ISSN 22773061

Overview of OLAP cubes, importance, build Considerations and querying with MDX.

Anil D Devangavi[1], Dr Puneet Aggarwal[2]
[1]Dept of ISE, B.E.C, Bagalkot, India
anildevangavi_s@yahoo.co.in
[2] I.M.E.C., Ghaziabad, India
apuneet15@yahoo.com

ABSTRACT

An OLAP cube is a method of storing data in a multidimensional form, generally for reporting purposes. The purpose of an OLAP cube is to store data in such a way that an end user can slice and dice through the data and the storage structure itself (multidimensional) while allowing for very fast access to the data. Cubes are characterized by dimensions and measures. Dimensions refer to how the data is organized and are usually made up of character fields which tell us about the data. Measures, on the other hand, are those elements in the data which quantify something (a number) and can usually be additive . The business intelligence industry uses the word "cube" because it best describes the resulting data. The query language used to interact and perform tasks with OLAP cubes is multidimensional expressions (MDX).In this paper an attempt is made to give the overview, importance and build considerations of Olap cubes. Also querying of OLAP cube with MDX is discussed.

KEYWORDS: OLAP, OLAP cube, MDX,OLAP Universe.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 9, No 1
editor@cirworld.com
www.cirworld.com, member.cirworld.com

INTRODUCTION

In computing, online analytical processing, or OLAP is an approach to answering multi-dimensional analytical (MDA) queries swiftly.[1] OLAP is part of the broader category of business intelligence, which also encompasses relational database, report writing and data mining. Typical applications of OLAP include business reporting for sales, marketing, management reporting, business process management (BPM),budgeting and forecasting, financial reporting and similar areas, with new applications coming up, such as agriculture. The term *OLAP* was created as a slight modification of the traditional database term OLTP (Online Transaction Processing).

OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives. OLAP consists of three basic analytical operations: consolidation (roll-up), drill-down, and slicing and dicing.Consolidation involves the aggregation of data that can be accumulated and computed in one or more dimensions. For example, all sales offices are rolled up to the sales department or sales division to anticipate sales trends. By contrast, the drill-down is a technique that allows users to navigate through the details. For instance, users can view the sales by individual products that make up a region's sales. Slicing and dicing is a feature whereby users can take out (slicing) a specific set of data of the OLAP cube and view (dicing) the slices from different viewpoints. Databases configured for OLAP use a multidimensional data model, allowing for complex analytical and ad-hoc queries with a rapid execution time.They borrow aspects of navigational databases, hierarchical databases and relational databases. Chapters 1,2 and 3 present the overview of olap and olap cubes. In chapters 4 and 5 we have discussed the importance and build process considerations of OLAP cubes. In chapter 6 query processing language MDX is explained with examples.Chapter 7 gives the brief introduction of OLAP universe and explains the differences between OLAP universe and OLAP cube.

1. OLTP AND OLAP

Online transaction processing, or OLTP, refers to a class of systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing . OLTP has also been used to refer to processing in which the system responds immediately to user requests .The main emphasis for OLTP systems is put on very fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second. In OLTP database there is detailed and current data, and schema used to store transactional databases is the entity model (usually 3NF).In general we can say that OLTP provides source data to data warehouses and the OLAP is used to analyze it .So OLTP is also referred as Operative Environment and OLAP as Informative Environment.

SAP BW enables Online Analytical Processing (OLAP) for the staging of information from large amounts of operative and historical data. OLAP technology permits multi-dimensional analyses according to various business perspectives. At the core of any OLAP system is the concept of an OLAP cube (also called a 'multidimensional cube' or a *hypercube*). It consists of numeric facts called *measures* which are categorized by dimensions .The cube metadata is typically created from a star schema or snowflake schema of tables in a relational database.

2. OVERVIEW OF OLAP SYSTEMS

The core of any OLAP system is an OLAP cube (also called a 'multidimensional cube' or a hypercube). It consists of numeric facts called *measures* which are categorized by *dimensions*. The measures are placed at the intersections of the hypercube, which is spanned by the dimensions as a Vector space. The usual interface to manipulate an OLAP cube is a matrix interface like Pivot tables in a spreadsheet program, which performs projection operations along the dimensions, such as aggregation or averaging. The cube metadata is typically created from a star schema or snowflake schema of tables in a relational database. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables. Each *measure* can be thought of as having a set of *labels*, or meta-data associated with it. A *dimension* is what describes these *labels*; it provides information about the *measure*. A simple example would be a cube that contains a store's sales as a *measure*, and Date/Time as a *dimension*. Each Sale has a Date/Time *label* that describes more about that sale. Any number of *dimensions* can be added to the structure such as Store, Cashier, or Customer by adding a foreign key column to the fact table. This allows an analyst to view the *measures* along any combination of the *dimensions*. For example:

```
 Sales Fact Table
+-------------+----------+
| sale_amount | time_id  |
+-------------+----------+           Time Dimension
|     2008.10|     1234 |---+     +---------+-------------------+
+-------------+----------+   |     | time_id | timestamp         |
                             |     +---------+-------------------+
                             +---->|   1234  | 20080902 12:35:43 |
                                   +---------+-------------------+
```

Multidimensional databases and aggregations: Multidimensional structure is defined as “a variation of the relational model that uses multidimensional structures to organize data and express the relationships between data”. [2] The structure is broken into cubes and the cubes are able to store and access data within the confines of each cube. “Each cell within a multidimensional structure contains aggregated data related to elements along each of its dimensions”.[9] Even when data is manipulated it remains easy to access and continues to constitute a compact database format. The data still remains interrelated. Multidimensional structure is quite popular for analytical databases that use online analytical processing (OLAP) applications (O’Brien & Marakas, 2009). Analytical databases use these databases because of their ability to deliver answers to complex business queries swiftly. Data can be viewed from different angles, which gives a broader perspective of a problem unlike other models [3]. *Aggregations*: It has been claimed that for complex queries OLAP cubes can produce an answer in around 0.1% of the time required for the same query on OLTP relational data [2]. The most important mechanism in OLAP which allows it to achieve such performance is the use of *aggregations*. Aggregations are built from the fact table by changing the granularity on specific dimensions and aggregating up data along these dimensions. The number of possible aggregations is determined by every possible combination of dimension granularities. The combination of all possible aggregations and the base data contains the answers to every query which can be answered from the data.

3. OLAP CUBE

An OLAP cube is an array of data understood in terms of its 0 or more dimensions. *OLAP* is an acronym for online analytical processing.[1] OLAP is a computer-based technique for analyzing business data in the search for business intelligence.

Terminology

A cube can be considered a generalization of a three-dimensional spreadsheet. For example, a company might wish to summarize financial data by product, by time-period, by city to compare actual and budget expenses. Product, time, city and scenario (actual and budget) are the data's dimensions. *Cube* is a shortcut for *multidimensional dataset*, given that data can have an arbitrary number of *dimensions*. The term hypercube is sometimes used, especially for data with more than three dimensions. Each cell of the cube holds a number that represents some *measure* of the business, such as sales, profits, expenses, budget and forecast. OLAP data is typically stored in a star schema or snowflake schema in a relational data warehouse or in a special-purpose data management system. Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.

To better understand the MDX language and the OLAP technology it supports, a basic understanding of the OLAP cube components is required.*Dimensions* are the top or highest categories of a cube. They contain subcategories of data known as levels and measures. A dimension can have multiple hierarchies and can be used in multiple cubes. A cube can have up to 64 dimensions. *Hierarchies*: A dimension might be categorized into different *hierarchies*. For example, a company might categorize its profit dimension along the verticals of geography, sales territory, or market. The elements of a dimension can be organized as a hierarchy [5] ,a set of parent-child relationships, typically where a parent member summarizes its children. Parent elements can further be aggregated as the children of another parent. For example May 2005's parent is Second Quarter 2005 which is in turn the child of Year 2005. Similarly cities are the children of regions; products roll into product groups and individual expense items into types of expenditure. *Levels* are categories of organization within a dimension. Levels are hierarchical, and each level that is descended in a dimension is a component of the previous level. For example, a time dimension could include the following levels: Year, Quarter, Month, Week, and Day. Members and Measures :An additional component of a dimension and a level is a *member*. A member is a component of a level and is analogous to the value of a variable on an individual record in a data set. It is the smallest level of data in an OLAP cube. In addition to creating dimension members, a user can create calculated members and named sets that are based on underlying members or on other calculated members and named sets. These user-defined objects are based on evaluated query data from the cube.

Calculated members and named sets can be created in three different ways: 1) Query scope calculated member is only available during the query that defines it. It is created by using the WITH MEMBER/SET keyword. 2)Session scope calculated member is available for the user that defines the object for the duration of that session. It is created by using the CREATE SESSION MEMBER/SET keyword. 3)Global scope calculated member is available for anyone to use and is stored with the cube. It is created by using the CREATE GLOBAL MEMBER/SET keyword. Named sets have the same three scopes. Calculated members can be created in the Measures dimension and can include any combination of members. Calculated members can also be created in any other dimension and are known as *nonmeasure-based calculated members*. Examples of measures include sales counts, profit margins, and distribution costs.

Operations

Conceiving data as a cube with hierarchical dimensions leads to conceptually straightforward operations to facilitate analysis. Aligning the data content with a familiar visualization enhances analyst learning and productivity [7]. The user-initiated process of navigating by calling for page displays interactively, through the specification of slices via rotations and drill down/up is sometimes called "slice and dice". Common operations include slice and dice, drill down, roll up, and pivot.

OLAP slicing

Slice is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension.

OLAP dicing

Dice: The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions. The new cube will show the sales figures of a limited number of product categories, the time and region dimensions cover the same range as before.

OLAP Drill-up and drill-down

Drill Down/Up allows the user to navigate among levels of data ranging from the most summarized (up) to the most detailed (down).[5] Roll-up: A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as "profit = sales - expenses".

OLAP pivoting

Pivot allows an analyst to rotate the cube in space to see its various faces. For example, cities could be arranged vertically and products horizontally while viewing data for a particular quarter. Pivoting could replace products with time periods to see data across time for a single product . The whole cube is rotated, giving another perspective on the data.

Mathematical Definition

In database theory, an OLAP cube is [7] an abstract representation of a projection of an RDBMS relation. Given a relation of order *N*, consider a projection that subtends *X*, *Y*, and *Z* as the key and *W* as the residual attribute. Characterizing this as a function,

$f : (X,Y,Z) \rightarrow W,$

the attributes *X*, *Y*, and *Z* correspond to the axes of the cube, while the *W* value into which each (*X*, *Y*, *Z*) triple maps corresponds to the data element that populates each cell of the cube. Insofar as two-dimensional output devices cannot readily characterize four dimensions, it is more practical to project "slices" of the data cube (we say *project* in the classic vector analytic sense of dimensional reduction, not in the SQL sense, although the two are conceptually similar),

$g : (X,Y) \rightarrow W$

which may suppress a primary key, but still have some semantic significance, perhaps a slice of the triadic functional representation for a given *Z* value of interest.The motivation behind OLAP displays harks back to the *cross-tabbed report* paradigm of 1980s DBMS. The resulting spreadsheet-style display, where values of *X* populate row $1; values of *Y* populate column $A; and values of $g : (X, Y) \rightarrow W$ populate the individual cells "southeast of" $B2, so to speak, $B2 itself included.

OLAP server

The chief component of OLAP is the OLAP server, which sits between a client and a database management systems (DBMS). The OLAP server understands how data is organized in the database and has special functions for analyzing the data. There are OLAP servers available for nearly all the major database systems.

4. IMPORTANCE OF OLAP CUBE

An OLAP cube is a specially designed database that is optimized for reporting. While most databases designed for online transaction processing such as those used in claims processing are designed for efficiency in data storage, OLAP cubes are designed for efficiency in data retrieval. This means that the data is stored in such a way as to make it easy and efficient for reporting. Regular “relational” databases treat all data into the database similarly, however .OLAP cubes categorize data into "dimensions" and "measures". Measures represent items that are counted, summarized or aggregated, such as costs or units of service. Dimensions are variables by which measures are summarized, such as hospitals, physicians, or dates of service. This organization of data greatly facilitates the ability to formulate data requests based on real-life situations.

In addition, many of queries that could be posed to the data are "pre-aggregated" in the database such that the answers have already been precalculated and can be reported without delay. The term "cube" comes from the geometric object that has three dimensions. OLAP cubes can have many more dimensions than 3, but the term continues to apply.

5. OLAP BUILD PROCESS CONSIDERATIONS

There are different options for building an effective OLAP cube and different considerations that must be taken into account . Working with a high volume of data forces a developer to think outside the norm to support reporting requirements but also design a cube that actually builds in time and still functions [8].

Overall Build Time

If the cube is completely rebuilt or incrementally updated, the availability of the OLAP cube to users for reporting is a basic requirement to consider. Larger data volumes result in larger cubes, which usually require a longer period of downtime to build as data grows.

Method of Cube Update

Completely rebuilding a cube ensures all historical data is summarized when the cube is built. A complete rebuild takes longer than an incremental update With large data volumes it can take considerably longer .An incremental update process leaves the OLAP cube available for reporting while new data is added. Incremental updates are efficient but historical data may n eed to be updated or removed .Always assess the possibility of an incremental update depending on reporting requirements and the necessity to update historical data in the OLAP cube.

Frequency of Cube Updates

Going hand -in-hand with build time and method of updating, how often the cube update process occurs needs to be considered. The volatility of source data and the business necessity of having aggregate level data presented should be a major consideration.

To Build or Not to Build the NWAY

The NWAY aggregation stores all measures aggregated to the crossing of every level defined in a cube. If the cube is taking too long to build, the volume and grain of the NWAY crossing may not be feasible to build. Try disabling the NWAY aggregation by adding the NO_NWAY option to the PROC OLAP statement. Not having the NWAY requires additional aggregations so the cube can perform nominally. If no aggregations exist, the OLAP server will aggregate source data at run time, which can impact performance, especially when working with big data.

Divide and Conquer

Build the cube using a “divide and conquer” technique described in SAS Global Forum paper 431-2012. This process loads data into an OLAP cube in segments to avoid physical constraints and allows access while the cube is being built.

Programming Tip

During the development of an OLAP cube, limit the amount of data processed by the cube so development cycles do not become extremely long because of high volumes of data. Use table views or subset the data into temporary tables.

6. MDX OVERVIEW

Multidimensional Expressions (MDX) is a powerful syntax that enables you to query multidimensional objects and provide commands that retrieve and manipulate multidimensional data from those objects. MDX is designed to ease the process of accessing data from multiple dimensions. It addresses the conceptual differences between two-dimensional and multidimensional querying. MDX provides functionality for creating and querying multidimensional structures called *cubes* with a full and complete language of its own. The Multidimensional Expressions (MDX) language allows users to describe queries and manipulate multidimensional information, such as the data stored in cubes. MDX functions can define calculated members and query cube data. MDX is similar to the Structured Query Language (SQL), and MDX provides *Data Definition Language* (DDL) syntax for managing data structures. However, its features can be more complex and robust than SQL’s features. The SAS 9.1 OLAP Server technology uses MDX to create OLAP cubes and data queries. MDX is part of the underlying foundation for the SAS 9.1 OLAP Server architecture, and it offers detailed and efficient searches of multidimensional data. With MDX, specific portions of data from a cube can be extracted and then further manipulated for analysis. This allows for a thorough and flexible examination of SAS OLAP cube data. Users of MDX can take advantage of such features as calculated measures, numeric operations, and axis and slicer dimensions.

Comparing an MDX Query to an SQL Query

The MDX and SQL languages have a similar structure, and include some of the same keywords. However, one of the significant conceptual differences between the two languages is that MDX builds a multidimensional view of the data, where SQL builds a relational view. Although it is possible to use SQL exclusively to query cubes, the MDX query was designed specifically to retrieve multidimensional data structures with almost any number of dimensions. Additionally, SQL was designed to handle only two-dimensional tabular data when processing queries, where MDX can process one, two, three, or more dimensions in queries. Each dimension in MDX is referred to as an axis, and the terms column and row are simply used as aliases for the first two axis dimensions in an MDX query (the alias itself holds no real meaning to MDX).Additional to the conceptual differences between MDX and SQL, both languages use different terms to describe their basic concepts. For example, the cube concept explained above is actually a table in the SQL language; and the multidimensional term level is a column in SQL (referring to a string or discrete number). Additionally, a dimension in MDX refers to what would be understood in relational terms as several related columns or a dimension table; and a measure in MDX refers to a discrete, continuous or numeric column in SQL. Finally, a member in MDX refers to the specific row and column of a dimension table.

Additional MDX Concepts and Expressions –Tuples and Sets

MDX extracts multidimensional views of data. A *tuple* is a slice of data from a cube.It is a selection of members (or cells) across dimensions in a cube. It can also be viewed as a cross-section or vector of member data in a cube. A tuple can be composed of member(s) from one or more dimensions. However, a tuple cannot be composed of more than one member from the same dimension. *Sets* are collections of tuples. The order of tuples in a set is important when querying cube data and is known as *dimensionality*. It is important to note that the order of the dimension members in every tuple must be the same. For example, if your first tuple is (time_dimension_member, geography_dimension_member), then every other tuple in that set must also have two members in it, the first from the time dimension and the second from the geography dimension[6].

Basic MDX Queries and Syntax

Basic MDX queries use the SELECT statement to identify a data set that contains a subset of multidimensional data. The basic MDX SELECT statement is composed of the following clauses: i) WITH clause (optional). This allows calculated members or named sets to be computed during the processing of the SELECT and WHERE clauses. ii) SELECT clause. The SELECT clause defines the axes for the MDX query structure by identifying the dimension members to include on each axis. The number of axis dimensions of an MDX SELECT statement is also determined by the SELECT clause. The members from each dimension (to include on each axis of the MDX query) must be identified. iii) FROM clause. The cube that is being queried is named in the FROM clause. It determines which multidimensional data source will be used when extracting data to populate the result set of the MDX SELECT statement. The FROM clause (in an MDX query) can list only a single cube. Queries are restricted to a single data source or cube. iv)WHERE clause (optional). The WHERE clause further restricts the result data. The axis that is formed by the WHERE clause is often referred to as the *slicer*. The WHERE clause determines which dimension or member is used as a slicer dimension. This restricts the extracting of data to a specific dimension or member. Any dimension that does not appear on an axis in the SELECT clause can be named on the slicer.

MDX queries, and specifically the SELECT statement, can have up to 128 axis dimensions. The first five axes have aliases. Furthermore, an axis can be referred to by its ordinal position within an MDX query or by its alias. In total you can have a maximum of 64 different axes. The SELECT clause of the statement supports using MDX functions to construct different members in a set on axes. The WITH clause of the statement supports using MDX functions to construct calculated members to be used in an axis or slicer. The following example shows the syntax for the SELECT statement:

```
[WITH
[MEMBER <member-name> AS ’<value-expression>’ |
SET <set-name> AS ’<set-expression>’] . . .]
SELECT [<axis_specification>
[, <axis_specification>...]]
FROM [<cube_specification>]
[WHERE [<slicer_specification>]].
```

Simple Examples

The data that is used in these simple examples is from a company that sells various makes and models of cars.

1) The company needs to report sales figures for different months. Here is a simple two-dimensional query:

```
select
{ [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] } on columns,
{ [DATE].[All DATE].[March], [DATE].[All DATE].[April] } on rows
from mddbcars.
```

2) Using this example code, you can flip the rows and columns:

```
select
{ [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] } on rows,
{ [DATE].[All DATE].[March], [DATE].[All DATE].[April] } on columns
from mddbcars.
```

3) Select a different measure (SALES_N) to be the default:

```
"select
{ [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] } on columns,
{ [DATE].[All DATE].[March], [DATE].[All DATE].[April] } on rows
from mddbcars
where ([Measures].[SALES_N])
```

4) Demonstrate ":" to get a range of members:

```
select
```

ISSN 22773061

```
{ [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] } on columns,
{ [DATE].[All DATE].[January] : [DATE].[All DATE].[April] } on rows
from mddbcars.
```

5) Demonstrate the .MEMBERS function:

```
select
{ [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] } on columns,
{ [DATE].members } on rows
from mddbcars.
```

6) Demonstrate the .CHILDREN function:

```
select
{ [CARS].[All CARS].[Ford].children } on columns,
{ [DATE].members } on rows
from mddbcars.
```

7)Select more than one dimension in a tuple:

```
select
{ ( [CARS].[All CARS].[Chevy], [Measures].[SALES_SUM] ),
( [CARS].[All CARS].[Chevy], [Measures].[SALES_N] ),
( [CARS].[All CARS].[Ford], [Measures].[SALES_SUM] ),
( [CARS].[All CARS].[Ford], [Measures].[SALES_N] ) } on columns,
{ [DATE].members } on rows
from mddbcars.
```

8) The crossjoin function makes tuple combinations for you:

```
select
{ crossjoin ( { [CARS].[All CARS].[Chevy], [CARS].[All CARS].[Ford] },
{ [Measures].[SALES_SUM], [Measures].[SALES_N] } )} on columns,
{ [DATE].members } on rows
from mddbcars.
```

Query-Calculated Member Examples

The data that is used in these examples is from a company that sells various makes and models of cars. The company needs to report on sales figures for different months. Example

This query creates a calculation for the average price of a car. The average price of a car is calculated by dividing the sales_sum by the count (sales_n). The query returns the sales_sum, sales_n, and the average price for March and April.

```
with
member [Measures].[Avg Price] as '[Measures].[SALES_SUM] / Measures].[SALES_N]'
select
{ [Measures].[SALES_SUM] , [Measures].[SALES_N], [Measures].[Avg Price] }on columns,
{ [DATE].[All DATE].[March], [DATE].[All DATE].[April] } on rows
from mddbcars.
```

Here is the resulting output:

```
Date Sales_sum Sales_n Avg Price
March $59,000.00 4 14750
April $34,000.00 3 11333.33.
```

7. OLAP UNIVERSE

A OLAP universe is a BusinessObjects universe that has been generated from a OLAP cube or query. The universe is created automatically from a selected connection to a OLAP data source using an OLAP query flattening driver that is installed as an add in to Designer XIR2.Once the universe has been created it can be exported to the Central Management System (CMS) as any other universe, and is then available to Web Intelligence users to run queries and create reports. You create a OLAP universe by selecting a OLAP connection to a QueryCube or InfoCube. The universe creation process is automatic once you have selected the connection. OLAP structures are mapped directly to classes, measures, dimensions, and details. The universe structure appears in the Universe pane. There is no table schema in the Structure pane. Once you have created the OLAP universe, you can modify any of the universe components as for any other universe. You can use Designer to create OLAP universes from the following OLAP data cube sources: i) BW SAP ii) Microsoft Analysis Services and iii) Essbase. Designer creates universe automatically from a connection to a InfoCube or QueryCube.The cube elements are mapped directly to equivalent structures in the universe. A OLAP flattening driver is used to build a relational view from the cube. The universe is then generated from the view.

An Olap cube is a prebuilt cube of data that has been aggregated according to pre-determined dimensions whereas a universe is a semantic layer over a set of joined tables that generates sql and aggregation on the fly depending what the user selects. The measures can use sum / average etc and then the dimension objects will be aggregated at run time - taking longer but more flexible. Simpy put an OLAP cube contains data; a universe doesn't.
OLAP cube: Data
Universe: Meta Data

And a universe can be built on top of an OLAP cube.

CONCLUSION

We have made an attempt to give the overview, importance and build considerations of Olap cubes. Also querying of OLAP cube with MDX is discussed. Several relevant examples have been illustrated. Comparison of OLAP cube and universe has been done.

REFERENCES

[1] Cabibbo, L. and Torlone, R.: A logical approach to multidimensional databases, *6th International Conference on Extending Database Technology (EDBT98, Valencia, Spain),* G. Alonso (eds.), LNCS, Vol. 1377, Springer, 1998

[2] S. Chaudhuri and U. Dayal. *An Overview of Data Warehousing and OLAP Technologies*. ACM SIGMOD Record 26(1), Marc 1997.

[3] M.C. Wu and A.P. Buchmann. *Research Issues in Data Warehousing*. BTW'97. 1997.

[4]Ammoura, A., Zaiane, O.R., & Ji, Y. (2001). Towards a Novel OLAP Interface to Distributed Data Warehouses.

Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery, LNCS Vol. 2114, 174-185.

[5]Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. *ACM SIGMOD Record*, 26(1), 65-74.

[6] Choong, Y.W., Laurent, D., & Marcel, P. (2003).Computing Appropriate Representations for Multidimensional

Data. *Data & Knowledge Engineering*,

45(2), 181-203.

[7] Codd, E.F., Codd, S.B., & Salley, C.T. (1993). Providing OLAP to User-Analysts: An IT Mandate. *E.F. Codd and Associates Technical Report*.

[8] Cuzzocrea, A. Saccà, D., & Serafino, P. (2006). A Hierarchy-Driven Compression Technique for Advanced OLAP Visualization of Multidimensional Data Cubes.*Proceedings of the 8th International Conference on Data Warehousing and Knowledge Discovery*, LNCS Vol. 4081, 106-119.