

Multi-dimensional Data Model Extensions to Data Warehouses

(This is a draft document, describing software under development. Changes may occur to software offerings described before general availability. No guarantee of features appearing as described is offered. This is our best good-faith effort to describe the work underway.)

OLAP as an enabling technology will mean different things to different audiences. Beginning with an IT audience, OLAP offers yet another way to deliver on the mission of providing end-user analysts access to relevant business information, in a timely fashion. It has the additional attraction of offering that access organized in ways that business end-users understand...by the dimensions of the business, rather than by tables of information. Relevant to IT reviewers will be the relative costs of storage, (i.e. will this take more disk space for new ways of storing data..and how much), network traffic (i.e. will user queries

require large amounts of data to be passed across the network, or will some work be done on server), its integration with current or planned data warehouse strategies and architectures, performance (i.e. time to build multidimensional models) and finally maintenance (i.e. update facilities, backup). In addition, an element of performance could be guaranteeing integrity during on-line updating.

What do OLAP buyers want?

■ Information Technology

- ◆ Enable end-users fast access to summarized data
- ◆ Must be manageable in data warehouse architecture
- ◆ Will be interested in build, maintain and performance capabilities
- ◆ Interested in hardware and network demands

■ Business End-Users

- ◆ Satisfy need to have data reflect their business vision
- ◆ Exploratory 'analysis', help to identify possible answers
- ◆ Need contact back to subsets of detail data
- ◆ Simple (i.e. spreadsheet-like) access

End-users see OLAP as the access to data about their business, organized along the categories that make most sense to them. It is data which is summarized at many levels of detail and, probably most important, has history, allowing for the examination of trends, both backwards and forwards. Most products in this space offer access to these data via spreadsheets or table-like viewers, reducing the need to learn something new to begin gaining value from the summarized data.

Fast Access to Shared, Multidimensional Information (FASMI)

- Target is to deliver most responses within 5 seconds, with simplest 1 second, few > 20 seconds
- Easy for user to conduct any business analysis
- Secure, and if update necessary, locking
- Multidimensional conceptual view, including hierarchies
- All data needed: amount of input data handled, and storage requirement

Shared = the system implements all the security requirements for confidentiality (possibly down to cell level) and, if multiple write access is needed, concurrent update locking at an appropriate level.

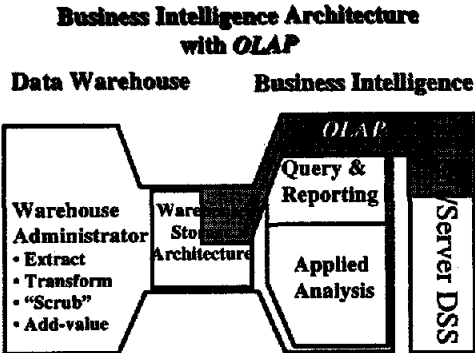
Multidimensional = provides a multidimensional conceptual view of the data, including full support for hierarchies and multiple hierarchies as this is certainly the most logical way to analyze business organizations. (This does not) specify what underlying database technology should be used providing that the user get a truly multidimensional conceptual view.

Information = all of the data and derived information needed, wherever it is and however much is relevant for the application. The measure is the capacity of the product in terms of how much input data it can handle, not how many gigabytes it takes to store it.

Fast Access to Shared, Multidimensional Information (Nigel Pendse & Richard Creeth)

Fast = system targeted to deliver most responses to users within 5 seconds, with the simplest taking around 1 second, and very few taking more than 20

Access = the system can cope with any business logic and statistical analysis that is relevant for the application and the user, and keep it easy enough for the target user.



OLAP is a technology which runs through a number of the "pieces" of a Data Warehouse/Business Intelligence environment. It has a place in the modeling of a data warehouse, both for the physical organization of the data model, and the metadata registration.

It also plays a key part of a reporting environment. The fundamental task which is being accomplished is to provide

multidimensional access to large amounts of data. This is done by summarizing the data along business defined dimensions, then making them accessible via common "viewer" style interfaces.

Finally, in the class of client/server packaged Decision Support products (i.e. SAS Business Solution for Financial Consolidation & Reporting), OLAP engines are key to providing multidimensional access to popular ERP based data systems.

"Slice and dice" has become the identifying metaphor for OLAP "analysis". It describes the activity of drilling through the available data, in an unanticipated manner, allowing users to make decisions on where to go next, at any time in their examination of the data. A typical "slice and dice" task might look like this:

"Let's look at sales, by region, for this time period, compared to same time last year. Now, let's break this out by product group. OK, let's just look at Product Group A...but now, let's look at the last six months. Finally, let's look at Product Group A sales, over the last six months by sales team, within each region." Multidimensional navigation is a key component enabled by the MDDB, and the viewers in SAS/EIS.

"Reach through" means to retrieve the detail data which makes up the summary being viewed. So, a user sees the above query, and would like to have all of the individual sales records which make up those summaries. The ability to easily request this summary back from the base table is reach through. This is enabled on SAS/EIS viewers as a pop menu item.

Client/server configurations suggest that MDDB servers can reside on a number of different platforms, and can be queried by a variety of possible client platforms. With our multi-platform support, MDDB's will be created and stored on ALL supported platforms. PC and UNIX clients will be able to send requests to remote MDDB servers, as well as process local MDDB copies.

SAS for OLAP addresses...(cont.)

- Application development tools within SAS re-architected to make use of both models of data (relational tables and multidimensional models)
- Enablement for:
 - ◆ Business End-users
 - ◆ Business Analysts
 - ◆ Business System Architects
- Reporting facilities across all levels of applications share a common metaphor of 'drag and drop'.

objects will simply create temporary MDDB structures, improving performance for the summarizations needed for EIS objects. An application assembler, then, can choose to point BOL objects at MDDB models, or tables. In either case, performance will be enhanced, while adding flexibility.

OLAP is an enabling technology for all levels of business analyst. The differing requirements are addressed via the interface made available. For the Business End-User (defined as someone whose primary jobs do not include programming, but have need to use computer technology in a true "end-user friendly" manner) the Report Gallery facility is

SAS for OLAP addresses...

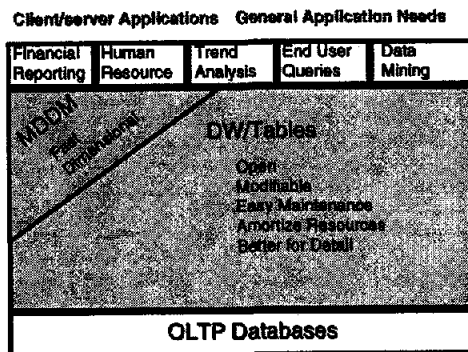
- The fundamental architectural task to be accomplished to enable "*slice and dice*" examination of data is summarization
- One of the key features available in this architecture is *reach through* or the ability to reconnect to the underlying detail which makes up the dimensional summaries
- These models will operate in a variety of *client/server configurations*, making use of all supported platforms

The structure of the MDDB, with its subtables and central fact table, require an intelligent "router" to identify which structure most efficiently satisfies the client request. This "router" is contacted by client applications via method calls. These method calls are built into the new SAS/EIS Business Object Library for release 6.12. This means that the objects in the BOL will be enabled to take advantage of the MDDB. They (the objects) will also, however, continue to be enabled act on tabular data. The

being made available. This will allow reports to be customized from templates, and stored in personal “folder EIS” applications.

For the more analytical user, viewers for doing OLAP applications are available in SAS/EIS. In addition, the ability to build new MDDB structures, reach through to detail data, and delivery of these detail extracts to any of the SAS System reporting or analysis capabilities is available.

OLAP: Tables & Multidimensional models



A comprehensive data warehouse/OLAP solution offers Business Analysts access to information in a variety of data models. The appropriateness of the models is dictated by the demands of the analysis and applications. For example, summarized, multidimensional models are very good for applications like financial reporting, because much of the analysis and reporting is organized around accounting conventions. New categorizations are typically just rearrangements of already defined dimensions. Summaries at a “fact table” level can be rearranged to satisfy most

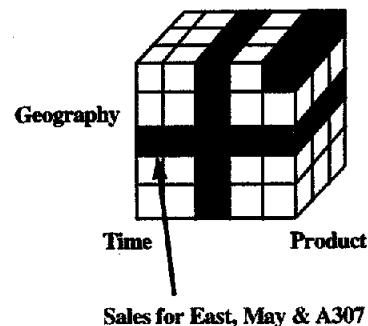
new reporting, without returning to the atomic level, transaction data.

Other applications, however, make use of variation calculated from the underlying detail data. The most notable example of this is the customer profiling information, known often as “data mining”. This analysis presumes no a priori organization of data, and searches for new “categories”.

The basic enhancement a focus on multidimensionality offers to client applications is access to data, organized in a fashion that aligns with the way business end-users understand their enterprise. This is accomplished by defining “dimensions” of the business, and then providing summaries of data within the definitions of those “dimensions”. This enables users to ask the more typical query which involves a number of factors (e.g. geography, time, product, budget vs. actual, etc.) without being familiar with the underlying organization of the data.

Table vs. Multidimensional data model

June	12Sept	AW	34	1837
South	12Sept	AME	27	2104
June	13Sept	EE	10	378
Florida	13Sept	EE	28	178
Wilson	12Sept	A307	39	1837
South	12Sept	BP	39	222

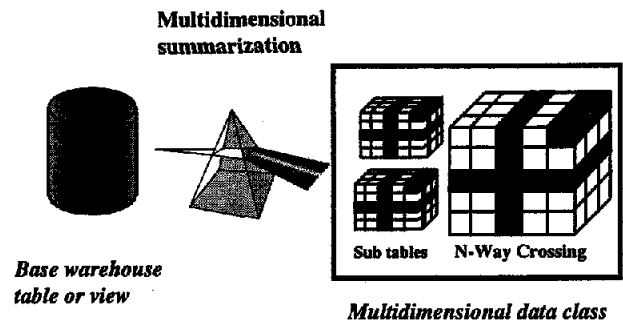


Tables, or relational tables, in data warehouses, typically are organized in a subject oriented fashion, which should cut down on the number times an end-user has to “join” tables together to answer a question. The use of SQL to manage retrieving data from tables is NOT an end-user task, and can actually be cumbersome, even for professional developers.

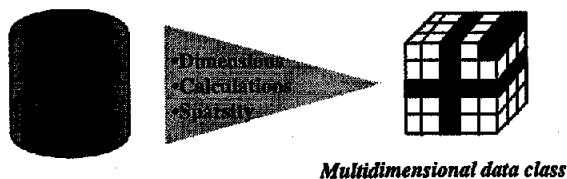
Integration of these, and other possible models, will become a key part of data warehouse design.

The SAS MDDB object follows a two step methodology for providing multidimensional (OLAP-style) access to end user applications. The first step is the creation of the MDDB N-Way crossing. This represents a "fact table" of the full list of crossings specified in the creation phase of the MDDB. Only crossings with valid values are stored, thus addressing the "sparsity" problem in the first phase. This step has shown significant reduction in size of data as compared to the target base table. Some of this reduction is due to subsetting the number of columns retained. Once this "fact table" is created, application programmers have two options. In one case, MDDB tables consolidated into defined hierarchies are created and stored. These hierarchical consolidations can be stored in the same location as the central "fact table", and are accessible to requesting applications. The performance implications for creating these specified consolidations ahead of time is improvement in access time when requested by the client application.

Mddb In Data warehouse architecture



Size Relationships: Base table vs. MDDB



*Base warehouse
table or view*

An MDDB will generally be much smaller than the base table or tables it is created from. The reasons for this include:

- atomic level detail not stored in MDDB
- storing calculations is optional
- summary records compressed where possible
- dimensional labels stored separate from tables, saving space on each record

The actual size reduction will vary depending on a variety of factors. So, a structure with many discrete dimensions will result in a larger MDDB than one with fewer. The number of analysis variables kept will effect size (e.g. more analysis variables, larger size). Whether calculations are stored also can have a significant impact on size. Calculations are often done more efficiently on the subset of interest to the client...and can be quickly accomplished on client machines. Storing pre-calculations can lead to much larger MDDB structures.

First a few terms:

- input cells - this concept (Pendse & Creeth) measures the actual level of input to the MDDB model. So, if the original file contained individual transactions for each customer, the actual "input" table might first be summarized up to a daily total for transactions by customer..thus achieving a significant reduction. In addition, the number of actual analysis variables is counted as a multiplier. So if the "input" table has 5 million rows, and 3 analysis variables (i.e. sale amount, invoice and quantity sold), the "input cells" would number 15 million.
- sparsity - when crossing values across dimensions, there will be occasions when no data are available. An example would be that, some products were not sold in some regions, on some days. The extent to which these missing cells appear is known as the sparsity of the data.
- potential cells - taking the input cells and sparsity together, the potential cells are all possible input cells, if there was no sparsity.
- Classification variables - the "categories" which make up a dimension. So, a geographic dimension may include Region, State, County and Sales territory. This would count as four classification variables.

Sample reductions via summarizations into MDDB PC Based

Base table:

- Records = 35.5 million
- Storage = 1.5 G
- Sparsity = 2.2%
- Dimensions = 5
- Classifications = 10
- Analysis = 4

MDDB class

- Input cells = 4.8 million
- Potential cells = 19.2 mil
- Storage = 377 Mb
65 Mb (min)

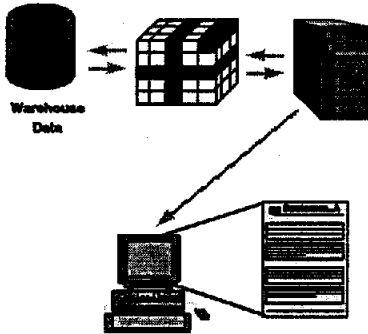
The SAS MDDB does not store missing, or sparse cells in the central fact table or in the subtables. Thus the actual input cells will be smaller than the number of potential cells, proportional

The difference between the Storage statistic and the minimum number reflects the storage of subtables. The minimum amount reflects the space required for the central fact table, without any subtable crossings. While this is a possible configuration, it would require that all client request retrieve their summarizations from this large table. Many subtable summarizations can be made in addition to the central fact table, to improve the performance of client applications.
to the sparsity of the data.

In a recent test situation, the above listed reductions were achieved when building and MDDB. This particular model was built on a PC platform, for use in a PC networked environment, or even in a "mobile" situation.

Using the definitions provided, it is clear that this MDDB model is able to achieve significant reduction in size from the base table. In addition, by only storing the real cells, where data actually exist, the MDDB model efficiently avoids the sparsity problem of "hyper cube" models.

**Client/server configurations:
Remote MDDB**



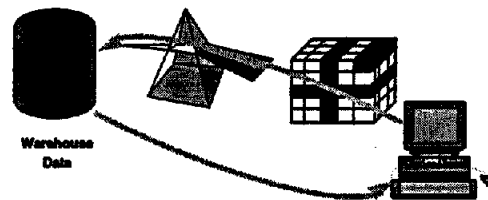
A remote MDDB configuration allows the multidimensional model to be stored and managed in a central server environment (with access to the underlying warehouse data). Access to this process from the client application is enabled via Remote Library Services (SAS/CONNECT or SAS/SHARE).

RLS has been tuned to understand that the nature of this access to MDDB's is different than to data tables which might require locking for update on a record level basis. Thus, larger

“bundles” of information can be sent down the wire, improving the performance in a distributed environment.

End-user “reach through” to warehouse data

“Reach through” is a critical component of an integrated Data Warehouse/OLAP architecture. This facility allows Business End-Users, viewing data in a multidimensional viewer application, to request an extract of the underlying detail data which made up the chosen summary. By drilling back to these base tables, the end-user can make selected subsets of data available to other reporting and analytical applications, some of which would require access to detail level data (e.g. customer profiling).



In an integrated Data Warehouse/OLAP configuration, the base warehouse information which made up the MDDB summaries is known. This means that it has also undergone the transformation and scrubbing logic inherent in data warehouse applications, to assure the consistency of the data. Pointing OLAP MDDB structures back to OLTP data would risk the possibility of creating results not checked for consistency, as well as potentially providing different data. The “different data” problem occurs when the OLTP data are updated, without those updates being reflected (yet) in the data warehouse.

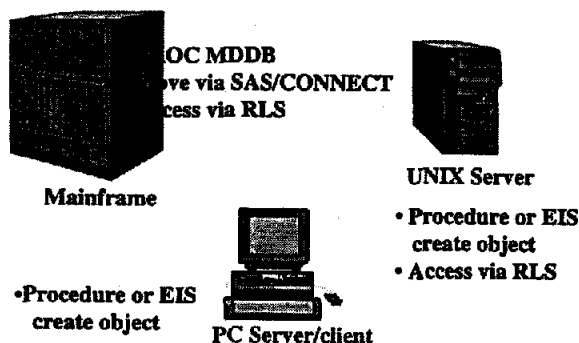
The steps to accomplish “reach through” include:

- identification of a summary of interest in the MDDB. This could mean all that is being viewed, or highlighted rows, columns or cells. Access to the underlying data is made available through a pop-menu selection “Show Detail Data”.
- SQL request passed back to underlying warehouse tables. Because the MDDB model can keep track of locations, a valid SQL statement can be generated to locate detail records which meet the criteria. This SQL request is sent back to the base table. The knowledge of this base table’s location is kept in the metabase.
- launch SAS reporting and analytical applications. Currently, a frame is being developed to display choices of a number of SAS applications, which would act on the extracted data from the underlying detail data.

The ability to create MDDB models will be included on all supported platforms. Most significantly, this includes creation of MDDB's on mainframe platforms. Because many, if not all, data warehouse projects begin with data which comes from a mainframe, the ability to create these MDDB models on the that platform will greatly reduce the amount of data

which must be sent to "server" platforms. In addition, the mainframe continues to be the superior platform for large, I/O intensive applications. So, a procedural interface will be shipped with BASE SAS (6.09E release) which will allow MDDB models to be built on host platforms.

MDdb's...where can they be built?



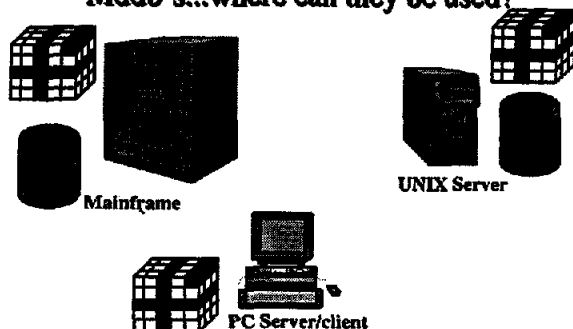
On the UNIX and PC platforms, the BASE SAS procedure will also be available, as well as a point and click interface in SAS/EIS, for building

MDDB models.

Because these are SAS data structures, they are also known to SAS/CONNECT, for purposes of moving the structures to other "server" platforms, or for access from client processes via

Remote Library Services (RLS). Customers will be able to choose the platform for building their MDDB's, and have facilities for moving them to other target platforms.

MDdb's...where can they be used?



With the ability to locate MDDB models on any supported platform, a variety of client/server configurations is possible. IT staff will be looking at:

- available disk space
- availability of CPU for builds and updates
- network bandwidth and projected amounts of data to be transferred
- network and server availability (e.g. mobile users)
- application access patterns