IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4777

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 since data are linked, and can easily extend the data cube by adding dimensions or hierarchies without re-building it.

When compared to array structures, MLL displays good performances both in terms of memory space and query execution time. Moreover, adding dimensions or hierarchies [HMV99] has no effect on the MLL memory space.

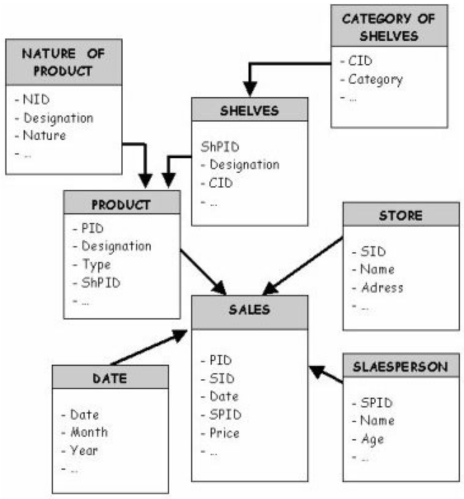
We propose three main algorithms: (1) loading raw data from different sources and structuring them into MLL, (2) handling the navigation into MLL, and (3) computing aggregations and storing them into MLL. For space limitation, we only present in this paper the Create MLL algorithm. The interested readers can find the whole algorithms in [BBD02].

Data cubes issues are introduced in Section 2. The MLL structure is covered in Section 3. The algorithm to create MLL is presented in Section 4. MLL and array structures are compared in terms of storage space and query execution time in Section 5. We conclude in Section 6 by anticipating on necessary extensions.

2 DATA CUBES ISSUES

To illustrate our approach, we use a botanic data warehouse that is modelled as a star schema [Kim96,Inm96] and concerns sales in French plant stores (Fig 1). We aim to study the sale activity through the measure "Price of products", according to the analysis axes *Product*, *Store*, *Period* and *Salesperson* dimensions. A data cube (Fig 2) is then defined by the following view.

Fig 1: Botanic star schema

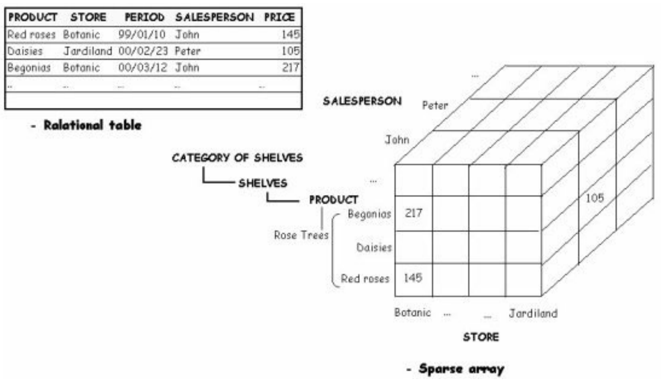


Create view dataset **as**
Store.SID, Product.PID, Period.Date, Salesperson.SPID, Price
From Store, Product, Period, Salesperson, Sales;

Formally, a data cube is composed by:

- A set of dimensions. Each dimension D=($Name_D$, A_D) is defined by its name $Name_D$ and a set of values A_D that are called attributes.
- A set of measures. Each measure M=($Name_M$, V_M) is defined by its name $Name_M$ and a set of possible values V_M (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=($Name_H$, P_H, n) is defined by its name $Name_H$, 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=($Name_{PH}$, PV) is defined by its name $Name_{PH}$ and is associated with a set of parameters values PV. PV is either a subset of $P_{H'}$ where H is a hierarchy of H'=($Name_{H'}$, $P_{H'}$, n-1) or a subset of A_D where H is a hierarchy of the dimension D=($Name_D$, A_D). Hierarchies of a dimension are noted D_i® H_{i1}® H_{i2}® H_{i3} ...

Fig 2: ROLAP and MOLAP representations of data cubes



PRODUCT	STORE	PERIOD	SALESPERSON	PRICE
Red roses	Botanic	99/01/10	John	145
Daisies	Jardiland	00/02/23	Peter	105
Begonias	Botanic	00/03/12	John	217
..	...	...	...	...

3 MULTI-LINK LISTS

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

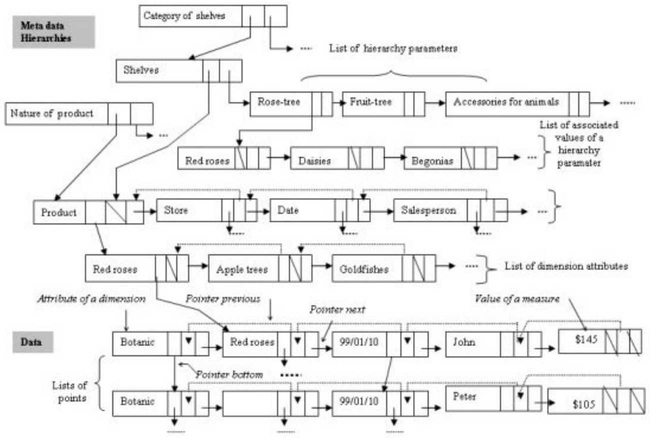
Data Lists

- *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.

Fig 3: MLL structure



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 (dcs: 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
                LAttDim.Create
                /* LinkBottom links a field of record or a list
                to another list by the bottom pointer)*/
                LDims.LinkBottom(LAttDim)
            End if
        Else
            LCoord.Insert(address of field)
            If field is not a measure Then
                /*To get the address of the list pointed by the
                bottom pointer*/
                LAttDim := LDims.GetBottom(field)
                /* Searches whether the coordinate is in the
                attributes list of the corresponding dimension*/
                Search (dimension, field, measure)
            End if
          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.

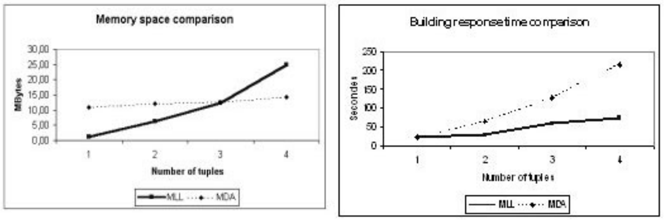
5 PERFORMANCE RESULTS

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

Figure 4: Comparison of building performances between MLL and MDA



and fixed-sized array that would require 2,972,200 cells (10*14*11*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 * (*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 (Fig5).

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

[AAD+96] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan and S. Sarawagi, *On the Computation of Multidimensional Aggregates*, In Proc.of the 22nd VLDB Conference, Bombay, Sept. 1996.

[BBD02] F. Bentayeb, O. Boussaid and J. Darmont, *Multi-Link Lists as Data Cube Structure in the MOLAP Environment,* Technical Report RR-0202 Eric Laboratory, University of Lyon2, pp 1-13, 2002.

[CD97] S. Chaudhuri and U. Dayal, *An overview of data Warehousing and OLAP Technology*, In ACM-SIGMOD, 1997.

[Cod93] E.F Codd, *Providing OLAP (on-line analytical processing) to user-analysts: an IT mandate*, Technical Report, E.F. Codd and Associates, 1993.

[GCB+97] J. Gray, S. Chaudhuri, BOSWORTH, A. Layman, D. Reichart and M. Venkatrao, *Data Cube: A relational Aggregation Operator Generalizing Group By, Cross-Tab, and Sub-Totals*, In Data Mining and Knowledge Discovery, Nb 1, 1997.

[HMV99] C. Hurtado, A.O. Mendelzon and A. A. Vaisman, *Maintaining Data cubes under Dimension Updates*, In Proc.of IEEE/ICDE, 1999.

[HRU96] V. Harinarayan, A. Rajaraman and J.D. Ullman, *Implementing Data Cubes Efficiently*, In Proc. ACM SIGMOD, 1996.

[Inm96] W.H. Inmon, *Building the Data Warehouse*, Jhon Wiley \& Sons, edition 2, 1996.

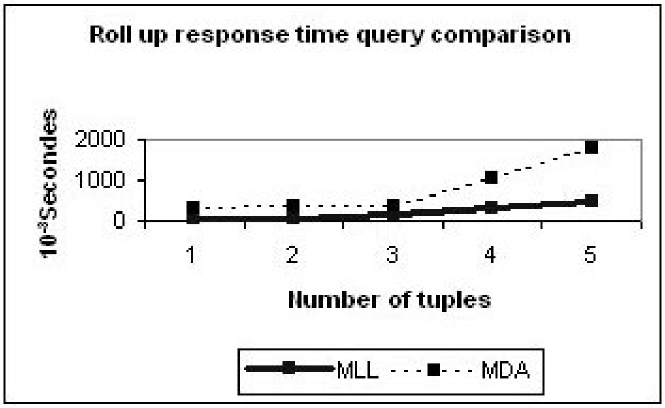
[JLS99] H.V. Jagadish, V.S. Lakshmanan and D. Srivastava, *What can Hierarchies do for Data Warehouses?* In Proc of the 25th VLDB Conference, Edinburgh, 1999.

[Kim96] R. Kimball, *The data warehouse toolkit*, Edition John Wiley, 1996.

[RS97] K. A. Ross and D. Srivastava, *Fast Computation of Sparse Datacubes*, In Proc.of the 23nd VLDB Conference, Athens, 1997.

[ZDN97] Y. Zhao, P.M. Deshpande and J.F. Naughton, *An Array-Based Algorithm for simultaneous Multidimensional Aggregates*, ACM-SIGMOD, 1997.

Figure 5: Comparison of query performances between MLL and MDA


Roll up response time query comparison
10^-3Secondes
Number of tuples
MLL
MDA