

Enterprise Data Warehouses and Essbase Data Marts

by

William Sterling, IBM, Inc.

Richard Sawa, Hyperion Solutions, Inc.

The following content was submitted to Wiley Press and was published in edited form as chapter 11, "Multidimensional Data with DB2 OLAP Server" in IBM Data Warehousing With IBM Business Intelligence Tools by Wiley Press, Michael L. Gonzales, Editor, 2003, ISBN 0-471-13305-1.

It is therefore not identical to the published chapter and may be said to be adapted from the published material or alternatively that the published chapter was adapted from the following material. In any case they are similar but not identical documents.

The term "DB2 OLAP Server" used in this document refers to Hyperion Essbase to which it was identical, and was sold by IBM at that time pursuant to an OEM agreement between IBM and Hyperion which as since been terminated. Therefore everything said about DB2 OLAP Server is also accurate about the Hyperion Essbase product of the time.

What you will learn

In this chapter we will convey the notion that DB2 OLAP Server is a multidimensional analytic database server platform that enables Business Analysts to analyze, manage and steer their respective Businesses. We define OLAP technologies as integral to BI/DW initiatives and essentially complementary to Data Mining and Relational technologies.

We will suggest a high-level architecture for implementing OLAP across the organizational enterprise that is really a variation of Ralph Kimball's bus architecture. We propose, like Kimball, that such implementations demand the coordination *and* cooperation of IT with Business personnel. The chapter also presents a brief discussion of basic DB2 OLAP Server functionality and storage implications and concludes with a list of additional reading and resource materials.

Introduction

Arbor Software introduced their **Extended Spreadsheet Services dataBASE** (Essbase) in the early 1990s. The name contains the innovation. Arbor engineers combined user-friendly spreadsheet client functionality with client-server modality.

Multi-dimensional databases existed before. Arbor Software's chief contribution was to provide a version for the client-server environment. They made Essbase fast and interactive and available for users within the current Windows[®] desktop environments. Arbor very quickly achieved market dominance.

IBM entered the OLAP marketplace in 1997 by licensing and reselling Essbase from Arbor Software.

In February 1998 IBM delivered DB2 OLAP Server V1.0 based on Arbor Essbase V5.0 on Windows NT, OS/2, AIX platforms. IBM called the product **DB2** OLAP Server because multidimensional data was stored in the DB2 relational database, while the Essbase product stores data in multidimensional arrays only. While arrays provided optimized performance, this also was not considered an open storage format. IBM provided the option to store data in relational Db2 tables as a means of opening up the data storage platform. IBM's [port of Essbase to DB2 OLAP Server subsequently evolved to support both storage formats.

Today the Hyperion and IBM products remain virtually identical except that IBM has added an OLAP Mining utility for DB2 OLAP Server.

What is OLAP?

OLAP stands for "Online Analytic Processing." Coined by Dr. E.F. Codd in the early years of Arbor Software, the term makes an obvious allusion to OLTP (online transaction processing.) It is an online as opposed to a batch process, which suggests having a dynamic rather than a static functionality. The objective of OLAP is to support the processing of business analytics and not business transactions. The acronym really

suggests that the *speed of processing* that had been brought to relational database transaction processing systems was now available to business users for analyzing data precisely in the form of Arbor Essbase.

The success of DB2 OLAP Server can also be related to extending the speed analogy to apply to OLAP *application development* as well. No small part of the initial success of OLAP tools in general is due to the fact that rapid deployment of *departmental* OLAP solutions provided an extremely fast return on investment (ROI) for organizations - Departmental OLAP application deployment times generally range between 30-90 days. Unfortunately *departmental solutions remain disjointed and, to that extent, can fail across the enterprise*. The point bears emphasis. It is often completely missed by OLAP-centric developers.

Just as unfortunate is that the observation of this disjointedness has led many IT professionals to the *incorrect* conclusion that OLAP tools are appropriate only for departmental initiatives and have marginal place within corporate IT initiatives. Providing OLAP across the enterprise brings to the Organization as a whole what OLAP traditionally provided to departments: *a controlled central data source for analytics*. The ability to provide a “single source of truth” to the organization in the form of advanced analytics that DB2 OLAP Server delivers should be, at the very least, a provocative notion for IT.

OLAP

Asking the question “What is OLAP” really implies asking a prior question, “What are analytics?” The reason why we start with a focus on the analytics question is that we find people in the marketplace often believe they know what OLAP is without having a good understanding of what is meant by business analytics.

As stated, OLAP means “Online Analytic Processing”. Business analytics as enabled by DB2 OLAP Server are profoundly successful at measuring, understanding, and predicting business performance. While these words may sound wonderful, what do they really mean?

The old adage that “you can’t manage what you can’t measure” gives us a starting point. We measure business performance with metrics, of course. At the most basic level, metrics are numbers that describe things. Business metrics become more useful when they measure things and events that **experience has shown let us manage our business**. For example, a metric called “sales” tells us the total sales dollars for quarter one. The ability to report this metric is of some interest, but has limited value as a means of piloting a business. It may be necessary, but it is far from sufficient.

Useful Metrics

Pair this with a metric like “sales for last quarter”, compute their difference, and we have a business measure useful for piloting a business. We can now measure *progress*.

Let’s add other metrics such as sales for quarter-one this-year and quarter-one last-year as a percentage of year-to-date sales. We’ve now enriched our metric set with new

information that will automatically enable us to ask, and get answers to, *new* questions, like “why were we farther ahead last year by this time than we are now?”

In fact, the process of asking questions of data is a sound-byte definition of what computer-based analysis is. And an analyst is an expert who knows both what questions to ask and how to formulate *new* questions. This heuristic iterative cycle of question asking and new question generation across business metrics defines what might be called the “analytic cycle”.

OLAP Technology Defined

An OLAP technology is one that supports and empowers users through the analytic cycle. This is what *we* mean when here we write or say “OLAP.” And the minimization of the duration of this analytic cycle across *complex* business metrics is what DB2 OLAP Server is all about.

Useful Metrics = New Questions

The first business task that we have is to generate useful metrics from our data. In the hands of experts, these metrics *automatically* enable them to generate new questions and new answers. So, what are useful metrics? Useful metrics are numbers that contain other numbers in “compressed form”.

Aggregates

For example, a sum obviously adds up numbers by some criteria. (“Total sales for the second quarter for ISO channel products for the East Region.”) This aggregate becomes very useful when compared with another aggregate, for example the same criteria varied by time (“for last quarter”) or when varied by geography (“for the West”). One way to describe basic OLAP that is not too misleading is that it’s a technology for quickly comparing useful aggregates with other useful aggregates. It is, however, much more than that.

Beyond Aggregates

No MBA worth his or her salt is going to pilot a business based solely on totals. The situation quickly becomes fascinating as people trained in quantitative analysis begin digging for metrics that will help them understand, change, and predict business performance.

Ratios

Ratios are deceptively simple but highly useful tools for analyzing performance. “What percentage of profit did our family x products deliver last quarter in Asia Pacific versus family y products”. Or, “What market share does brand y enjoy versus brand z?”

Allocations, forecasts, and so on

Talented MBA’s and other people trained in quantitative analysis can’t wait to get their minds around different ways of looking at a business analytically. All of the numbers they generate in some way contain or represent other numbers in compressed form. Sums, averages, percentages, all are derived from base quantities and represent them in

some more general way that allows comparing, contrasting, predicting, and understanding a business. Modern enterprise is so complex that if we did not have some way of compressing and representing events, we couldn't manage them.

The Relationship of OLAP Metrics to Traditional Statistics

Suggesting that OLAP metrics are numbers that represent other numbers in compressed form invites a comparison with classical statistical analysis. We find that both the OLAP discipline and the discipline of statistics overlap and serve a similar purpose. Unfortunately, we also observe that statistical insights are usually not as compelling to decision makers as are often simpler OLAP metrics. While OLAP Metrics can be classic statistical measures like standard deviation and correlations, they can also be simple ratios and percentages. Telling a decision maker “we lost money on footwear in Massachusetts last quarter but made money in New York State” is a **lot more galvanizing** than to say “the variance of profit in the northeast was greater than in the southwest in quarter 1”.

Traditional statistics usually don't speak to the average business person. OLAP metrics do. While we'd like to see the day when statistical insight pervades all cultures, OLAP is pervasive now.

What are OLAP Questions?

We can see by the above discussion of aggregates that an OLAP question is not, “How much did Mrs. X spend on shoes in Singapore store 1234 on Saturday at noon?” But from this ‘fact’ a related or derived OLAP question would be, “How did the marketing initiative we put on apparel in Asia Pacific effect profit for footwear in 3rd quarter this year compared to 3rd quarter last year?” That's an OLAP question.

Why OLAP in a Data Warehouse book?

Ralph Kimball seems to have been the first to systematically argue that normalized relational database schemas supporting transaction processes aren't appropriate for supporting analytic processes. His idea of a dimensional model stems in a large way from the fact that analytic processing requirements are not the same as transaction processing requirements. ‘In the final analysis,’ a Kimballist would say, ‘a dimensional business model is more easily understood by business-users than a normalized business model is.’ Kimball also noted that speed advances in transaction processing came from altering the schema. We want to carry this observation further and suggest that advanced analytic requirements need to be supported by a different storage structure as well.

Dimensional Modeling

Dimensional modeling goes a long way toward supporting decision processing in at least two important respects. First, users understand a star schema more easily than other modeling pictures. Second, by properly configuring aggregation tables within these star schema structures, system response greatly improves. So the net result of star database schemas is a more responsive and user-friendly system.

The relational database star schema provides the lion's share of analytic value to a large portion of business users. The advanced use of OLAP technology, on the other hand, is more precisely focused and targets those people in the organization who *manage* the business. That is, it is focused on those whose job is to see that the business is viable and competitive.

Steering the Wheels

We can illustrate this well by extending an analogy Ralph Kimball uses in *The Data Warehouse Toolkit*, where he says that “users of an OLTP system turn the wheels of the organization.” and that “users of a data warehouse, on the other hand, are *watching* the wheels of the organization” (Kimball, p. 3.)

We propose that advanced OLAP users are the people who *steer* the wheels of the organization. They are looking at the biggest business questions possible: Are we profitable? Where? Why? Why not? The OLAP users are active. They decide where the enterprise is going, or provide information directly to people who do decide.

Speed-of-Thought Analysis

Ultimately, then, an analytic tool is a tool that enables business analysts to test different business models for the purpose of asking and answering OLAP-type questions. The speed at which this process occurs is critical. Does the analyst wait 1 second, 1 minute, 1 hour, or 1 day (or more!) before viewing the results that will lead to the next question?

The answer is important. The duration between question and answer can be of such a length that the originating question needs to be recalled and reformulated before results can be optimally interpreted. The necessity of having to constantly re-place results *in context* hinders the analytic process.

The OLAP functionality implemented by DB2 OLAP Server supports rapid question-and-answer response times and has been provocatively described as “*analysis at the speed-of-thought.*”

A drill-down is *much more* than a movement down to a level of higher detail, it is a report that remains intimately connected to, and answers the question that generated it. This is precisely why a drill-*up* is every bit as powerful for a business analyst as drill-*down* because they *both* are question-answer reports in context. To repeat, a drill down is a report in context.

Each report is new, arrives instantaneously and is not constrained by the boundaries of canned reports that use replaceable parameters. The rapid question-and-answer functionality that accrues to DB2 OLAP Server is truly definitive for the business analyst. It is difficult to overstate the analytic value of this functionality.

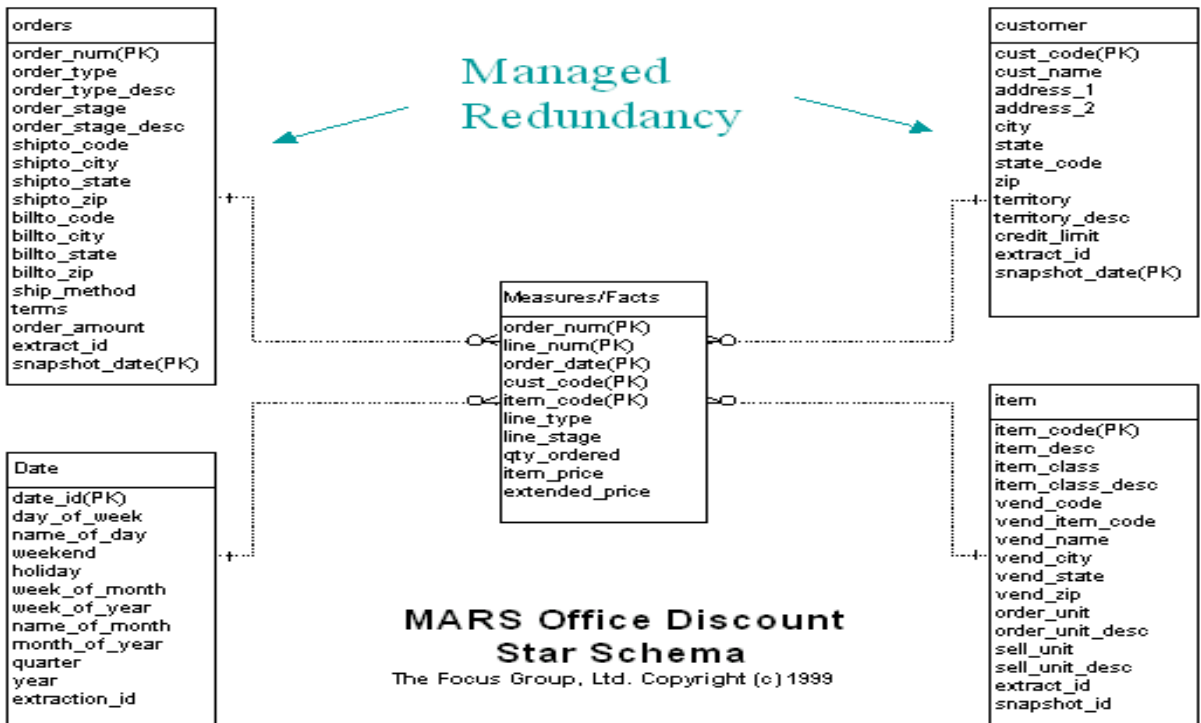
We are suggesting that the *quantitative* change in speed of answers makes a critical *qualitative* change in the ability to pilot the business. Because users can think more quickly they are freed to think more deeply and in an unconstrained manner. They are

able to be more creative. Using DB2 OLAP Server you can steer the enterprise faster, which means that you can steer it *better*.

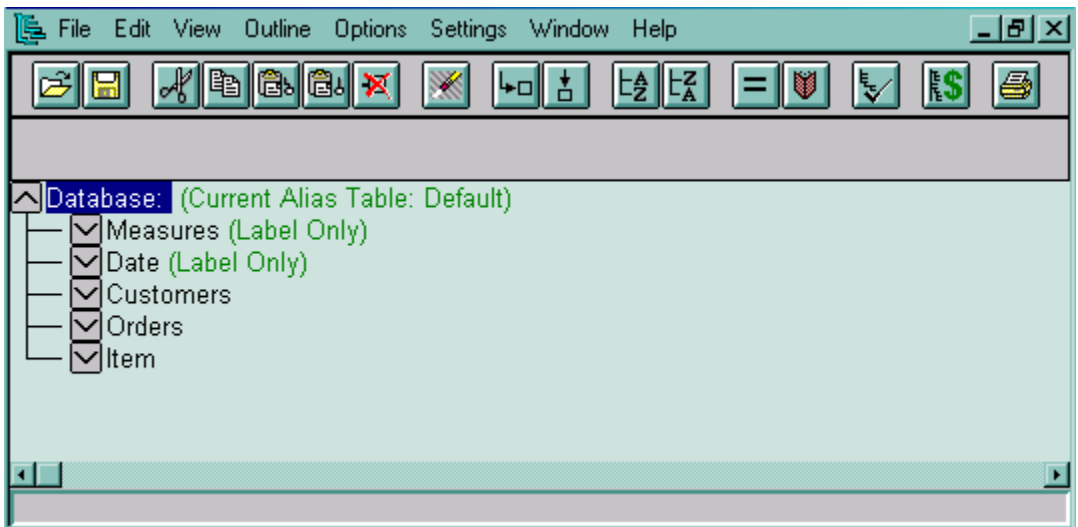
New Schema = New Structure: What is an Outline

The DB2 OLAP Server Outline is a place where the data model and the business model are the same, and are represented graphically. The DB2 OLAP schema is an “outline” of the business. The following discussion will illustrate.

The relation between a star schema and a DB2 OLAP Outline (schema) is illustrated by the following two diagrams:



This star schema is represented in the following DB2 OLAP Server database Outline:



There are many differences to take note of and we will proceed briefly to describe only the most important. The first is that DB2 OLAP Server implements an array storage structure.

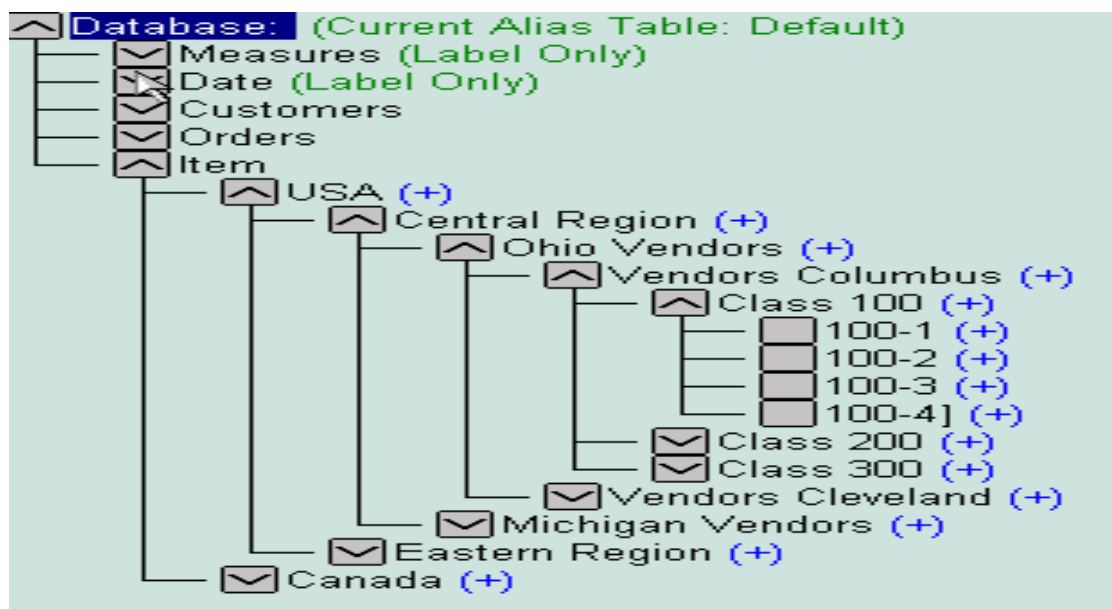
Relational Database Contents Constitute the Array Schema

The relational star has **five** tables (four dimension tables and a fact table) and the OLAP outline schema has **five** dimensions. The five dimensions can correctly be thought to represent tables. The business, however, is being modeled and stored differently in these two databases.

In the relational schema each business entity (for example, product code 100-1) is stored as part of the *contents* of a schema component (i.e. table). These contents are accessed using SQL. On the other hand, in the DB2 OLAP Server Outline, instances of actual named business entities *are* the database schema. In the star schema, product code 100-1 is the name of a place where a specific item code is stored for each business event being managed. In DB2 OLAP Server, each specific item_code is an entry in the schema.

The DB2 OLAP Server schema is a way of picturing an enterprise that is a closer approximation to how business people experience the business than relational schemas are. Business users experience specific named entities, like product 100-1. They don't experience the "categories" of entities like a table column called item_code, which are the meat of relational dimensional modeling.

It is quite amazing to visit users and see that, day-in and day-out, they live with complex coding references. Everybody within the same work group would know that 100-55 is a **new** product for which the tooling hasn't been done yet, but the marketing brochures already tell the story. Users will exchange knowing glances during OLAP design sessions about specific arcane codes that even a close outsider from IT can barely understand. When users see their codes embedded in a DB2 OLAP Server "business model" (outline) they "get it" right away. That OLAP database metadata is composed of business (not database schema) entities is manifest by the fact that the best and most effective OLAP database designers are business users (see discussion below on **Support Requirements**.)



You Can See the Hierarchies

In a similar way, OLAP outlines make obvious the way the business aggregates. Note in the outline fragment above that the “form” given to the enterprise by the rollup structure is immediately obvious. This is important. Three inches of green bar paper of printed account codes is an organizational *list*, not an organizational *structure*. The DB2 OLAP Server outline, however, not only shows the list, it allows these details to reveal the organizational the structure.

Information Hiding

Data processing people have long provided for themselves the ability to hide information in order to disclose form. The invention of the program subroutine and modular programming provided a way to isolate details but also to emphasize logical form. Programmers have long used this ability to understand the programs they write. This powerful capability is known as information hiding and has become such a staple of programming that it is no longer even mentioned.

Structural Help

In the same way we can create computer programs we can’t understand without help, we can create organizations we can’t understand without help. In OLAP Server, double clicking on an outline parent expands or collapses the information. Many users have never been able to visualize how account codes report to other account codes, but they can when they use a DB2 OLAP outline. The information hiding ability of the OLAP outline gives to business analysts the schematic clarity long available to data processing analysts.

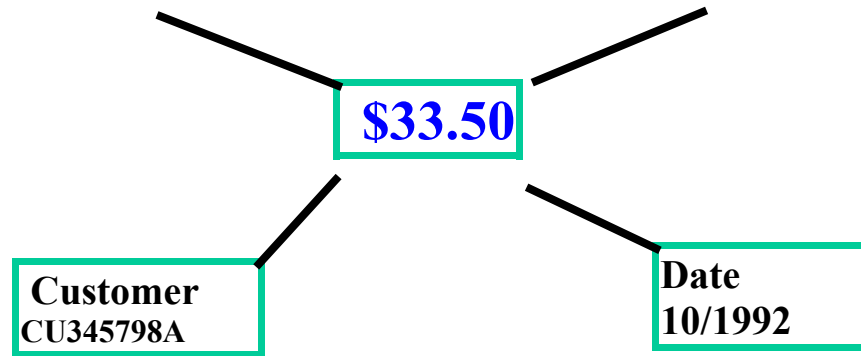
In some OLAP modeling sessions, users will excitedly grab the input mouse and start moving schema objects around. This ability to move, reshape and restructure the organization at the computer desktop is a core part of On Line Analytic Processing experience that is provided by DB2OLAP Server.

“What if Alice’s group reported to the Northeast?” can be modeled almost in one mouse-click. Users are close to their data and close to their organization’s structure (i.e. multidimensional metadata). In a DB2 OLAP Server environment, *you are automatically close to both*. This is powerful organizational and analytic modeling. It is made possible by the pictorial nature of the OLAP Schema where the business model and the data model are identical.

The OLAP Array

The entity names in the OLAP Server outline are called “members”. DB2 OLAP server





stores a number at the intersection of every member by every other member in the outline, across all dimensions at all levels of hierarchy. The storage structure supporting this storage is a multidimensional array. Accessing numeric data in a DB2 OLAP Server array is achieved by supplying appropriate co-ordinates or intersection points.

IBM DB2 OLAP Server is a multidimensional OLAP (MOLAP) tool. In DB2 OLAP Server, the array is defined to hold only numbers and numbers only are “data” to the tool. Some relational “data” (which has become “metadata” to DB2 OLAP Server) is stored in a physically separate Outline object. Pointers connect Outline members to array storage locations. From this architecture derives much of the modeling quickness that OLAP Server provides. To manipulate the outline schema, you don’t need to touch a vast data array. To manipulate the vast data array, you touch the Outline as needed. The partnership between objects is intimate, patented, tested, and found to be compelling.

Business Logic in the Schema

As mentioned above, within the Item dimension SKU-level members aggregate to class members, class members aggregate to vendor members, and so on. Organizational structure is therefore represented *by* the database schema, and *not* as contents of the schema. The point still bears further emphasis and consideration and we want briefly to sketch here what we perceive as a limit on the ability of relational modeling to successfully represent enterprise structure and business rules.

Asymptotic Limits

A relational database schema (star or otherwise) can only be said to approach business logic asymptotically. This is so precisely because the business entities are really *contents* of relational database schema objects. Schema components reflect business logic according to the way(s) that they are *related* to each other. But the schema objects themselves are not business entities.

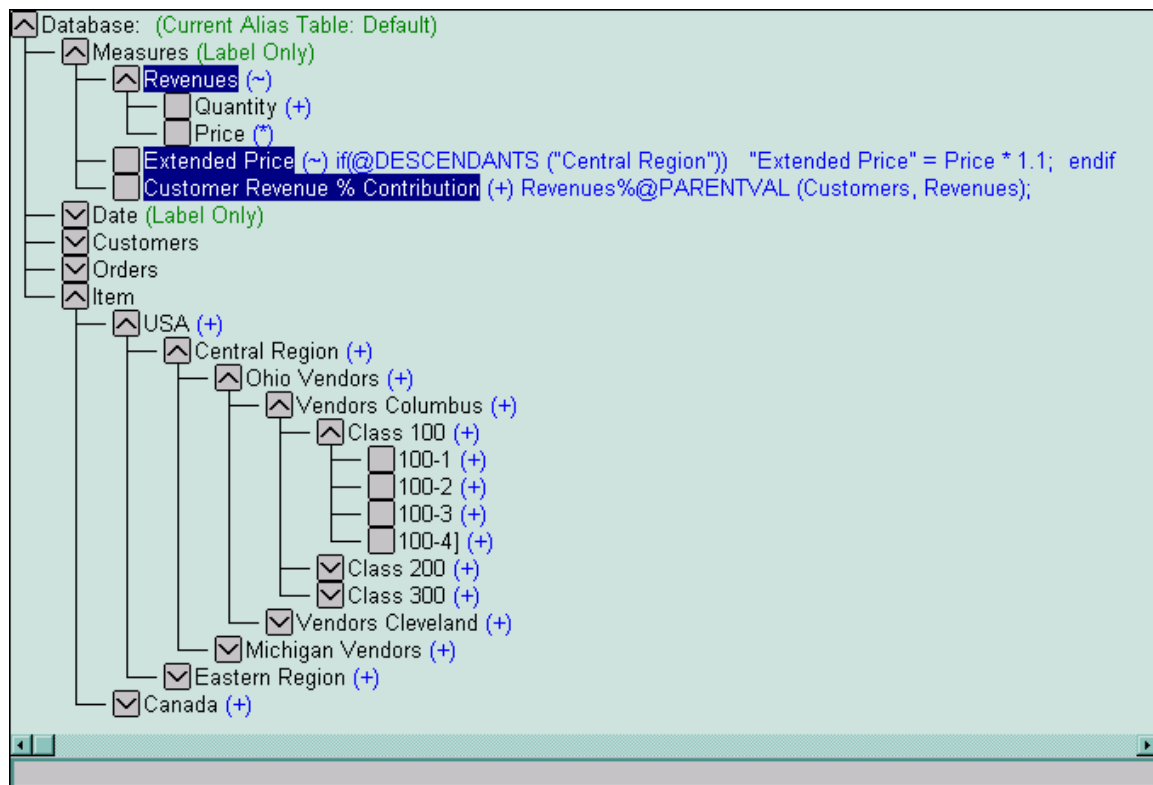
Even though the suggestion that database schema objects (like tables and indexes) should represent business entities probably sounds idiotic to relational data modelers, DB2 OLAP array schema objects (like dimensions and members) do. Array schema objects can also reflect relationships, as in the simple example of days aggregating to months.

Indeed very complex business relationships, or business logic, can be easily embedded in a DB2 OLAP Server schema. The point we want to emphasize here is that the array storage structure *removes a level of abstraction*, and thus enables modelers to alter the schema and more efficiently and transparently create *new* business models.

Using our example above, the level of abstraction that is removed is that in the OLAP Server outline there is no column name called `item_code` whose contents instantiate the business structure (and rollups) to be modeled. This level of indirection, as it were, is eliminated. Users are directly connected to the business names they use daily. We suggest that column names containing instances of business names constitute an asymptotic limit to user understanding which relational modeling, as powerful and successful as it is, cannot by definition cross.

Derived Measures

Consider the following DB2 OLAP schema extract where the Revenues metric is calculated by multiplying the Quantity by the Price metric:



The derived value (**Revenues**) is in no significant way different from other members and is also part of the database schema. It is important to understand that in the above DB2 OLAP Server database, **Revenues** are calculated (see **Batch vs Dynamic Calculations** below) for every intersection point across each of the other dimensions in the database (i.e. every Item by every Order for every Customers by every Date.) **Extended Price**, however, is calculated only (note the Boolean syntax within the formula) for members intersecting with the descendants of the Central Region Items. Finally the **Customer Revenue % Contribution** formula relates the revenue contribution of each customer with its parent (as a percent). Once again this calculation would occur across every other dimension in the database.

These are simple examples that illustrate the manner in which very complex business logic can be modeled using DB2 OLAP Server. By removing a level of abstraction present in relational dimensional modeling, we have “dropped down” into an arena where we manipulate business names directly in the picture object (Outline) to help reveal the underlying business structure. We also add computations to the aggregations to reveal and test business rules. The point is that Business Analysts are very comfortable using DB2 OLAP Server to continue to manipulate business hierarchies and logic to create complex metrics to be used in piloting the business.

Measuring the Measures

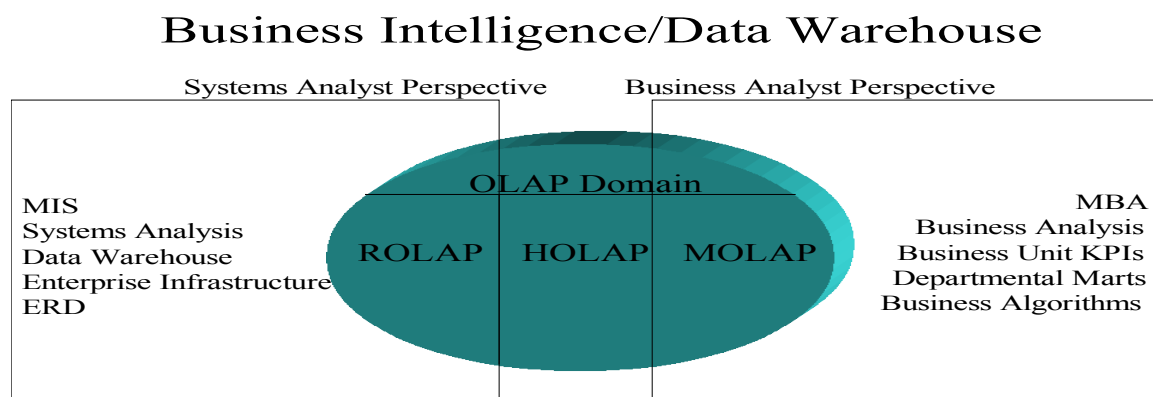
Business users are generally pleased to see that complex business analytics, hitherto hidden away in database programming logic, surface in the OLAP outline and are literally published for the entire user community to see, understand and critique. **“You can’t manage what you can’t measure” applies to the measures, too.**

The Skills View of OLAP

In recent years Business Intelligence (BI) products arose claiming to be “MOLAP” (Multidimensional array storage supporting OLAP), “ROLAP” (Relational storage supporting OLAP), or “HOLAP” (Hybrid storage supporting OLAP) in nature. We intend to stay out of strict definitions of these differences in this discussion, except to point out that their relative positioning becomes clear when you evaluate the analytic world by skill set.

We suggested above that a systems analyst looks at column names whereas a business analyst looks at business names. Exploring the differences between how systems professionals and business professionals see things will be useful to help further understand OLAP.

Consider the following diagram depicting perspectives of the BI world:



The way one views OLAP depends upon what side of the analytic space you come from. The whole “rolap, holap molap” debate comes from this difference.

The systems analysis perspective and the business analysis perspective apparently divide the BI world. The relevant academic degree on the “left” is from Computer Science whereas the relevant academic degree on the “right” is from Business Science. Key business performance indicators drive the Business Analysts, while database or system application performance indicators drive the Systems Analyst. The view on the left is systems architectural in nature, the view on the right is business logical in nature. *Neither view is entirely correct.* Or, rather, **both** together are correct, which is to say that the skills differences between Information Systems professionals and Business Management professionals cause them to understand analytic processing in different but essentially complementary ways.

For the Systems Analyst, entity relationship diagrams rule the world, whereas for the MBA business algorithms rule it. Many Business Analysts think at the departmental level because business performance indicators are usually relevant to some specific task-set that maps to a specific organizational subsystem. Systems Analysts, on the other hand, more often think at the corporate or enterprise level because data warehouses are thought of as corporate application assets, not departmental tools. Business analysts who do think at the all corporate level have a particularly disquieting job, as metrics need to be defined and aggregated for the corporation as a whole and stored somewhere that permits asking OLAP questions of them in the analytic cycle at the speed of thought. The physical location of this analytic cube in the corporate information structure has never been easy or very clear.

As in most things, “where you stand depends on where you sit”. People skilled in business analytics are likely to be drawn to an OLAP tool that presents to them things they understand, like business names. People skilled in systems analysis are likely to be drawn to an OLAP tool that presents to them things that they understand, like column names.

On the one hand, we do not want to comment on the appropriateness of any *specific* tool for any given task. Since pure objectivity on this matter is unlikely, it is useful to know the background from which tool recommendations are being made. On the other hand, we do want to suggest that there are more or less appropriate tools for every task and that it is the mandate of data warehouse architects and to understand and implement tools that generate the largest return on investment (ROI) for their organizations.

In the final analysis it makes as little sense to attempt build a single OLAP cube and call it a warehouse as it does to populate summary tables at every possible intersection point across every conceivable column combination in the DW and call it OLAP!

The debate about Relational vs. OLAP technologies is misguided. It is time to recognize Relational, OLAP and Data Mining as **complementary** technologies.

Enterprise OLAP Architecture

The above discussion made several observations that we now want to bring and briefly examine in closer proximity with each other:

1. Many KPIs are relevant only within organizational subsystems
2. A star schema implies one OLAP cube and possibly many more

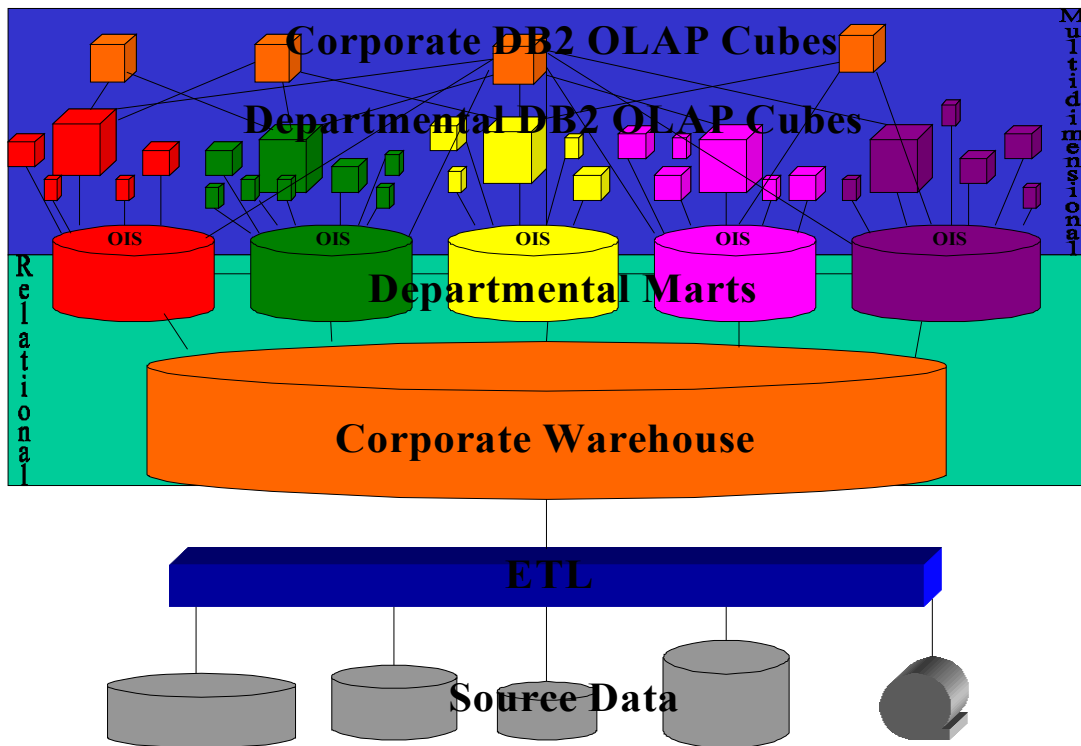
3. OLAP application development delivers rapid ROI
4. OLAP applications steer the business

It follows from the first statement that there are metrics crucial to the management of departmental business units that are not appropriately stored within the DW. For example, at the Corporate-level a Bank does not care which customers make the best loan candidates. The Bank only cares that the loan operation is profitable. The loans division of Retail Banking Department clearly does need to track loan customers at that level. Many Departmental level KPIs are different from the corporate-level KPIs that have to apply across the organization. The former properly need to be derived within departmental data marts.

We have suggested above that the departmental business analysts are experts at manipulating these metrics in order to test and steer their respective units, and, moreover, that OLAP technology really targets these individuals as the ideal advanced user group. Using a DB2 OLAP Server environment to provide complex department-specific metrics for Business Analysts, we can end up proposing an architecture that is really a variation on Kimball's bus architecture.

The diagram attempts to isolate BI/DW *data* structures. Non-horizontal lines that link objects generally indicate data flow and move from the bottom-up. The horizontal connecting lines between Departmental Marts indicate a conformed rather than a stove-pipe relational data mart environment. OIS is the MOLAP generating tool between relational star schema structures and DB2 OLAP cubes.

Note the Corporate DB2 OLAP Cubes. They are the same color as the Corporate Warehouse because they contain metrics of interest at the corporate level but are (can be!) created by connecting information contained in Departmental DB2 OLAP Cubes. This really illustrates that the proposed architecture is a variation of Kimball's bus architecture in that certain departmental or subject oriented database metrics are brought together to comprise a corporate perspective.



The graphic should not be taken to prescribe any particular relational architecture. Whether, for example, the **Departmental Marts** in the picture taken together is considered the ‘Warehouse’, or whether there is a highly normalized database structure like that represented by the **Corporate Warehouse**, is irrelevant. We consider DB2 OLAP Server as essentially neutral regarding Relational *architecture* except that OIS needs star schemas to perform efficiently.

We not want to convey any ideas regarding the relative *importance* of MOLAP vs. Relational analytics. Space constraints make it difficult to adequately depict the important *relational* components that provide analytic reporting and underpin enterprise-wide analytic (ROLAP or otherwise) applications. We want to emphasize *where MOLAP resides within the BI/DW and position it as necessary and complementary technology*.

In the production environment, organizational subsystem data is cleansed and scrubbed within the relational database. This cleansed data is used as source data that feeds star schema data marts. These data marts can be typical Kimball marts in every respect except insofar as they will assume characteristics necessary to support DB2 OLAP Server cube creation and maintenance.

The polar discussion/debate concerning whether to begin constructing the entire warehouse or to begin building the data marts is very worn today. Clearly a coordinated effort is required to build a functional and efficient BI environment. But we want to point out that OLAP cubes quickly provide ROI by fulfilling business analytic requirements within a 30-90 day timeframe. Moreover, cube schemas can be used to help define

underlying star-schema requirements that, in turn, can be used to refine source data requirements for the warehouse.

In this way, DB2 OLAP Server can have a powerful role in **prototyping the data warehouse**. Organizations have found that it is a relatively inexpensive, safe, and effective way to start a warehouse project. You deliver valuable business answers to grateful users while simultaneously discovering that your enterprise has five product master files, three general ledgers, and that “customer x” is spelled 7 different ways. Since multiple metadata pointers to the same metadata entity (10 ways to spell “AT&T”) will not pass edit in DB2 OLAP Server, you get to conform your designs as you proceed to quickly deliver individual actual working models. Then, the harder work of creating permanent data scrubbing and cleansing processes for overall warehouse use can begin, but in the context of a successful although bounded delivery of answers.

Sequentially adding OLAP applications while simultaneously conforming the dimensionality of the underlying relational data marts effectively implements (M)OLAP across the enterprise. A fully functional BI/DW can be implemented by coordinating (conforming?) the design of the OLAP supporting relational schema(s) with the relational Warehouse schema that provides enterprise-wide analytics using ROLAP or other SQL technologies. In other words, relational designs that support OLAP requirements can effectively be used to prototype the warehouse. More importantly, the warehouse is gradually constructed ***through the process of providing business-relevant metrics that are used to produce a rapid ROI for the organization.***

Building a BI/DW environment through OLAP applications offers IT the ability to *minimize* application development time, *maximizes ROI* and create a data repository that is conformed across the enterprise, provides pertinent high- and low-level analytic functionality to business users from the start. Hence it is our view that relational and OLAP development needs to be coordinated, and done in parallel.

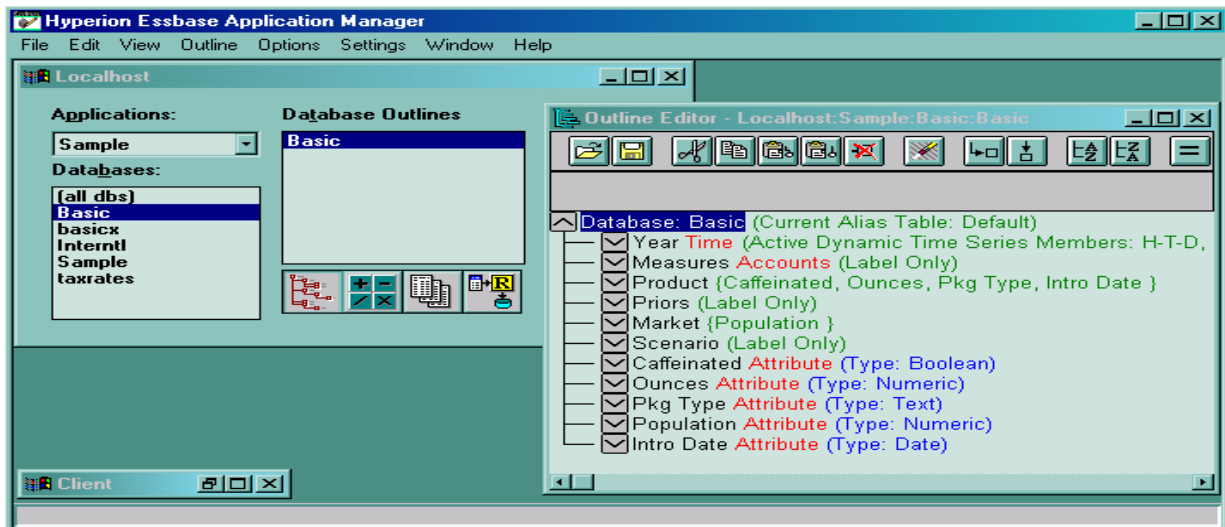
Given this high-level description of enterprise OLAP, what are some of the practical considerations requisite to implementing DB2 OLAP Server?

Database Design: Building Outlines

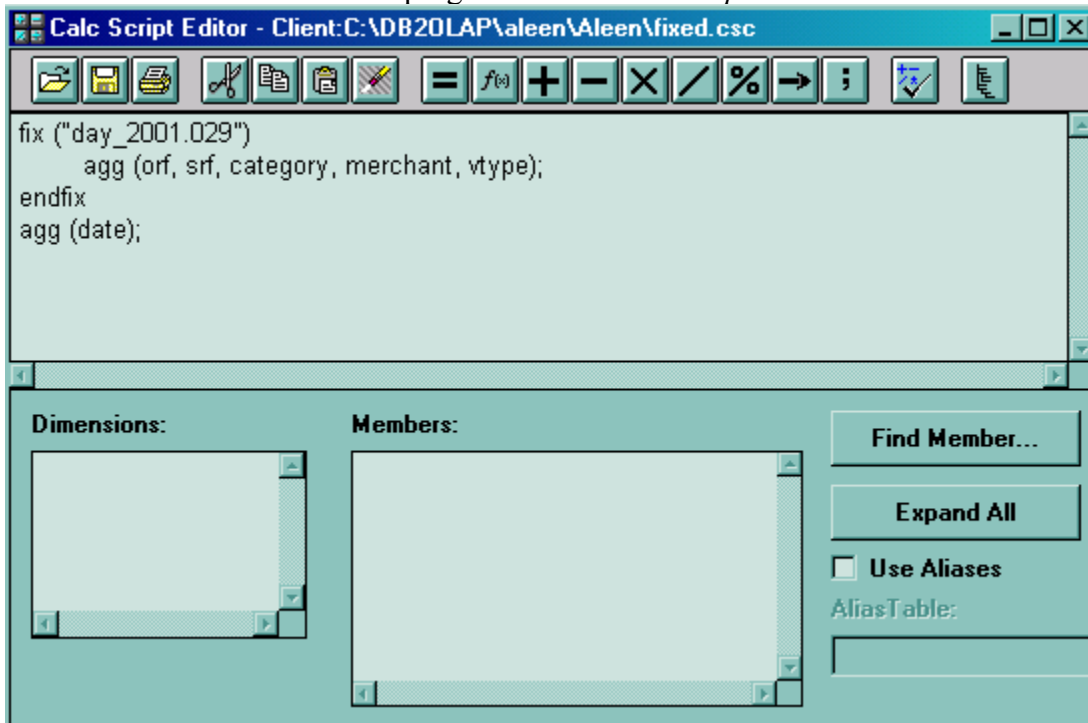
There are 3 ways to build a DB2 OLAP Server Outline. Users can manually build dimensions and members through the Application Manager GUI, they can build a coordinated set of files and generate the outline in batch mode using the ESSCMD or MaxL utilities, or they can use OLAP Integration Server (OIS.)

Application Manager

Application Manager (App Man) is the central administrative tool for DB2 OLAP Server. It offers all administrative functions for building and maintaining DB2 OLAP applications and databases. Through App Man an administrator can modify database schemas by manually altering dimensions and dimension members. They can also:



- create database calculation programs called *calc scripts*



- create *rules files* for building dimensions and loading data

Data Prep Editor - Client: C:\DB2OLAP\aleen\Aleen\Level.rul

	LEVEL0,Product	LEVEL1,Product	LEVEL2,Product
1	600-10-11	600-10	600
2	600-20-10	600-20	600
3	600-20-18	600-20	600

- generate reports

Report Editor - Client: C:\DB2OLAP\aleen\Aleen\Asym.rep

```

<PAGE (Measures, Market)
South Sales
<ASYM
<COLUMN (Scenario, Year)
Actual Budget
Jan Feb
<ROW (Market)
<IDESC "100"
!
  
```

- perform user security definition and maintenance

User / Group Security

Server: Localhost

Users:

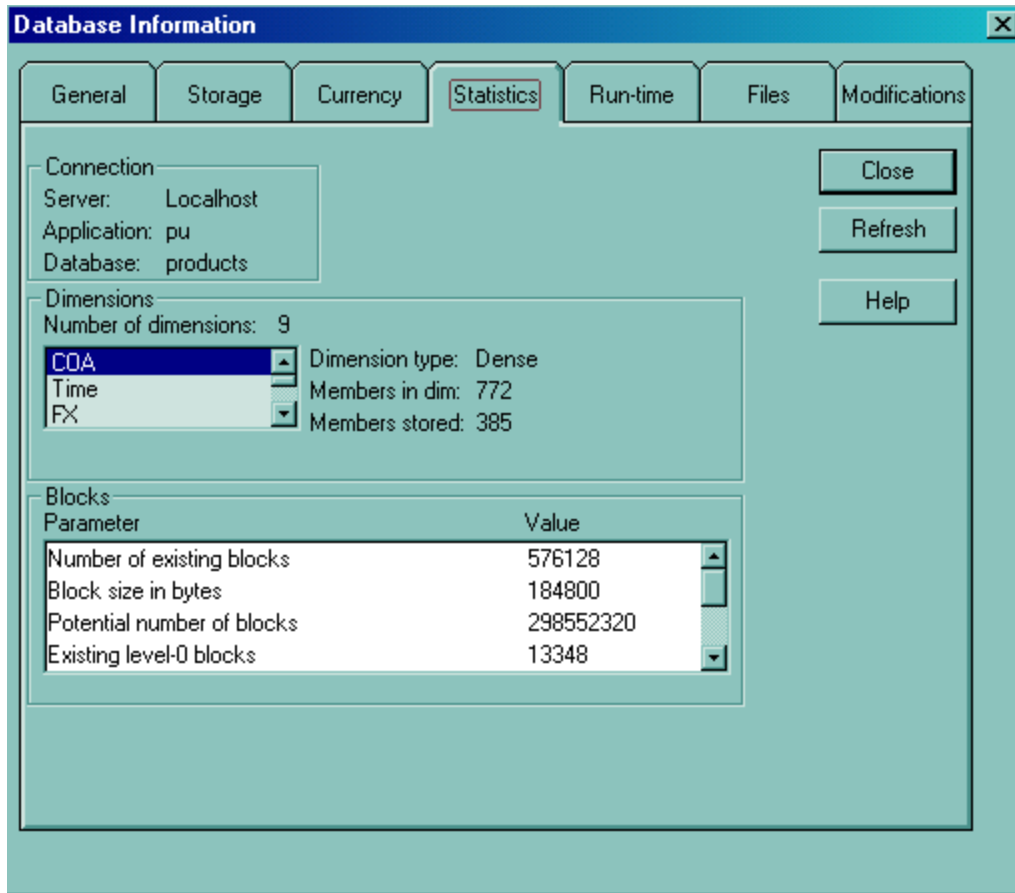
Admin
essbase
hyperion
me
supervisor

Groups:

Administrators

Buttons: New User..., Edit User..., Copy to New..., Delete User, Rename User..., New Group..., Edit Group..., Copy to New..., Delete Group, Rename Group..., Close, Help

- monitor database statistics reported at the database, application and server levels



- and perform routine database maintenance routines as required.

ESSCMD & MaxL

The command line utilities that supports 100% of App Man functionality in batch mode are ESSCMD (pronounced es command) and MaxL (pronounced mac sel.) Traditionally, the database designer uses App Man to create Rules files that enable external files (of specific design) to be used to build database dimensions (dimension build Rule files) as well as populate databases with data (data load Rule files.) Users have the ability to load data from flat ASCII files, spreadsheets or from RDBMS tables using the optional SQL Interface.

An administrator would be responsible for creating and configuring a batch environment to effect a lights-out production environment using ESSCMD or MaxL. (For a complete introduction to App Man, MaxL and ESSCMD, the reader should refer to the documentation that accompanies DB2 OLAP Server.)

Application Manager, ESSCMD and MaxL enable a powerful and flexible operating OLAP environment. Data can be loaded to a DB2 OLAP database from nearly every conceivable ODBC data source that includes flat files, RDBMS tables and spreadsheets.

Moreover, DB2 OLAP Server supports out-of-the-box database write capabilities. This functionality is crucial for supporting, for example, budgeting applications where the users need to be able to quickly adjust and re-adjust figures and test outcomes. In this

process, users that have an appropriate security profile can write data to the database directly from (standard) spreadsheet client desktops.

Security can be maintained from providing users with full database access all the way down to having access to a single-cell. DB2 OLAP Server supports a development of a full complement of OLAP applications out-of-the-box, and applications can be completely user-defined and dynamic through the fully functional C and VB APIs.

The ability to build and load data from multiple sources eases prototyping efforts. Furthermore, the App Man Outline development GUI can be used to function very effectively as a rapid application development (RAD) tool enabling business users to convey complex business notions to IT efficiently and with minimal effort.

Batch process limitations

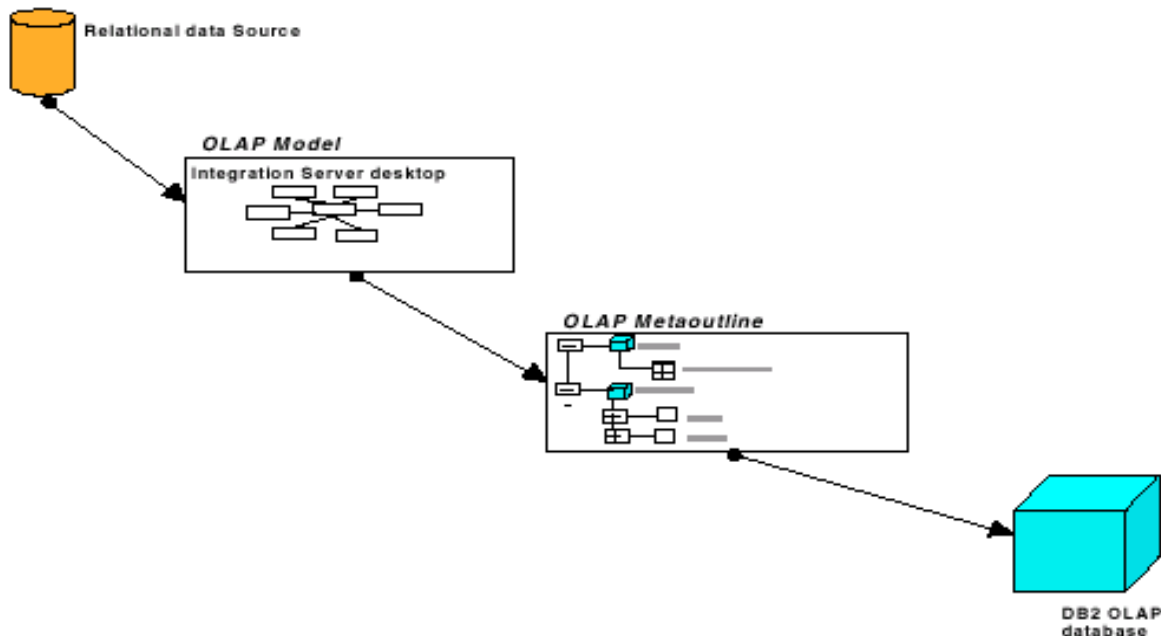
Although the APP Man, ESSCMD/MaxL tools represent a powerful OLAP application development setting, they really are best applied within small information systems environments.

What works for application prototyping does not necessarily work in a production environment and new OLAP application development here remains unfortunately tied to a cumbersome development process. New business models can be generated only after the newly defined set of supporting dimension build and data load files have been created. The result is an environment where a potentially chaotic proliferation of files and/or database tables has to be monitored and maintained. More often than not, these environments are de-centralized and re-use of an object is more a result of luck than planning and design.

To bring OLAP to the enterprise, what is needed is a tool that effectively coordinates relational (star schema) metadata with OLAP metadata.

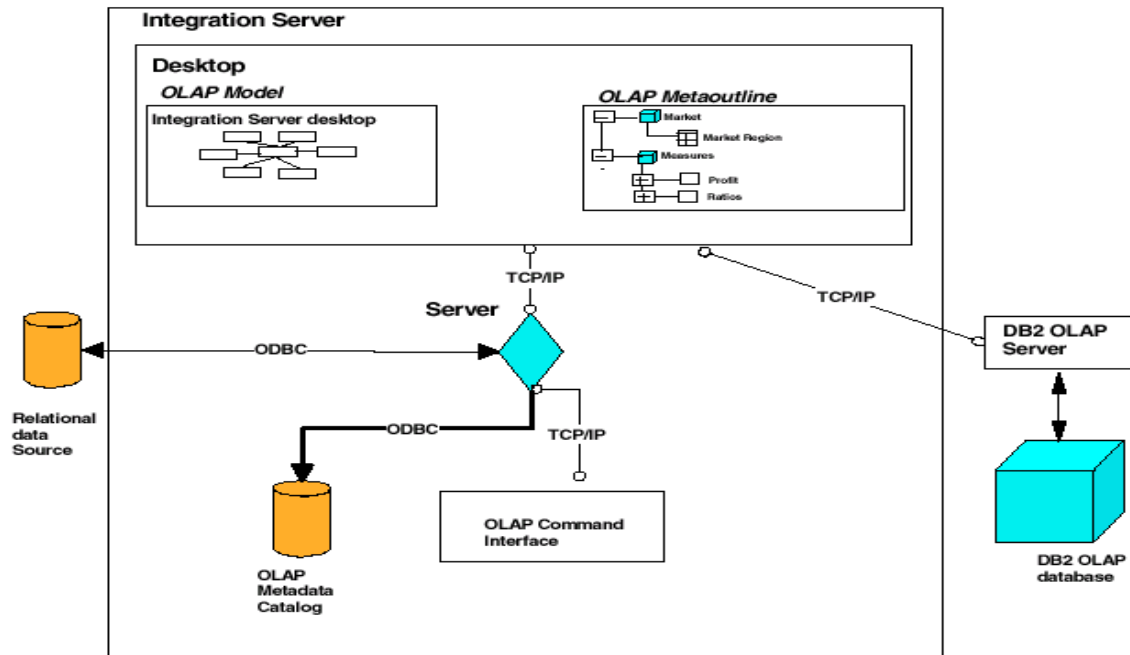
OLAP Integration Server

DB2 OLAP Integration Server very precisely fits the role as metadata coordinator. Using OIS, a user can create, populate and then calculate databases directly from RDMS tables.



The diagram shows the basic outline of the steps required to move relational data and metadata to multidimensional data and metadata.

In greater detail, the architecture looks like this:



The **OLAP Model** “understands” source relational metadata. Users have the ability to select from a subset of the **OLAP Model** objects to generate **OLAP Metaoutlines**. From an **OLAP Metaoutline** the user is able to generate a single DB2 OLAP database. The ability to create many **OLAP Metaoutlines** from a single **OLAP Model** enables users to efficiently create many OLAP cubes from a single relational star schema.

Relational data mart designers should take advantage of the one-to-many relationship between **OLAP Models** and **OLAP Metaoutlines** and design the relational database *to support the generation of multiple OLAP databases*. Which is to say that data mart design and contents be expressly expanded to let users take advantage of this functionality. For a concise overview of relational schema design techniques to support OIS driven DB2 OLAP Server database development, please refer to “**Appendix B. Integration Server implementation guidelines**” in the *IBM DB2 OLAP Server Theory and Practices* Redbook, IBM Inc., 2001.

The slides show the correct flow of data and metadata but an incorrect ordering of *development* events. The best-practice OIS methodology actually works *back* to the relational star requirements from a set of OLAP requirements. Once the analytic requirements are sketched out for an OLAP database, star schema relational requirements can be developed. After the dba has created the appropriate **Relational data Source** (star schema) to support **OLAP Model** generation, **DB2 OLAP Databases** can be created, populated and calculated using the OIS **OLAP Metaoutline** interface.

OIS is a Super-User Tool

In fact the OIS tool supports the database creation activities for super-users, and the tool really aims at these individuals as its intended users. The advanced use of cube

generation functionality is sometimes referred to as cube-on-the-fly. The idiom attempts to convey that users can efficiently generate a *new* DB2 OLAP Server database for further analysis as a result of questions and issues generated while performing analysis in *another* cube. To support this, OIS enables users to select a subset of metadata (dimensions *and* members) and numeric data from the data mart and generate new OLAP databases.

The real enabler of cube-on-the-fly functionality is the design of the star schema that underpins cube generation. The relational schema must contain data that supports these ad hoc cube-creating excursions through data.

For example, suppose that a KPI on an Executive report prepared from data in a corporate cube for the CFO indicates a problem with sales in the Eastern Region for Diet Products. The CFO might pick up the phone (or have someone *else* pick up the phone) and ask the Eastern Region Manager of Diet Products something like, “What is going on over there in the East?”

In a properly designed OIS/OLAP environment, a data analyst would quickly create a new **OLAP Metaoutline** from an existing **OLAP Model**. A new cube would be generated having a dimensionality and data that narrowly focuses on the details of Eastern Region and Diet products. These details would not have been present in the corporate cube. Moreover, OIS can design cubes that enable a *drill-back* to even more granular relational detailed data stored in the star schema to further assist the Analyst in her task!

Once the investigation is completed to the CFO’s satisfaction, the new cube can be thrown away. But the prerequisite would be that these Regional and Product *details* be available for use on-demand *in the Star schema*. Ultimately then, everything rests on the details of the design.

To sum-up, the concept of cube-on-the-fly and throw away cubes really presupposes an environment where the OLAP databases are being designed and generated by users. If so, then what is the role of IT within an enterprise OLAP environment if it is not to design and build the OLAP cubes to user specifications? After all, creating database schemas is not generally considered to be a user function.

Support Requirements

Let’s examine the statement that DB2 OLAP Server metadata is comprised of Business entities. The most important implication that we want to draw here is that, with minimal technical training, business users can become excellent OLAP modelers. As we said above, they “get it” really fast.

Removing the layer of abstraction extant of the RDBMS schema components enables the organization to appropriately *keep* business logic in the hands of the business user and also *put into* their hands the ability to embed that business logic into a highly functional computer-based structure called DB2 OLAP Server. OLAP application development does *not* have to consist of transferring/translating business logic to a technician so that the technician can transfer that logic to a schema. In OLAP Server, business users build

business models. The collaboration of IT and Business users in an OLAP environment enables each to remain largely within the domains of their respective expertise.

Requirements Supporting Enterprise OLAP

Business users who build OLAP models (i.e. super-users) need to know what buttons to push and what key-strokes to make in OIS and App Man to enable them to achieve their objective. There are relational and multidimensional components that have to be understood in order for users to meet these requirements. This is why this group of users are *super*.

In order to acquire this skill set, all that is really required is that they receive the appropriate end-user training (see courses listed in **Training Programs** below.) These users will have to understand enough about storage implications to be able to understand why they need to work very closely with IT in the database creation process.

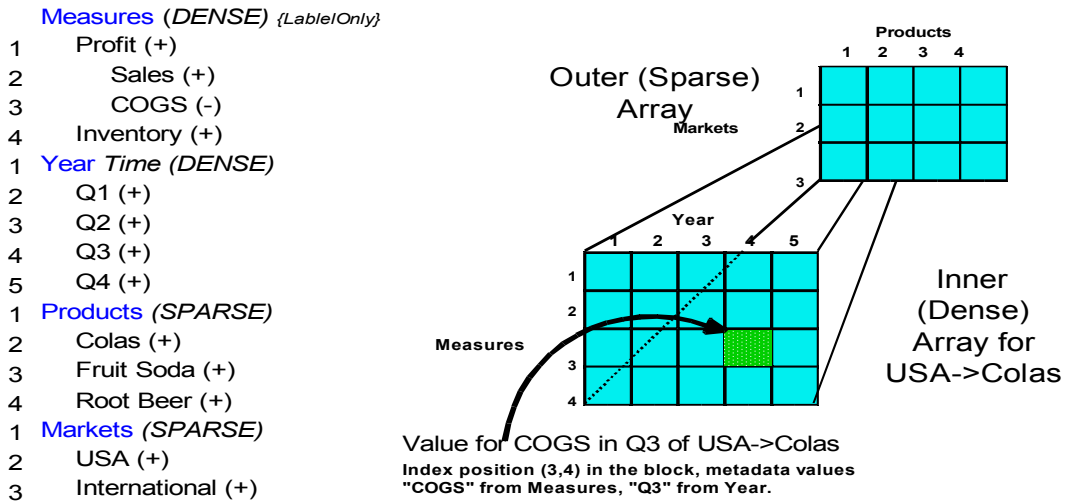
Are business users being set free to run rampant within an enterprise infrastructure? No, they are not. The traditional role of IT is expanded in one way. Of course IT needs to know in detail the hardware requirements necessary to support the entire OLAP environment. And in order to do this effectively, IT must know the details about the storage implications of DB2 OLAP Server databases.

Someone in IT should follow the training path outlined for system administrators. But they will especially have to understand in detail the concepts related to sparse matrix management as they pertain to DB2 OLAP Server. This is because OLAP storage constructs are, as will be seen below, data driven. This is to say, that the underlying sparse nature of OLAP data sets is a major factor in the resource requirements necessary to compute them.

DB2 OLAP multidimensional databases are matrices

The notion that DB2 OLAP storage structures implement data storage as a matrix (or array) is perhaps the most important concept for a DB2 OLAP developer and IT resource manager to learn. Understanding matrix management eventually will include an understanding of the corollary concept of sparseness. That is, as more dimensions are added to the matrix, proportionally fewer intersection points (or cells) across the matrix actually contain values.

OLAP Server Array Storage Model



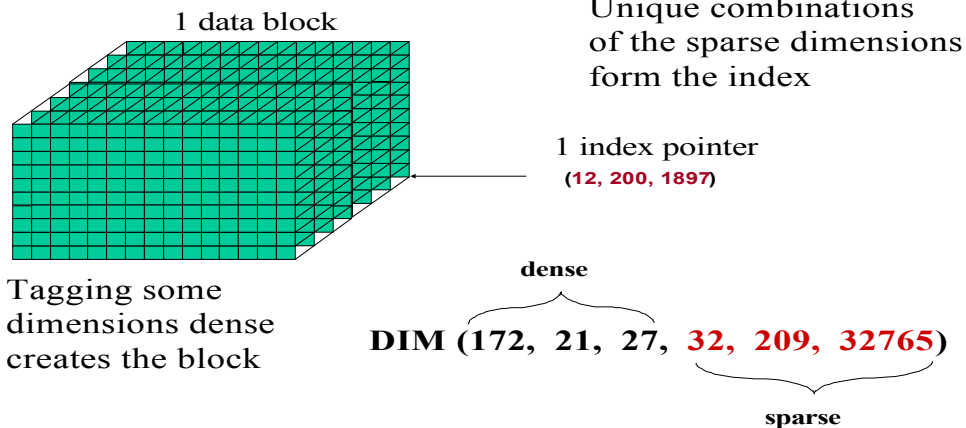
The data block and index explained

Consider the following array declaration containing 21,370,660,375,680 intersection points:

DIM (172, 21, 27, 32, 209, 32765)

The creators of Arbor Essbase enabled dimensions to be tagged or defined as either dense or sparse. When a dimension is tagged as dense, it becomes part of the storage structure called the data block.

OLAP Storage Structures Explored



Every data block that is created in the database has an identical structure. In our example, it contains precisely $172 * 21 * 27 = 97,524$ cells, or intersection points.

All data blocks are stored on disk within the ESS*.PAG files. Addressing, or locating, blocks of data is provided by means sparse member combinations. These combinations become part of the storage structure called the index and are stored on disk with the ESS*.IND files.

Enabling these two definitions of dimensions, the creators of Arbor Essbase made the matrix modular. The data block is a fixed format data structure the existence of which is driven by data-relevant sparse member combinations in the index. By data-relevant we mean that *only where business data actually exists across sparse member combinations will a data block be generated.*

So, for example, if we do not sell any Sun tanning oil in January in the Arctic, we do not reserve any space in our array for those intersection coordinates. One of the differences between the DB2 OLAP storage structures and relational ones is that a relational index is optional. In DB2 OLAP the index is not. Deleting an index for a relational table has no effect on the table data. Deleting the index from a DB2 OLAP database corrupts the database.

The small subcomponents of the array (the data block and its index address) are quite readily moved between disk and working memory. These structures mesh very well with the general user requirement of only being interested in sub-sets of information from the array at any one point in time.

Block creation explored

The above database above contains 6 dimensions with the following DB2 OLAP configuration:

1. **dense** dimension #1 containing 172 members
2. **dense** dimension #2 containing 21 members
3. **dense** dimension #3 containing 27 members
4. *sparse* dimension #1 containing 32 members
5. *sparse* dimension #2 containing 209 members
6. *sparse* dimension #3 containing 32,765 members

Which data blocks are actually created depends upon unique sparse combinations that contain data.

In our example, a block with address **(12, 200, 1897)** has been generated because *and only because* a business event has occurred at that intersection point. We could convert or translate the index node into something like “A&P (customer 1897 of sparse dimension #3) sold colas (member 12 of sparse dimension #1) in New York (member 200 of sparse dimension #2.)”

Matrix explosion

The three defining characteristics of a DB2 OLAP Server array are:

1. The number of dimensions
2. The number of members within each dimension
3. The hierarchical relationship of the members within each dimension

Data explosion can occur across each characteristic individually and concurrently having a combined (that is Cartesian) impact. For example, if we increase sparse dimension #1 to include 5000 members, the number of potential intersection points increases from $2.3 * 10^{13}$ to $3.3 * 10^{15}$! In similar fashion, adding a completely new dimension will explode the number of potential intersections points. We can do both at once, adding more members to an existing dimension at the same time as we add a completely new dimension to the database.

Database Tuning Requirements

There are four major areas to concentrate on when performance tuning a DB2 OLAP database:

How to handle the characteristics of the dimensionality and embed business logic

How to implement Member Tags

How to handle outline/database consolidation and business formulae

How to determine optimal dense/sparse settings

All of these revolve around the single most important DB2 OLAP construct: the database outline.

In a DB2 OLAP Server environment the database outline is the database schema. We have said it is a place where the business model and the data model are the same and are represented graphically. The storage characteristics of the Outline are no less important to a multidimensional model than are the storage characteristics of the schema to a relational one. A relational schema tuned for OLTP or query processing will pay very different attention to the schema design and the associated storage structures to achieve the desired performance characteristics.

Goal of the database tuning:

The goal of optimally configuring sparse-dense settings for a database tuning is twofold:

1. Create as few blocks as possible.
2. Create blocks that are as densely populated as possible.

Dense dimensions are implemented to reflect the *density* of the data set and **sparse** dimensions are implemented to reflect, or reduce the effect of, the *sparseness* of the data set. We write **reduce** because it is not realistic to expect to be able to eliminate sparseness from a DB2 OLAP Server matrix. If the dense nature of a data set is **not** contained within the block (a dense dimension is tagged sparse) then an explosion of the overall number of data blocks that the configuration generates will result. Conversely, large and empty data blocks indicate that sparseness has not been effectively eliminated from the data set.

Generic outline tuning considerations

The main considerations are:

1. **How many dimensions are there?** Implementers are limited by the total number of dimensions that can be modeled as well as by the hardware configuration on which the model is being developed.
2. **How large are they?** The total number of members ultimately determines the sparseness of the data set.

3. **How deep are they?** Database performance characteristics will also vary according to the depth of the hierarchies of the dimensions. Designers must be able to identify and adjust database configurations according to the demands of the specific (practical) database being developed.

Sparse/dense settings might have to be altered to accommodate specific client requirements, for example, to support member calculation or query retrieval requirements.

Database configurations will be optimal not necessarily according to the best sparse/dense configuration but according to requirements specific to the model at hand.

Databases that are incrementally updated across time almost by definition preclude the possibility of tagging the time dimension dense, even though it properly adheres to the density of the dataset.

Batch Calculation and Generic data storage considerations

The batch calc process for DB2 OLAP Server databases optimizes run-time performance by making the vast majority of data set values persistent. The cycle of database build and refresh will regularly include time for loading data *as well as* time to replenish derived and aggregation values along the dimensional hierarchies. The period of time required to refresh data is commonly referred to as the **batch window**. The batch calculation process will have significant disk storage implications. To help defray the cost of auxiliary storage and I/O, *the objective of the database designer is to implement the database configuration that generates the least number of blocks that has the highest density.*

Member Tags and Dynamic Calculations

DB2 OLAP Server enables users to tag members as dynamic calculations. Members so tagged will have their calculation removed from the batch calculation process. They will be dynamically calculated for the user at query retrieval, or run time. Positive effects of implementing dynamic calculation member tags can be:

1. to reduce batch calculation window
2. to reduce data block size
3. to reduce overall database size

Member tags are not only implemented for storage implications. Tags also enable users to take embed business logic coded in the DB2 OLAP Server calculation engine within the Outline. The DB2 OLAP Server database has been coded to ‘understand’ a certain amount of business logic.

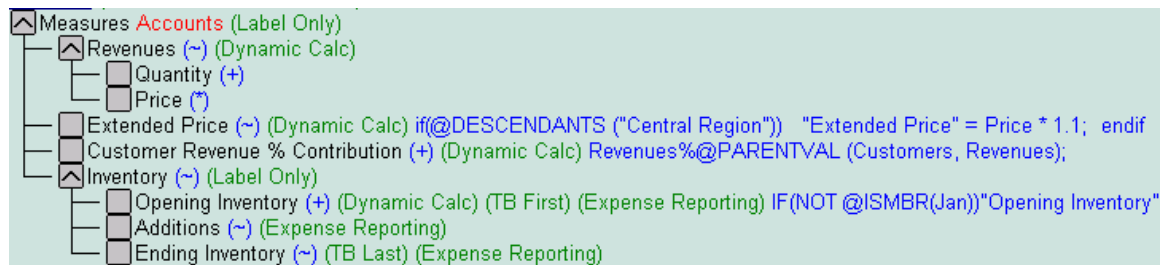
For example, “Time Balancing” is the term DB2 OLAP Server uses to describe business quantities that do not aggregate over time. Having 5 headcount in January, 6 in February, and 10 in March, does not mean we have 21 headcount for Quarter 1. The same is true for inventory items, and (*alas*) for balance sheet items such as your checkbook. There is a critical business need to represent these metrics as desired at upper levels of time. They cannot be left to the standard engine aggregation.

In DB2 OLAP Server, you click a button in the outline to make a quantity **TB First** (said “Time Balance First”), or “**TB Last**”, or “**TB Average**”. This will cause the database engine to report correctly at upper levels of time the **first** period value, the **last** period value, or the **average** period value.

There are also ways to enable the database engine to calculate values according to business logic that either includes or skips values that are missing from the database. To implement, for example, the business-relevant difference between the average revenue for *all products*, and the average revenue for all products *sold*.

DB2 OLAP Server enables this business logic to be implemented by the database engine. Implementation is simple, intuitive, and a mouse-click away and works the same across all levels of time.

The following Outline fragment illustrates a small sample of Tagging capabilities of DB2 OLAP Server:



Note first that the **Measures** dimension itself has been tagged as the **Accounts** dimension. This is a *dimension* tag that enables the implementation of other member tags like, for example, **Expense Reporting**. The **Dynamic Calc** tag indicates that member calculations are performed at query time rather than during batch window calculations. The **Label Only** tag causes DB2 OLAP Server to treat the member as a navigation point, and DB2 OLAP Server does not reserve any storage space for *numeric* data for these members. The **TB First** tag implements the time balancing functionality discussed above, and the **Expense Reporting** tag implements balance sheet expense account functionality. And finally note that the **Opening Inventory** member has no less than **three** member tags as well as a **member formula** associated with it.

The small Outline fragment helps illustrate why business users find it easy to use to build business models. The Outline is, in the final analysis, a user-friendly object for Business Analysts. Users can be allowed (or disallowed) to view Outline contents directly in order to understand the definition of metrics. The same user-friendly functionality cannot be said to apply to other OLAP products.

Summary

In this chapter we conveyed the notion that DB2 OLAP Server is a MOLAP tool that enables Business Analysts to **steer** the Business. We defined OLAP as essential and

integral to BI/DW initiatives. Its place is complementary to Data Mining and Relational technologies within the BI/DW.

We provided a high-level architecture for implementing OLAP across the enterprise and argued for the strategic importance of using the fulfillment OLAP requirements to help IT prototype the data warehouse. Implementations of this type necessitate the coordination and cooperation of IT with Business User personnel.

A discussion of basic DB2 OLAP Server functionality and storage implications was also presented and the reader can find below references to additional information and resources.

Bibliography And Additional Reading

Kaplan, Robert S., Norton, David P., *The Scorecard: Translating Strategy into Action*, Harvard Business School Press, 1996.

Inmon, William H., *Building the Operational Data Store*, Wiley, John & Sons, 1999

Kimball, Ralph, *The Data Warehouse Toolkit*, Wiley, John & Sons, February 1996

Kimball, Ralph, *The Data Warehouse Lifecycle Toolkit*, Wiley, John & Sons, August 1998.

Tillman, George, *A Practical Guide to Logical Data Modeling*

Appendix

Hyperion Solutions

In 1998 Arbor Software merged with Hyperion Software to form Hyperion Solutions.

The functionality of DB2 OLAP Server is derived directly from Hyperion Solutions' Essbase OLAP Server. Taking version release timing into consideration, Hyperion Solution's Essbase and IBM's DB2 OLAP are identical products.

Hyperion Solutions and IBM continue to dominate the OLAP market today. Together they retain over 30% of the OLAP market share.

History

In its first release, DB2 OLAP provided only relational storage on DB2 Universal Database and no multidimensional storage. In 10/1998, version V1.0.1 was extended to new UNIX platforms including SUN SOLARIS and HP/UX.

In 09/1999 IBM delivered DB2 OLAP Server V1.1, which provides both relational and multidimensional storage and was based on Essbase V5.0.2. Version 1.1 has been available on the OS/390 platform since 02/2000 and on the AS/400 platform since 06/2000.

IBM has delivered DB2 OLAP V7.1 based on Essbase Server Version 6.0 on the UNIX and Intel platforms since 06/2000; on AS/400 since 12/2000; and has delivered it on the OS/390 platform in 11/2000.

Additional Licensing

In addition to DB2 OLAP Server, IBM licenses from Hyperion Solutions Hyperion Analyzer and resells it as IBM Analyser. This product delivers ad hoc or EIS OLAP reports to power users or the executive suite. Also, IBM OEM's Hyperion Integration Server as OLAP Integration Server (see more below).

Functions

To support calculations DB2 OLAP Server has over 200 built-in functions in nine groupings and includes the ability to create user-defined functions:

[Allocation](#) functions allocate values that are input at the parent level.

[Boolean](#) functions.

[Date & Time](#) function converts date strings to numbers for use in calculations.

[Forecasting](#) functions manipulate data for the purpose of calculating future values.

[Mathematical](#) functions return calculated values based on specified parameters.

[Member Set](#) functions return a list of members.

[Range and Financial](#) functions.

[Relationship](#) functions look up data within a database during a calculation.

[Statistical](#) functions calculate classical statistical metrics such as standard deviation.

[User Defined Functions](#) enabling users to create their own functions.

Scorecard

IT personnel who want to understand better what managers are thinking when they ask for metrics for the data warehouse could look at *The Scorecard: Translating Strategy into Action* by Robert S. Kaplan, David P. Norton ISBN 0875846513. This book is driving much current thinking of what to measure and how to measure it in piloting modern enterprises.

Three Report Constructs

In the early 70's the Dutch computer scientist Edsger W. Dijkstra published "Notes on Structured Programming" in C.A.R. Hoare, Ed., *Structured Programming*, Academic Press, 1972. He proposed that a correctly written program performs **only three logical constructs**: sequence, decision, and repetition.

We have noticed a pleasing but wholly coincidental parallel to OLAP operations. Once the numbers are computed, there are only three things that an OLAP engine needs to support in order to permit making any OLAP report. **Drilling** to move quickly up and down dimension hierarchies, **keep only** to expand or contract report parameters and **pivot** to transform rows into columns and vice versa.

While the parallelism is coincidental, it does for OLAP what Dijkstra did for programming and brings discipline to an otherwise chaotic arena.

It can be very confusing for an IT specialist trying to understand all the things that users want to do with information. Knowing that there are only three things they need to do to make any OLAP report (drill, keep and pivot) at the very least makes the design sessions psychologically easier.

Gartner Article

Readers might want to refer to the Gartner Tactical Guideline TG-14-2780. The authors propose that the creation of a "BI competency center" to "[h]elp the IS organization realize that users will need multiple BI technologies to meet their varied analytical needs, while getting users to support the IS organization's need to provide a platform that will support changing user requirements."