Multi-Link Lists as Data Cube Structure in the MOLAP Environment

Fadila Bentayeb, Omar Boussaid, Jérôme Darmont
BDD - ERIC - Université Lumière Lyon 2
Bâtiment L, 5 avenue Pierre-Mendès-France - France
69676 BRON Cedex — FRANCE
bentayeb@eric.univ-lyon2.fr
[boussaid, jdarmont]@univ-lyon2.fr

ABSTRACT
In the area of “On Line Analytical Processing” (OLAP), the concept of multidimensional databases is growing in popularity. Several efficient algorithms for Relational OLAP (ROLAP) have been developed to compute the cube. Multidimensional OLAP (MOLAP) systems present a different challenge in computing the cube. The main difference resides in the data storage structures. ROLAP systems store data in relational tables while MOLAP systems use sparse arrays. In this paper, we aim to provide a new data cube structure whose characteristics are interesting compared to fixed arrays. Indeed, the data cube structure we propose is a set of multi-link lists that is dynamic rather than fixed-sized arrays. It avoids the sparse data problem and presents good performances in terms of space and query response time when compared to arrays.

1 INTRODUCTION
Multidimensional databases and OLAP technology [Cod93] provide efficient solutions to manipulate and aggregate data in databases [CD97]. Several efficient algorithms for ROLAP [AAD+96,RS97] and MOLAP [ZDN97] systems have been developed to compute the cube. The respective data structures used for data storage are fundamentally different. ROLAP systems store their data in relational tables, while MOLAP systems store their data as sparse arrays. For removing the unused storing space and improving the performance of the queries, compression and indexing techniques are used [ZDN97,RS97]. In this paper, our objective is to provide an efficient solution to the problem of sparse data in the multidimensional environment by proposing a new data structure which is a set of dynamic multi-link lists (MLL). With MLL, we avoid the sparse data problem by representing only real data, facilitate navigation through the data for further OLAP operations and query execution time when compared to arrays.

When compared to array structures, MLL displays good performances both in terms of memory space and query execution time. Moreover, adding dimensions or hierarchies without re-building it.

Create view dataset as
Store.SID, Product.PID, Period.Date, Salesperson.SPID, Price
From Store, Product, Period, Salesperson, Sales;

Formally, a data cube is composed by:
- A set of dimensions. Each dimension \(D=\text{Name}_{d}, A_{j}\) is defined by its name \(\text{Name}_{d}\) and a set of values \(A_{j}\) that are called attributes.
- A set of measures. Each measure \(M=\text{Name}_{m}, V_{d}\) is defined by its name \(\text{Name}_{m}\) and a set of possible values \(V_{d}\) (domain values).
- A set of facts called points. Each point is defined by its coordinates (valid combination) and its values.
- A set of hierarchies \(HRU96[JLS99]\). Each hierarchy \(H=(\text{Name}_{h}, P_{h}, n)\) is defined by its name \(\text{Name}_{h}\) the set of its values \(P_{h}\) called parameters and their level in the hierarchy \(n\). A hierarchy of level \(n\) is related to a hierarchy of level \(n-1\) that may be a dimension. Each parameter \(P_{h}(\text{Name}_{h}, P_{h}, n)\) is defined by its name \(\text{Name}_{h}\) and is associated with a set of parameters values \(V_{h}\). \(PV\) is either a subset of \(P_{h}\) where \(H\) is a hierarchy of \(H=(\text{Name}_{h}, P_{h}, n)\) or a subset of \(A_{h}\) where \(H\) is a hierarchy of the dimension \(D=\text{Name}_{d}, A_{j}\). Hierarchies of a dimension are noted \(D \circ H_{1} \circ H_{2} \circ H_{3} \ldots\)

Copyright © 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
3 MULTI-LINK LISTS

The MLL structure includes two types of lists: (1) data lists and (2) metadata lists (Fig 3).

Data Lists

- Lists of Points: Each list represents a point. Each cell in the list contains either a coordinate or a value. The coordinates of a given point in MLL are linked via two pointers: next and previous. Different points may be linked via one pointer (bottom pointer) and represent some facts according to a given attribute dimension. The aggregations are more easily computed because these three pointers allow scanning the lists of points.

- List of Dimensions: It contains the names of the selected dimensions and thus defines the multidimensional space. Any dimension may be selected as a measure. Every cell of the dimension list points to its set of attributes.

- Lists of Dimension attributes: The number of cells in every list of dimension attributes corresponds to the size of the dimension. Every cell of a dimension attribute list is a head pointer to the list of points sharing the same attribute.

Metadata Lists

- Lists of Hierarchies: Every dimension can be summarized according to the different hierarchies and can have several levels of granularity [RS97]. A cell of the hierarchy list points to the hierarchy parameters and to the hierarchy of lower level if any exists, and otherwise to the dimension it summarizes.

- Lists of Hierarchy parameters: Each one contains a set of hierarchy parameters. Every cell of these lists either points to a list of associated parameters of a lower hierarchy, or to a list of attributes of a dimension.

4 ALGORITHMS

To every list of dimension attributes (respectively a list of hierarchy parameters), we add the ALL attribute (respectively the ALL parameter) [GCB+97] to represent the aggregated points.

Procedure CreateMLL (dcx: text file or relational table)
/* This algorithm uses the lists LDims (list of dimensions), LAttDim (list of dimension attributes) and Lcoord (lists of points) */
Begin
For each record in raw data
/* We create a new dimension list and its attributes lists */
If FirstRecord Then LDims.Create
/* We create a new point list to contain this record */
LCoord.Create
/* field denotes an attribute or a measure of the record */
For each field of record
If FirstRecord Then
LDims.Insert(field)
If field is not a measure Then
LDims.LinkBottom(LAttDim)
End if
Else
LCoord.Insert(address of field)
If field is not a measure Then
LAttDim := LDims.GetBottom(field)
/* To get the address of the list pointed by the bottom pointer */
LAttDim := LDims.LinkBottom(LAttDim)
End if
Else
LCoord.Insert(address of field)
End if
Next field
If FirstRecord Then FirstRecord := False
Else
LCoord.Free
End if
Next record
End

For each point to be added in MLL, a new attribute is created in LAttDim if it does not exist and each cell of the point is linked to the corresponding list.

5 PERFORMANCE RESULTS

The performance survey has been achieved on a Pentium III 750 MHz with 128 MB RAM under Windows 2000 using Delphi programming language.

The database of reference (Fig 1) contains one million tuples. The data set used has four dimensions (Store: 10 attributes, Product: 14 attributes, Salesperson: 11 attributes and the Period: 1930 attributes).

Comparisons were made between MLL and the multidimensional array data cube (MDA). Results w.r.t. memory space and building time (Fig 4) show better performance for MLL that strongly depends on the number of stored points (Delphi pointer management uses up an important part of this space). Beyond 100,000 tuples, The MLL curve oversteps the array curve. This disadvantage is counterpoised by the space
and fixed-sized array that would require 2,972,200 cells ($10 \times 14 \times 11 \times 1930$), which corresponds to the size of the MDA. Whatever the number of tuples taken in the performance survey, the array size is limited by $2,972,200 \times (the\ size\ of\ a\ cell)$. This fixed-sized array may be sparse. Indeed, if we increase the size of a given dimension, the MDA size grows very quickly and oversteps the MLL size. It is also true that the growth of the number of dimensions increases the MDA size considerably, whereas the same variation does not have the same effect on MLL. Additional dimensions would indeed generate a lengthening of the list of dimensions. Moreover, for each additional dimension, a list of its attributes would be created. Note that the metadata on hierarchies as well as dimensions and their attributes do not exceed 1% of the space occupied by the MLL structures.

In addition, the building time of the data cube in MLL is distinctly lower than that of the MDA case (Fig 4). Response times for aggregation queries were also compared and the results are displayed showing the extent to which the MLL response time is better than the MDA’s (Fig 5).

6 CONCLUSION AND PERSPECTIVES

In this paper, we introduced the Multi-Link Lists data structure. This structure avoids the problem of sparse data often encountered in classical data structures in the MOLAP environment. Our performance results show that MLL performs much better than array structures w.r.t memory space and building time. The variation of the number of dimensions and their attributes, and the growth of hierarchies and their parameters has no effect on the MLL structure space memory.

We intend to study optimization techniques to both improve space memory and running time queries in MLL and be able to compare our solution to chunked arrays [ZDN97]. Moreover, we will give a more in-depth treatment for experimenting our MLL structure to the data cube computation methods.

7 REFERENCES

Related Content

**The Still Image Lossy Compression Standard - JPEG**
[www.irma-international.org/chapter/the-still-image-lossy-compression-standard---jpeg/112337](www.irma-international.org/chapter/the-still-image-lossy-compression-standard---jpeg/112337)

**An Efficient Random Valued Impulse Noise Suppression Technique Using Artificial Neural Network and Non-Local Mean Filter**

**Towards a Methodology for Semantic and Context-Aware Mobile Learning**
[www.irma-international.org/chapter/towards-a-methodology-for-semantic-and-context-aware-mobile-learning/113041](www.irma-international.org/chapter/towards-a-methodology-for-semantic-and-context-aware-mobile-learning/113041)

**Modified Distance Regularized Level Set Segmentation Based Analysis for Kidney Stone Detection**

**Analysis of Gait Flow Image and Gait Gaussian Image Using Extension Neural Network for Gait Recognition**