.

Example 2. We explain General_Aggregation using the example shown in Fig. 1. First, the algorithm partitions the grouping domain space $X : [0, 99] \times Y : [0, 99]$ into aggregation windows $X : [0, 49] \times Y : [0, 49]$, $X : [0, 49] \times Y : [50, 99]$, $X : [50, 99] \times Y : [0, 49]$, and $X : [50, 99] \times Y : [50, 99]$. Then, the algorithm computes the partial aggregation for each aggregation window using the range

Algorithm General_Aggregation

Input: (1) Multidimensional file *md-file* that contains OLAP data
(2) Set G of grouping attributes
(3) Aggregation attribute A

Output: Result of aggregation

Algorithm:

1 Partition the grouping domain space into aggregation windows.
2 For each aggregation window, DO
  2.1 Construct a range query. Here, the query region consists of the intervals corresponding to the aggregation window for the grouping attributes and the entire domain of each attribute for the other organizing attributes.
  2.2 Process the range query against *md-file*.
    2.2.1 For each record retrieved, using the values of the attributes in G, find the corresponding entry from the result table, aggregate the value of the attribute A, and store the result into the entry.

Fig. 2. The general aggregation algorithm General_Aggregation that uses a multidimensional file.

query. For example, the range query for the aggregation window $X : [0,49] \times Y : [0,49]$ is $X : [0,49]$, $Y : [0,49]$, and $Z : [0,99]$, which is represented by the shaded bar in Fig. 1.

When we partition the grouping domain space in Step 1 of General_Aggregation, it is desirable to select aggregation windows so as to make the resulting sizes of partial aggregations similar to one another. This partition reduces the (maximum) size of the result table that must reside in the main memory. We propose a method that selects aggregation windows using a histogram. This method selects aggregation windows by bisecting the grouping domain space recursively until the resulting size of the partial aggregation for each aggregation window is less than the result table size. The number of records in an aggregation window is an upper bound of the size of partial aggregation. To estimate this number, we use a multidimensional histogram for the grouping attributes. Here, we omit the details of the estimation due to space limitation. Details can be found in Ref. [10]. In the remaining part of this paper, we assume that the aggregation windows are given, and focus on Step 2.

3.3. The role of the buffer in aggregation computation

Computing an aggregation with General_Aggregation may cause multiple disk accesses for the same pages. This is because, in Step 2.2 of the algorithm, a page in the multidimensional file is accessed once for each aggregation window that overlaps with the page. In multidimensional files, the sizes of page regions vary because they are determined by the data density in these regions. Hence, the pages that belong to a larger region are more frequently accessed since they tend to overlap with more aggregation windows.

In general, we use the buffer to reduce the number of disk accesses. So can we for the General_Aggregation algorithm. In order to maximize the effectiveness of the buffer, a page should be accessed in such an order that it resides in the buffer until the next access. Therefore, it is desirable to traverse the aggregation windows overlapping with a particular page contiguously. A very interesting observation is that, when we compute aggregations using a multidimensional file, we can achieve such traversal by using the relationships among aggregation windows and page grouping regions. We discuss this issue in Sections 4 and 5.

4. Aggregation computation model based on disjoint-inclusive partition of multidimensional files

The page regions in multidimensional files have various shapes, and thus, the topological relationships among page regions and aggregation windows are

complex. When these relationships have certain properties, we can improve the performance of computing aggregation by taking advantage of them. In this section, we first define the notions of the disjoint-inclusive relationship and the disjoint-inclusive partition. We then define our aggregation computation model that employs these notions for a multidimensional file. We next discuss controlling the page access order such that pages to be accessed multiple times are accessed in contiguous partial aggregations.

4.1. Disjoint-inclusive partition (DIP) in multidimensional files

Definition 1. Two regions S_1 and S_2 satisfy the *disjoint-inclusive* relationship if they satisfy Eq. (1).

$$S_1 \cap S_2 \neq \emptyset \Rightarrow (S_1 \supseteq S_2 \vee S_1 \subseteq S_2). \tag{1}$$

Definition 1 states that if two regions overlap, one includes the other.

Definition 2. Let $\mathbb{D}$ be the domain space with the organizing attributes $A_1, A_2, \ldots, A_n$. A *disjoint-inclusive partition* of $\mathbb{D}$ is a set of regions $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ satisfying the following conditions:

(1) $(\bigcup_{i=1}^{k} Q_i = \mathbb{D}) \wedge (Q_i \cap Q_j = \emptyset, i \neq j)$.

(2) $(\forall G \subseteq \{A_1, A_2, \ldots, A_n\} \forall i, j (1 \leqslant i, j \leqslant k))$ ($\prod_G Q_i$ and $\prod_G Q_j$ satisfy the disjoint-inclusive relationship).

We call a multidimensional file whose page regions form a DIP a *DIP multidimensional file*. Condition (1) of Definition 2 is a necessary and sufficient condition for $\mathcal{Q}$ to be a partition of $\mathbb{D}$. Condition (2) indicates that, when two regions in $\mathcal{Q}$ are projected onto any $\prod_G \mathbb{D}$ space, the projected regions must satisfy the disjoint-inclusive relationship.

Example 3. Fig. 3 illustrates two examples of possible partitions of the domain space in a multidimensional file having three organizing attributes X, Y, and Z. Fig. 3(a) satisfies Condition (1) of Definition 2 since the domain space has been partitioned into six regions $A \sim F$. Furthermore, the projected regions of any two regions in Fig. 3(a) onto the space $\prod_G \mathbb{D}$, where $G = \{X, Y\}$, satisfy the disjoint-inclusive relationship. Since the disjoint-inclusive relationship also holds for any other combination of attributes for G, Fig. 3(a) is a DIP. Fig. 3(b) also satisfies Condition (1) of Definition 2. As shown in this figure, however, in the space $\prod_G \mathbb{D}$ where $G = \{X, Y\}$, $\prod_G A$ and $\prod_G D$ (also $\prod_G A$ and $\prod_G E$) overlap without satisfying the disjoint-inclusive relationship. Therefore, Fig. 3(b) is not a DIP.

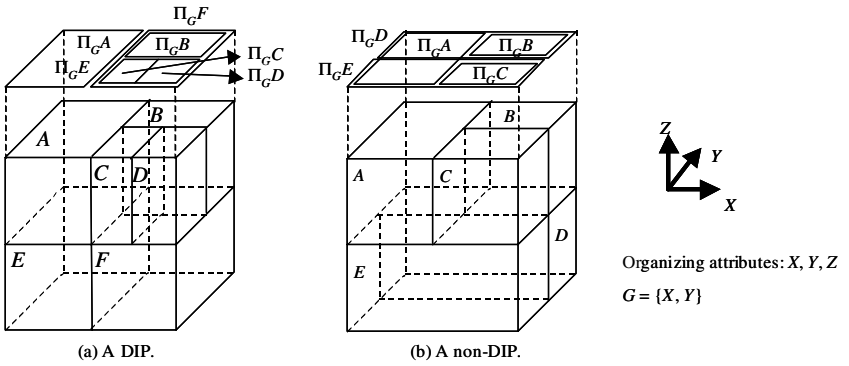

Fig. 3. A DIP and a non-DIP.

Lemma 1 presents a sufficient condition for a multidimensional file to have a DIP.

Lemma 1. *If a multidimensional file satisfies the following splitting rules* (1) *and* (2), *the set of regions in the domain space resulting from the splits forms a DIP.*

(1) *The multidimensional file uses region-oriented splitting.*

(2) *Let* Q_i *and* Q_j *be regions in the domain space and* $SplitAxes(Q_i)$ *and* $SplitAxes(Q_j)$ *be multisets of split axes used in obtaining* Q_i *and* Q_j, *respectively. Then, either* $SplitAxes(Q_i) \subseteq SplitAxes(Q_j)$ *or* $SplitAxes(Q_i) \supseteq SplitAxes(Q_j)$.

Proof. Let $A_1, \ldots, A_n$ be the organizing attributes of the multidimensional file and $\mathcal{Q} = \{Q_1, \ldots, Q_k\}$ the set of regions in the domain space resulting from the splits. Then, for any two regions Q_i and Q_j, the following holds:

(1) Since region-oriented splitting bisects the region upon split, the projections of Q_i's onto the domain of an attribute A_l must be one of those intervals that can be obtained by bisecting the domain of A_l recursively. Thus, the projections $\prod_{A_l} Q_i$ and $\prod_{A_l} Q_j$ satisfy the disjoint-inclusive relationship. Now, let $\mathrm{NS}_{A_l}(Q_i)$ be the number of splits on the attribute A_l necessary for obtaining Q_i. If $\prod_{A_l} Q_i$ and $\prod_{A_l} Q_j$ overlap, one with the smaller value of $\mathrm{NS}_{A_l}(\cdot)$ includes the other.

(2) Since either $\mathrm{SplitAxes}(Q_i) \subseteq \mathrm{SplitAxes}(Q_j)$ or $\mathrm{SplitAxes}(Q_i) \supseteq \mathrm{SplitAxes}(Q_j)$, either $\forall_{A_l}(\mathrm{NS}_{A_l}(Q_i) \leqslant \mathrm{NS}_{A_l}(Q_j))$ or $\forall_{A_l}(\mathrm{NS}_{A_l}(Q_i) \geqslant \mathrm{NS}_{A_l}(Q_j))$ holds.

Now, consider Q_i and Q_j such that $\prod_G Q_i$ and $\prod_G Q_j$ overlap, where G is any subset of $\{A_1, \ldots, A_n\}$. Then, $\prod_{A_l} Q_i$ and $\prod_{A_l} Q_j$ must also overlap for all A_l in G. This fact along with Conditions (1) and (2) leads to the conclusion either

$\forall_{A_l} \in G(\prod_{A_l} Q_i \supseteq \prod_{A_l} Q_j)$ or $\forall_{A_l} \in G(\prod_{A_l} Q_i \subseteq \prod_{A_l} Q_j)$. In other words, either $\prod_G Q_i \supseteq \prod_G Q_j$ or $\prod_G Q_i \subseteq \prod_G Q_j$ holds meaning that one region includes the other. Therefore, $\mathcal{Q}$ forms a DIP. □

We identify a special case satisfying the splitting rule (2) of Lemma 1. Let SplitAxesSeq(Q_i) be a sequence of splitting axes used in obtaining Q_i. Then, a condition that SplitAxesSeq(Q_i) is a prefix of SplitAxesSeq(Q_j) or *vice versa* is a sufficient condition for the splitting rule (2). The cyclic splitting strategy [9], which selects the splitting axis cyclically, is an example that satisfies the condition.

4.2. The aggregation computation model

Definition 3. The *DIP computation model* for computing aggregations using a multidimensional file is the one that satisfies the following four conditions:

(1) A DIP multidimensional file is used.
(2) The aggregation with respect to the grouping domain space is computed as the union of partial aggregations, each of which is computed with respect to an aggregation window. Here, aggregation windows form a partition of the grouping domain space.
(3) Disjoint-inclusive relationship is satisfied among aggregation windows and page grouping regions.
(4) Each partial aggregation is computed by retrieving records through a range query against the multidimensional file.

The DIP computation model computes aggregations using a DIP multidimensional file. Conditions (2) and (4) allow us to compute aggregates by taking advantage of multidimensional clustering. We have discussed such an aggregation method, General_Aggregation, in Section 3.2. Conditions (1) and (3) allow us to use the DIP property. They provide the basis for developing an efficient aggregation algorithm and for analyzing the algorithm formally.

We note that it is always possible to partition the grouping domain space into aggregation windows that have the disjoint-inclusive relationship with page grouping regions as required by Condition (3). Page grouping regions of a DIP multidimensional file mutually satisfy the disjoint-inclusive relationship. Thus, if we select each aggregation window as a union of multiple page grouping regions, aggregation windows have the disjoint-inclusive relationship with the page grouping regions.

4.3. Page access order

Definition 4. Let $R = \prod_G \tilde{P}$ be the page grouping region of a page P. We say that P is an *L-page* (*a large page*) if R properly includes at least one aggregation window.

Our algorithm controls the page access order so that repeatedly accessed pages (L-pages) are accessed in contiguous partial aggregations. The objective is to let an L-page to be accessed all at once while it remains in the buffer so that it does not have to be accessed and loaded into the buffer any further. To achieve this objective, we need to contiguously traverse the aggregation windows that overlap with those L-pages. To obtain such a traversal order, we use the notion of a space filling curve [6] induced from a given set of regions.

Definition 5. For a given set $\mathscr{S} = \{S_1, S_2, \ldots, S_n\}$ of regions in the multidimensional space $\mathbb{D}$, where elements of $\mathscr{S}$ satisfy the disjoint-inclusive relationship, we define the *induced space filling curve* (*ISFC*) as a space filling curve satisfying the following condition:

Condition [*contiguous interval*]: Let *the ISFC value*, ISFC(p), is the value allocated to a point p in $\mathbb{D}$ by the ISFC. For any region S_i, points in S_i map into a contiguous interval of ISFC values. Formally, for any region S_i and any ISFC value v such that $\min_{p\in S_i}\{\mathrm{ISFC}(p)\} \leqslant v \leqslant \max_{p\in S_i}\{\mathrm{ISFC}(p)\}$, a point p satisfying $\mathrm{ISFC}(p) = v$ must be in S_i and *vice versa*.

Here, we call $\mathscr{S}$ the *ISFC basis*. We define the *ISFC value of a region* S_i, ISFC(S_i), as $\max_{p\in S_i}\{\mathrm{ISFC}(p)\}$.

Lemma 2. *For a given set* $\mathscr{S} = \{S_1, S_2, \ldots, S_n\}$ *of regions in the multidimensional space* $\mathbb{D}$, *where elements of* $\mathscr{S}$ *satisfies the disjoint-inclusive relationship*, *there exists at least one ISFC with* $\mathscr{S}$ *as the ISFC basis*.

Proof. Since the regions in $\mathscr{S}$ satisfy the disjoint-inclusive relationship, we can partition $\mathscr{S}$ into subsets $\mathscr{U}_1, \mathscr{U}_2, \ldots, \mathscr{U}_m$ $(1 \leqslant m \leqslant n)$ as follows. First, let $\mathscr{U}_1$ be $\{S_i \in \mathscr{S} | \forall_{S_j\in\mathscr{S}}(S_i \not\subset S_j)\}$. That is, a region S_{1_i} in $\mathscr{U}_1$ is an element of $\mathscr{S}$, but is not properly included by any region $S_j \in \mathscr{S}$. Next, let $\mathscr{U}_2$ be $\{S_i \in \mathscr{S} - \mathscr{U}_1 | \forall_{S_j\in\mathscr{S}-\mathscr{U}_1}(S_i \not\subset S_j)\}$. That is, a region S_{2_i} in $\mathscr{U}_2$ is an element of $\mathscr{S} - \mathscr{U}_1$, but is not properly included by $S_j \in (\mathscr{S} - \mathscr{U}_1)$. We note that each region S_{2_i} in $\mathscr{U}_2$ is properly included in a region in $\mathscr{U}_1$; otherwise, it would be contained in $\mathscr{U}_1$ instead of $\mathscr{U}_2$. In a similar manner, let $\mathscr{U}_l$ $(1 < l \leqslant m)$ be $\{S_i \in \mathscr{S} - (\mathscr{U}_1 \cup \cdots \cup \mathscr{U}_{l-1}) | \forall_{S_j\in\mathscr{S}-(\mathscr{U}_1\cup\cdots\cup\mathscr{U}_{l-1})}(S_i \not\subset S_j)\}$. Then, each region S_{l_i} in $\mathscr{U}_l$ is properly included in a region in $\mathscr{U}_{l-1}$.

Now, we prove the existence of an ISFC with $\mathscr{S}$ as the basis by constructing one ISFC. First, for $S_{m_i} \in \mathscr{U}_m$, we select an arbitrary order that starts at the lower-left corner of S_{m_i}, ends at the upper-right corner of S_{m_i}, and traverses all the points in S_{m_i}. We use this order as a space filling curve for S_{m_i} $(\in \mathscr{U}_m)$ and call it SFC_{m_i}. Next, for $S_{(m-1)_i} \in \mathscr{U}_{m-1}$, we also select an arbitrary order, which starts at the lower-left corner, ends at the upper-right corner, and traverses all regions S_{m_k} that are properly included in $S_{(m-1)_i}$. We use this order as a space filling curve for $S_{(m-1)_i}$. Here, for each $S_{m_k} \in \mathscr{U}_m$ traversed, we insert the order of S_{m_k}, SFC_{m_k}, into that of $S_{(m-1)_i}$, $\mathrm{SFC}_{(m-1)_i}$. In a similar manner, we select an arbitrary order SFC_{l_i} for $S_{l_i} \in \mathscr{U}_l$ $(1 \leqslant l < m)$. Here, for each $S_{(l+1)_k} \in \mathscr{U}_{l+1}$ traversed, we insert the order of $S_{(l+1)_k}$, $\mathrm{SFC}_{(l+1)_k}$, into that of S_{l_i}, SFC_{l_i}. We repeat this procedure until we reach $\mathscr{U}_1$. We now make an SFC by combining all the SFC_{1_i}'s of S_{1_i}'s in $\mathscr{U}_1$. It is trivial to show that the SFC satisfies the condition of ISFC. Therefore, there exists an ISFC that use $\mathscr{S}$ as an ISFC basis. □

Definition 5 indicates that, in an ISFC order, all smaller regions included in a larger region S_i are traversed first, and then, those that are not included in S_i are traversed. We take advantage of of the notion of the ISFC in our aggregation algorithm: when there are smaller aggregation windows overlapping with (i.e., contained in) a larger page grouping region for a page P, we use an ISFC order to make those aggregation windows traversed contiguously, so that the page P may reside in the buffer without swapping.

We now present a method that uses an ISFC as the traversal order of aggregation windows and discuss its characteristics. Let $\mathscr{R}$ be a set of page grouping regions in a DIP multidimensional file and $\mathscr{W}$ a set of aggregation windows, where elements of $\mathscr{R}$ and $\mathscr{W}$ satisfy the disjoint-inclusive relationship. We define the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ as the ISFC in the grouping domain space using $\mathscr{R} \cup \mathscr{W}$ as the ISFC basis.

Example 4. Fig. 4 illustrates examples of an $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ and a non-$\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$. Fig. 4(a) and (b) show page grouping regions (R_i's) and aggregation windows

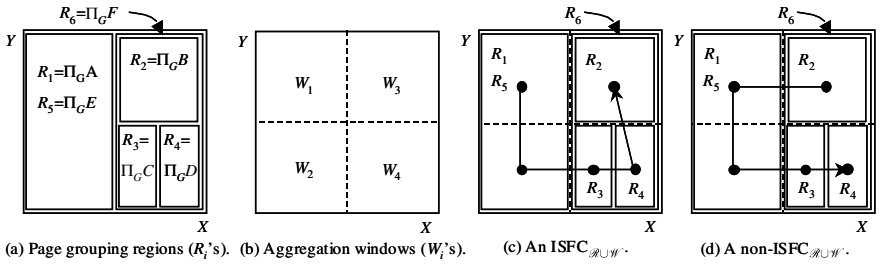

Fig. 4. An $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ and a non-$\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ $(G = \{\mathrm{X}, \mathrm{Y}\})$.

(W_i's), respectively, used in Example 1. Fig. 4(c) and (d) show the overlay of page grouping regions shown in Fig. 4(a) on top of aggregation windows shown in Fig. 4(b). Fig. 4(c) satisfies the definition of the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$: the order in which the regions are traversed is $R_1(S_1 \to S_2) \to R_6(S_4(R_3 \to R_4) \to R_2)$. Fig. 4(d) does not, however, because points in region R_1 are traversed while those in region R_6 still are.

Lemma 3. *Consider computing aggregations under the DIP computation model. Let* $\mathscr{W} = \langle W_1, W_2, \ldots, W_k \rangle$ *be a list of aggregation windows, where* W_i*'s are ordered in an* $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ *order, i.e., if* $i < j$, *then* $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}(W_i) < \mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}(W_j)$. *Then, the aggregation windows that overlap with an L-page are contiguous, i.e., they are* $W_l, W_{l+1}, \ldots, W_h$ $(1 \leqslant l \leqslant h \leqslant k)$.

Proof. Let $R = \prod_G \tilde{P}$ be the page grouping region of an L-page P and $\mathscr{W}' = \langle W_{i_1}, W_{i_2}, \ldots, W_{i_m} \rangle$ $(1 \leqslant m \leqslant k)$ be a list of aggregation windows overlapping with R. Then, $W_{i_j} \subset R$ by the definition of the L-page. This leads to $\bigcup_{j=1}^{m} W_{i_j} \subseteq R$. In addition, since the aggregation windows in $\mathscr{W}$ form a partition of the entire grouping domain space, R must be included in the union of the aggregation windows that overlap with R. This leads to $R \subseteq \bigcup_{j=1}^{m} W_{i_j}$. Thus, $\bigcup_{j=1}^{m} W_{i_j} = R$. In addition, since $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ is an ISFC that uses the union of page grouping regions and aggregation windows as the ISFC basis, all the points in W_{i_j}'s have the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ values between the maximum and minimum of the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ values in R. Therefore, when the aggregation windows are ordered in the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ order, W_{i_j}'s are contiguous. □

By Lemma 3, if we compute aggregations traversing aggregation windows in the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ order, the aggregation windows that overlap with an L-page are processed contiguously. In other words, repeatedly accessed pages are accessed in contiguous aggregation windows.

In an actual implementation, a specific $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ is determined by the intrinsic characteristics of a multidimensional file. We present a way of determining an $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ order based on the splitting rules of Lemma 1. All page grouping regions in $\mathscr{R}$ have been created according to the splitting rules of Lemma 1. In addition, since the set of aggregation windows $\mathscr{W}$ must satisfy the disjoint-inclusive relationship with the page grouping regions, we select the aggregation windows from the regions that can be created according to the splitting rules. These regions are the ones resulting from splitting the domain space recursively. Under these circumstances, we can use as the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ a space filling curve having the recursive property that accords with the order of splitting axes selected. For example, when a multidimensional file uses the cyclic splitting strategy, we can use Z-order [6] as the $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$. We use Z-order in our experiments.

5. DIP aggregation algorithm

In this section we present the aggregation algorithm based on the DIP computation model. Fig. 5 shows the algorithm DIP_Aggregation that extends the algorithm General_Aggregation as follows: (1) Step 1 partitions the grouping domain space into aggregation windows satisfying the disjoint-inclusive relationship with page grouping regions; (2) Step 2 traverses aggregation windows in an $\mathrm{ISFC}_{\mathscr{R}\cup\mathscr{W}}$ order. By Lemma 3, the algorithm DIP_Aggregation lets repeatedly accessed pages be accessed in contiguous aggregation windows and, thus, maximize the effectiveness of the buffer.

6. Performance evaluation

In this section we present the result of the performance evaluation for the DIP_Aggregation algorithm. Section 6.1 explains the experimental environment and data sets. Section 6.2 presents the experimental results.

6.1. The experimental environment and data sets

6.1.1. Data sets

We use both synthetic and real data sets for the experiments. The synthetic data set has 5,000,000 records consisting of six attributes: five are organizing attributes representing the dimensions, and one is the measure. The data types of all the attributes are 4-byte integers, whose domain is $[-2^{31}, 2^{31} - 1]$. To simulate the OLAP data where records are distributed in many clusters, we use a distribution that superposes 100 overlapping multivariate normal distributions for data distributions of the organizing attributes. The ith attribute of the

Algorithm DIP_Aggregation
Input: (1) DIP multidimensional file *md-file* that contains OLAP data
(2) Set G of grouping attributes
(3) Aggregation attribute A
Output: Result of aggregation
Algorithm:
1 Partition the grouping domain space into aggregation windows so that aggregation windows and page grouping regions satisfy the disjoint-inclusive relationship.
2 Traversing aggregation windows in an $\mathrm{ISFC}_{\mathcal{R}\cup\mathcal{W}}$ order, for each aggregation window, DO
2.1 Construct a range query. Here the query region consists of the intervals corresponding to the aggregation window for the grouping attributes and the entire domain of each attribute for the other organizing attributes.
2.2 Process the range query against *md-file*.
2.2.1 For each record retrieved, using the values of the attributes in G, find the corresponding entry from the result table, aggregate the value of the attribute A, and store the result into the entry.

Fig. 5. The algorithm DIP_Aggregation.

multivariate normal distribution has a normal distribution of $N(\mu_i, \sigma^2)$, where the mean μ_i is randomly selected within the range $[-2^{31}, 2^{31} - 1]$, and the standard deviation σ is 2^{28}. We call this data set *SYNTHETIC-DATA*. The size of SYNTHETIC-DATA is 303.8MB (74,169 pages).

The real data set that we use is the Forest Cover Type database from the UCI KDD archive [2]. It has about 581,012 records consisting of 54 attributes, 10 of which are numerical. We use five-dimensional projection of the data set (the projected data set has 580,616 records) and use these five attributes as the dimensions and organizing attributes. These attributes are Elevation, Aspect, Slope, Horizontal_distance_to_hydrology, and Vertical_distance_to_hydrology. We use a dummy attribute as the measure. We call this data set *REAL-DATA*. The size of REAL-DATA is 34.3MB (8784 pages).

6.1.2. Methods of experiments

We have used three grouping attributes among the five organizing attributes. As the DIP multidimensional file storing the OLAP data, we have used the MLGF. [1]

The page size used is 4KB. The result table size is set to 0.1% of the database size, while the buffer size is varied from 0% to 10% of the database size to analyze the effect of the buffer size. We use as the buffer replacement policy the CLOCK policy, which is a simple and widely used approximation of LRU [5]. Since disk I/O has a major effect on the performance of aggregation [7], we use the number of disk accesses that occur during aggregation computation as the performance measure.

6.1.3. Algorithms compared

For providing a basis of evaluation, we compare the DIP_Aggregation algorithm with a straightforward algorithm *Naive_Aggregation* derived from the algorithm General_Aggregation in Section 3.2. The Naive_Aggregation algorithm obtains aggregation windows using the partitioning algorithm of the equi-depth histogram [12] and traverses the aggregation windows using the row-major order. Here, the aggregation windows do not satisfy the disjoint-inclusive relationship with page grouping regions. Thus, Naive_Aggregation does not conform to the DIP computation model.

[1] We use the MLGF for the experiments here, but our methodology is applicable to other kinds of DIP multidimensional files such as the quad tree [14] and the grid file [13]. Other multidimensional files can be made DIP multidimensional files by limiting split policies. The buddy tree [15] is an example. On the other hand, some cannot be made DIP multidimensional files. The R*-tree [3] is an example.

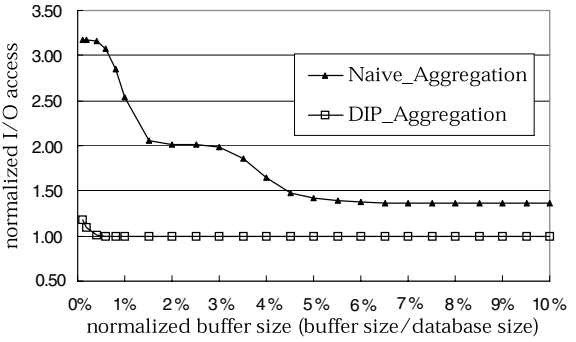

Fig. 6. Normalized I/O access for SYNTHETIC-DATA (74,169 pages).

6.2. Experimental results

Fig. 6 shows the experimental results for SYNTHETIC-DATA. The horizontal axis represents the size of the buffer. The vertical axis represents the *normalized I/O access*, which is defined as the number of page accesses normalized by the total number of pages in the file. Fig. 6 shows that DIP_Aggregation has a far better performance than Naive_Aggregation. The result verifies the effectiveness of our approach of using the DIP computation model where the aggregation windows satisfy the disjoint-inclusive relationship with page grouping regions, and the $ISFC_{\mathcal{R}\cup\mathcal{W}}$ is the traversal order among the aggregation windows.

Fig. 7 shows the experimental results for REAL-DATA. Similar to Fig. 6, Fig. 7 indicates that DIP_Aggregation has a far better performance than Naive_Aggregation. The phenomenon is more marked here since there are more random variation of data distribution in real data. In Fig. 7 we observe

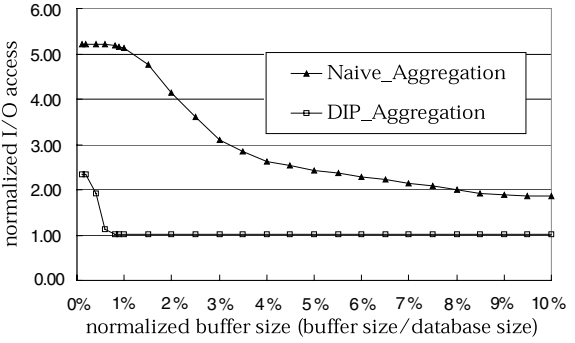

Fig. 7. Normalized I/O access for REAL-DATA (8784 pages).

that DIP_Aggregation reduces the number of disk accesses by up to 5.09 times compared with Naive_Aggregation. An interesting observation is that we can get a near optimal result (i.e., normalized I/O access = 1.01) with less than 1.0% of the database size as the buffer, even with the result table size added.

7. Conclusions

Efficient aggregation algorithms are crucial for achieving good performance in OLAP systems. In this paper, we have presented a dynamic aggregation algorithm that uses multidimensional files in MOLAP. We have presented the new notion of the disjoint-inclusive partition and proposed the aggregation computation model, the DIP computation model, using this notion. In Lemma 3, we have proved that we can maximize the buffering effect by controlling the page access order with the ISFC to process the repeatedly accessed pages (L-pages) in consecutive aggregation windows. Based on the model, we then have proposed the aggregation algorithm, DIP_Aggregation, that computes aggregation using the ISFC.

To verify the performance of DIP_Aggregation, we have performed experiments with the synthetic and real data sets. Experimental results for a real data set show that our algorithm reduces the number of disk accesses by up to 5.09 times compared with Naive_Aggregation. The results also show that we need the main memory less than 1.0% of the size of the database to optimally process the OLAP aggregation. We believe that our work provides an excellent formal basis for investigating further issues in computing aggregations in MOLAP.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References

[1] S. Agarwal, R. Agrawal, P.M. Deshpande, et al., On the computation of multidimensional aggregations, in: Proceedings of the International Conference on Very Large Data Bases, Mumbai (Bombay), India, 1996, pp. 506–521.

[2] S.D. Bay, The UCI KDD Archive, Department of Information and Computer Science, University of California, Irvine, CA, 1999. Available from <http://kdd.ics.uci.edu/>.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Atlantic City, NJ, 1990, pp. 322–331.
[4] S. Chaudhuri, U. Dayal, An overview of data warehousing and OLAP technology, ACM SIGMOD Record 26 (1) (1997) 65–74.
[5] W. Effelsberg, T. Haerder, Principles of database buffer management, ACM Transactions on Database Systems 9 (4) (1984) 560–595.
[6] V. Gaede, O. Günther, Multidimensional access methods, ACM Computing Surveys 30 (2) (1998) 170–231.
[7] G. Graefe, Query evaluation techniques for large databases, ACM Computing Surveys 25 (2) (1993) 73–170.
[8] Y. Kotidis, N. Roussopoulos, An alternative storage organization for ROLAP aggregate views based on cubetrees, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Seattle, Washington, 1998, pp. 249–258.
[9] J. Lee, Y. Lee, K. Whang, Region splitting strategy for physical database design of multidimensional file organizations, in: Proceedings of the International Conference on Very Large Data Bases, Athens, Greece, August 1997, pp. 416–425.
[10] Y. Lee et al., An aggregation algorithm using a multidimensional file in multidimensional OLAP, AITrc Technical Report 01-11-012, Advanced Information Technology Research Center (AITrc), KAIST, Taejon, Korea, June 2001.
[11] J. Li, D. Rotem, J. Srivastava, Aggregation algorithms for very large compressed data warehouses, in: Proceedings of the International Conference on Very Large Data Bases, Edinburgh, Scotland, UK, 1999, pp. 651–662.
[12] M. Muralikrishna, D. DeWitt, Equi-depth histograms for estimating selectivity factors for multi-dimensional queries, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Chicago, Illinois, 1988, pp. 28–36.
[13] J. Nievergelt, H. Hinterberger, K.C. Sevcik, The grid file: an adaptable, symmetric multikey file structure, ACM Transactions on Database Systems 9 (1) (1984) 38–71.
[14] H. Samet, The quadtree and related hierarchical data structures, ACM Computing Surveys 16 (2) (1984) 187–260.
[15] B. Seeger, H.-P. Kriegel, The buddy-tree: an efficient and robust access method for spatial data base systems, in: Proceedings of the 6th International Conference on Very Large Data Bases, 1990, pp. 590–601.
[16] P. Vassiliadis, T. Sellis, A survey of logical models for OLAP databases, ACM SIGMOD Record 28 (4) (1999) 64–69.
[17] K. Whang, R. Krishnamurthy, Multilevel grid files, IBM Research Report RC 11516(51719), 1985.
[18] K. Whang et al., Dynamic maintenance of data distribution for selectivity estimation, The VLDB Journal 3 (1) (1994) 29–51.
[19] Y. Zhao, P.M. Deshpande, J.F. Naughton, An array-based algorithm for simultaneous multidimensional aggregates, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Tucson, Arizona, 1997, pp. 159–170.