DB2 OLAP Server Theory and Practices

Matrix management and storage structures in depth

Advanced OLAP design practices

Practical implementation experiences

Corinne Baragoin
Jorge Bercianos
Janez Komel
Gary Robinson
Richard Sawa
Erik Schuinder

Redbooks

ibm.com/redbooks

SG24-6138-00

International Technical Support Organization

DB2 OLAP Server Theory and Practices

April 2001

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special notices” on page 241.

First Edition (April 2001)

This edition applies to IBM DB2 OLAP Server Version 7, Release 1, on every platform on which it is available.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This IBM Redbook explores useful design procedures and shares significant implementation experiences for DB2 OLAP server and Essbase. The non-exhaustive list of procedures spans from design tips to successful tuning. The implementation and deployment experiences are described through customer and partner interviews.

This redbook focuses on IBM DB2 OLAP Server Version 7 — with the multidimensional storage option — and Hyperion Essbase OLAP Server Version 6.0. We use the term DB2 OLAP throughout the book to refer to both products.

Table 1 below lists the version and releases of IBM DB2 OLAP Server and Hyperion Essbase OLAP Server.

Hyperion	IBM
Essbase Server V5.0	DB2 OLAP Server V1.0
Essbase Server V5.0.2	DB2 OLAP Server V1.1
Essbase Server V6.0	DB2 OLAP Server V7.1

Table 1. Hyperion and IBM: versions and releases

In 02/1998, IBM delivered its new analytical software: DB2 OLAP Server V1.0 based on Arbor Essbase V5.0 on Windows NT,OS/2, AIX platforms. 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 announced it on the OS/390 platform in 11/2000.

In 12/2000, IBM introduced DB2 OLAP Server Analyzer. This is based on Hyperion Analyzer Version 7.1 (called Analyzer throughout this redbook), an easy-to-use OLAP client for Windows and the Web.

Designers and technical people with an understanding of DB2 OLAP fundamentals will benefit from reading this redbook to design and implement future DB2 OLAP solutions.

How this book was written

This redbook was originally conceived as a “best practices” book for DB2 OLAP. An exhaustive knowledge of best practices in any discipline rarely rests within the grasp of one or two people so this book represents the contribution of many individuals. It really is the result of a collaboration of the highest order.

Some of the contributors are anonymous. They are an extremely important group of people whose work in OLAP principles and practices was ground breaking. Their practices are today so foundational that it is often overlooked that these were ever “developed” in the first place. But it simply remains beyond our ability to present these people individually. There are, however, four groups of identifiable contributors (IBM, Hyperion Solutions, Business Partners and DB2 OLAP Clients) that deserve mention.

The team that wrote this redbook

From IBM there was the DB2 OLAP redbook residency team working at the International Technical Support Organization San Jose Center. Its task was to manage this redbook project, research and test many of the recommended practices, and finally produce the document you have before you today.

From Hyperion Solutions there were people who assisted in every aspect of this project with perhaps the exception of final document preparation and delivery.

Business Partners of both IBM and Hyperion Solutions made contributions as well. A considerable portion of this text was prepared and submitted by such individuals.

Finally, in the form of IBM conducted interviews, OLAP clients made contributions by being willing to divulge practices and procedures that currently reflect their own DB2 OLAP production environments.

Some of the individuals contributed by:

- Writing sections of the text: **Corinne Baragoin, Gary Robinson, William Sterling, Richard Sawa, Cheryl A. McCormick, Paul Turner, Debra McRae, William Hodges, Bob Wallace,** and **Dave Nolby.**

- Performing research or answering interviews and providing contents to be incorporated within these pages: **Daniel DeKimpe, Christopher Dzekian, Jim Burnham, Steward Teed, Sujata Shah, Leah Wheelan, Rich Semetulskis, Alan Farkas, George Trudel, Joe Scovell, Jacques Chenot, Mark Rich, Steve Beier,** and **Aster Hupkes.**
- Researching, documenting, and testing: **Jorge Bercianos, Janez Komel,** and **Erik Schuinder.**

As a result, you will not find a single voice speaking in the pages that follow. Hopefully this unbalanced style will be more than compensated for by substance and content.

The IBM team:

Corinne Baragoin is a Business Intelligence specialist at the International Technical Support Organization, San Jose Center. Before joining the ITSO, she worked as Technical Presales Support in IBM France and has been working on data warehouse environments since 1996.

Jorge Bercianos is a Data Mining specialist in IBM Uruguay. He has been involved in data mining and OLAP integrations for the last two years.

Janez Komel is a Software Technical Sales Manager in IBM Slovenia. He has 12 years of experience in database management and data warehouse implementation.

Gary Robinson is a Senior Software Engineer with IBM Corporation, based at Silicon Valley Lab in the USA. Gary has over 12 years experience in Business Intelligence and Decision Support. He is a member of the DB2 OLAP Server development team, working with customers and partners deploying DB2 OLAP Server.

Erik Schuinder is an IT specialist on Business Intelligence in IBM Global Services in The Netherlands. He has been involved as OLAP designer/consultant for implementing OLAP solutions on large data warehouses.

William Sterling is a worldwide DB2 OLAP technical specialist in IBM.

Daniel DeKimpe is a Software Advisory Engineer in IBM Silicon Valley Lab.

The Hyperion Solutions team:

Richard Sawa is the Hyperion Solutions Channel Sales Technology Manager for IBM and ShowCase. He has 11 years of experience in relational and multidimensional database technologies and specializes in Essbase database tuning and optimization.

Cheryl A. McCormick is a Senior Principal Consultant in the Data Integration Integration Services group.

Paul Turner is a Senior Technical Manager in Essbase Business Development.

Debra McRae is a Regional Practice Manager in the eCRM Services group.

Christopher Dzekian is a Director of Business Development and Product Management in the Hyperion Analysis Tools division.

Jim Burnham is a Principal Software Engineer in the Information Technology Services group.

Steward Teed is a Senior Development Manager in the Information Technology Services group.

Sujata Shah is a Senior Product Assurance Engineer in the Customer Product Assurance Lab Essbase Technologies.

The Business Partners team:

William Hodges is a partner at Tech-OLAP, Inc., an OLAP consulting firm with offices in Montreal, QC, Canada and Stamford, CT, U.S.A. and a professor of MIS on leave from the University of Quebec at Montreal.

Bob Wallace and **Dave Nolby** are working in LumenSoft Corporation, in St. Paul, Minnesota which is dedicated to delivering value added OLAP solutions through Consulting Services, Product Development and Training. Bob, a certified OLAP professional, has 9 years of experience implementing Business Intelligence solutions and is currently working with the Product Development Department. Dave has over 10 years of Business Intelligence experience and is a certified OLAP professional and trainer.

Leah Wheelan is the President of Beacon Analytics, Inc., a consulting firm that specializes in OLAP implementation.

Rich Semetulskis and **Alan Farkas** are working in ThinkFast Consulting. Alan Farkas, a Senior OLAP Consultant, has over six years of Essbase experience and has been providing OLAP solutions to clients since 1988.

Customers and Customer representatives through IBM Global Services:

George Trudel is Director of Information Services at Scrip Pharmacy Solutions and he has 25 years of business and technology experience. He started working with multi-dimensional databases in 1990 and was the first business user to beta test DB2OLAP. Scrip Pharmacy Solutions summarizes prescription transaction data as part of its knowledge delivery to its customers.

Joe Scovell and **Jacques Chenot** at DST Systems Inc., of Kansas City, a Transfer Agent in the Mutual Fund Industry. Joe Scovell is a Client Services Manager. He has 13 years experience working with many diverse Mutual Fund clients. His area of expertise centers around providing clients value added services that enable them to intelligently mine their data.

Mark Rich has been a financial and business analyst for 16 years in IBM. In 1994 he was the technical lead for IBM's World Wide Consolidation Accounting project, implementation of Hyperion Enterprise. In 1998 he was brought on board to lead IBM's World Wide Planning System, currently based on IBM DB2 OLAP on IBM RS6000 technology.

Steve Beier is working as a Senior IT specialist at IBM Global Services. He currently works for IBM Storage Division on an enterprise information system and his primary responsibilities include project architecture, technical lead and "renaissance" support.

Aster Hupkes is working as an IT specialist in the Business Intelligence team of IBM Global Services in the Netherlands. She has over two years experience in designing and implementing OLAP solutions at several customers in the Netherlands.

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 263 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introducing OLAP

In this chapter we will attempt to communicate to you the importance of recognizing the distinct place in an information systems infrastructure that an On-Line Analytical Processing (OLAP) server can occupy. Our discussion transitions quickly to a description of the DB2 OLAP array storage structures and how they are the means by which database designers come to practical terms with the realities of (sparse) matrix management. Finally we end by suggesting strategies that will assist designers to refine their skills at managing the data explosion that can occur when implementing a DB2 OLAP array. We believe that these three topics are intimately related.

1.1 Best practice begins with good theory

This is essentially a practices book for OLAP implementers. A book on recommended practices of any kind, it seems, should not require a lengthy explanation or argument in favor of the corresponding technology. Rather, it should concentrate on providing the reader with guidelines as to how to best use the technology, how to avoid misinterpreting its functionality, how to prioritize implementation goals to get a high cost/benefit ratio, and so on. In other words, the author(s) should be able to assume that the reader has had the opportunity to witness beneficial results or, better yet, has sufficient understanding of the underlying fundamental principles to be able to visualize the raw power of this technology.

Yet, there is enough evidence within the IT profession to conclude that OLAP is still a poorly understood IT solution. When it is implemented, this is often done because it happens to be available, it is new, it seems to work and/or it is sufficiently easy to apply in some form or fashion. Rarely is it implemented because of a deep understanding that it is in fact the most appropriate solution.

There is also sufficient evidence in the commercial world to conclude that OLAP has many definitions. OLAP has reached such a status as a buzzword, term or label that it is desirable and effective to use it and to that end it is whatever its proponent decides it should be. Here is a case in point: OLAP is a very good technology for the execution of financial data consolidations, better and easier to use for such a purpose than SQL, for example. But a tool or product that performs consolidations is not, therefore, an OLAP tool or product, no matter how efficiently and elegantly it performs those consolidations.

Why is this so? Simply because OLAP is, and is expected to be, many other things as well. In fact, consolidations are among the very minimum that an OLAP tool ought to be able to do and are, for the most part, a feature taken for granted. An OLAP specialist asked to implement consolidations using an OLAP product would hardly consider the work interesting unless it involved other more significant challenges.

It is for reasons of this nature that we have concluded that a practices book on OLAP must begin by establishing in a clear and concise way the technology's *raison d'etre*. It appears that a deeper understanding of the principles that originated the technology is required to help implementers make better use of it. To some people, OLAP looks too much like a "gimmicky technology", an optional end-user's tool, and not enough like a necessary component of a serious, reliable, robust, efficient, effective, widely focused organizational information systems infrastructure. DB2 OLAP is not an end-user tool; *it is a matrix-oriented application server*. In fact, it might even be considered the only OLAP server worthy of the name on the market. What it can do, no other type of data server can do, and customers need to understand this. It may be too much to say that every company should have one, but this is not an outrageous exaggeration. There are many very valid reasons why a company or an individual should not own the best technology available.

OLAP's role in an organization's IT infrastructure

As indicated by the words used to construct its acronym ("on-line," "analytic" and processing"), OLAP's role in an organization is to provide easy interactive access to analytic resources for the purpose of supporting a management or decision-making role. Historically, decision support theory has recognized two types of analytic resources: data (static information) and models (dynamic information) (Steven Alter *"Decision Support Systems: Current Practice and Continuing Challenges"*, Addison-Wesley (1999)). OLAP, fits the definition perfectly: it is an information server that responds to queries about raw and derived data. Raw data is data loaded into its database; derived data may be, for example, a summary of that data, a forecast derived from the same or calculated independently, a what-if scenario, and so forth, any of them computed by formulae or programs resident within the same data store (not by external programs). In summary, OLAP is a technology that combines better than others access and computational ease.

It so happens that in an overwhelming majority of cases organizing data as a matrix or collection of matrices best fulfills this role. Consequently OLAP, a term which in itself does not imply any specific data structures, has become synonymous with “multidimensional database” and produced two major challenges:

1. Managing matrix size and sparsity
2. Modeling business environment (or other) behaviors as matrix-oriented formulae and scripts.

These two challenges imply a third:

3. Developing the ability to mentally visualize multidimensional data arrangements and data computations.

1.2 The role of theory in OLAP practice

Theory (the explanation of relevant facts and relationships embedded in a phenomenon) can certainly help us to most successfully face these challenges. On the one hand, no OLAP theory has to date been formally or completely proposed. Yet, a handful of vary basic principles is sufficient to begin to recognize OLAP’s potential as a critical component in any organizational informational infrastructure. This is all we really need. They are simple but powerful; they should not be ignored.

First, as already suggested in the above paragraphs, OLAP is supported by decision support theory. Secondly, as will be later explained, it is an application of fundamental computer science principles related to the combination of algorithms and data structures to produce computational power. Thirdly, also later discussed, it is a most suitable environment for the implementation of business models that apply system dynamic principles, the only discipline, according to some experts (for example, Peter Senge in his book *“The Fifth Discipline: The Art and Practice of the Learming Organization*”, Currency/Doubleday (1991)), that can actually help us to deal with complex business environments.

1.2.1 Decision support theory

The art of building decision support systems is the art of building computer-based solutions to undefined problems. The secret to building these solutions is to apply a multi-layered development approach whereby generic functionalities are embedded in the lower layers and can be purchased so that an end solution can be constructed and modified rapidly just by adding the upper layers. Decision theory identifies three generic functionalities: data management, model management and interface management (as mentioned by Ralph Sprague and Eric Carlson in *“Building Effective Decision Support Systems”*, Prentice-Hall, Inc. (1982)). A development environment that provides these three functionalities is known in the decision support literature as a DSS generator. DB2 OLAP is a DSS generator.

1.2.2 Fundamental computer science principles

OLAP's credibility as a fundamentally different and more appropriate database technology was unfortunately tainted by the way its theory was initially presented to the public by Codd, who has written "*Providing OLAP to User-Analysts: An IT Mandate*". This book can be found on `www.essbase.com`). And this theory was not even complete, nor was it a real theory, but rather, just a list of requirements. Other authors have attempted to be more thorough (for example, Erik Thomsen who has written "*OLAP Solutions: Building Multidimensional Information System*", John Wiley & Sons, Inc. (1997)). However, the bottom line is that we still do not have a theory and possibly never will, at least not at the level of abstraction that Erik Thomsen suggests it should be developed.

But if we recognize OLAP not only as a data server (as is a RDBMS) but also as a modeling tool, an application development environment, a prototyping environment, then its theory should not be constructed along the same lines of reasoning as the relational database model. For example, if we look elsewhere, in the classic computer science literature we find that DB2 OLAP needs not to be justified by set theory (as did Codd with respect to relational databases) or the significance of missing values (as does Erik Thomsen) but more simply by the principle that computer performance is based on the interaction of two components: data structures and algorithms.

Probably the most emphatic statement in this regard is Niklaus Wirth's book titled: "*Algorithms + Data Structures = Programs*", Prentice Hall,Inc. (1976). In this book, the author lists three fundamental data structures: the record, the matrix and the bitmap. DB2 OLAP is a perfect example of an effort to apply the best available data structure to the problem at hand. It can store data both as records and as matrices, it uses matrix-oriented algorithms to perform simple computations as well as to build complex models and it uses bitmaps to speed up access to the data used and/or generated by these models.

1.2.3 Model management and system dynamics

It would be difficult to imagine the explosive evolution of computing from the late 70's until today if we eliminated the electronic spreadsheet from the computing scene. Many have attributed the success of the personal computer to Visi-Calc and then Lotus 1-2-3, which made visible and accessible to the masses a computer’s ability to perform a series of computations in order to obtain numerical results. Today, most spreadsheet users know that not only can we program a spreadsheet to compute a profit, but we can also request that the spreadsheet estimate the revenue necessary to produce a desired profit, all from formulae that were not designed to compute revenue. We call this “goal seeking.”

The point to be made here is we can perform goal seeking and thus turn premises into conclusions, because a collection of formulae is a model of an aspect of reality that can be manipulated in order to better understand that reality, just as a miniature airplane can be used to test the aerodynamic characteristics of a real airplane.In order to be effectively used, models need easy access, easy manipulation, easy development, and easy storage. DB2 OLAP provides all of these features.

1.3 Position dependence versus position independence

One of the contributors to the theoretical elegance of the relational database model is the fact that position in a “relation” (a table) is irrelevant. Where a record appears in a table and where a field appears within a record has no bearing on what a query against the table will produce in terms of results. Position independence is one of the features that helped relational database technology replace older technologies (hierarchical databases such as IMS, and CODASYL databases), in spite of the obvious performance advantages of the older technologies, which used pointers, and to this extent, they were position-dependent. Position independence makes a database more flexible and a better choice for ad-hoc querying.

In the early years of programming, one would occasionally compare and evaluate programming languages by asking the question: “Does the language include arrays?” and the question: “What is the maximum number of dimensions allowed in these arrays?” APL, an early computer language, included matrix operations as defined in linear algebra. For example, the linear algebra expression A=B*C, where A, B, C are matrices, implies a multi-term operation of the type Aij = somme Bi*Cj, in other words, data structures and native computational algorithms well beyond what standard computer languages can perform “out of the box”.

But computer users whose problems were not suited for or required data representation in the form of a matrix, or computer users who could not begin to conceptualize their computational problems as matrices, could not have taken advantage of APL, no matter how sophisticated APL is as a programming language and even more so was, comparatively, in the early years of computing.

Not all computational problems require, lend themselves to, or can benefit from, a matrix-oriented treatment. Those that do, on the other hand, still face the problem of managing data storage efficiently. Matrices are by their very nature space allocators. They are analogous to beehive-type mailbox ensembles where, once labeled, a box will reserve, in other words eliminate the flexible use of, a physical space, whether its beneficiary ever receives mail or not. People who have mailboxes of this kind at their office can locate their mail very quickly simply by going directly to their mailbox. These same people would have a slightly more difficult time locating their mail before it is placed in the boxes, even if all the envelopes are properly addressed and even if the envelopes have been sorted in alphabetical order and neatly stacked inside a general-purpose box. But this general-purpose box would by its nature occupy less space than the beehive mailboxes.

Thus, what in one situation is boasted as an advantage in another could be suffered as a disadvantage. Finding data in a table can be burdensome and time consuming even when indexes are available. Finding data in a matrix when its basic coordinates are known is very simple. You jump directly to the corresponding cell address. It may be said that matrices are to tables what RAM is to tapes. In RAM, as in matrices, when you know what you are looking for, you know where exactly it may be found.

1.4 A different data model at a very fundamental level

There is a very fundamental difference between matrices and tables: relational tables (RDBMS) model the world as a collection of interrelated entities that happen to have attributes. The attribute cannot exist independent of its entity. Some attributes may become the entity identifiers but this is not required. In conclusion, tables (and RDBMS) are entity-oriented.

On the other hand, matrices are data-oriented. In the specific case of OLAP they are numeric-data-oriented. When we store a number in memory using a programming language we use variables. For example, to place the number 546 in memory we can state VAR1=546. To store four numbers in memory we can use four variables or we can use a one-dimensional matrix (see Table 2): VAR(1)=546, VAR(2)=765, VAR(3)=762 and VAR(4)=951.

Table 2. One-dimensional matrix

	A	B	C	D
1	546	765	762	951
2				
3				

We can store the same four numbers in a two-dimensional matrix (see Table 3): VAR(1,1)=546, VAR(1,2)=765, VAR(2,1)=762 and VAR(2,2)=951. And we can use a spreadsheet the same way (a spreadsheet is a matrix).

Table 3. Two-dimensional matrix

	A	B	C	D
1	546	765		
2	762	951		

How or why would we select one or the other arrangement? We would select the arrangement based on the advantages that position would grant us in each case.

For example, the second arrangement would allow us to easily compute various subtotals as in Table 4.

Table 4. Computing subtotals

	A	B	C	D
1	546	765	1311	
2	762	951	1713	
3	1308	1716	3024	

Therefore: position becomes useful in terms of the operations it facilitates. More than this, if we change the row and column labels (see Table 5), position may be used to recognize, acknowledge, document the actual meaning of these numbers.

Table 5. Naming rows and columns

	Tables	Chairs	Both products
New York	546	765	1311
Boston	762	951	1713
Both Cities	1308	1716	3024

We purposely did not write “give meaning.” The data, if it came from a real life situation, already had its meaning, we just had not yet recognized it. This is an important point. OLAP does not add dimensionality to data, it recognizes it and brings it to the fore. In this sense, OLAP is one of at least two different ways available to represent dimensionality, another one being the star schema design proposed by Ralph Kimball in “*The Data Warehouse Toolkit*“, John Wiley&Sons, Inc. (1996).

Having given meaningful names to columns and rows, formulae can now be expressed as, for example, Chairs = Tables * 4, indicating that each table comes with four chairs (see Table 6).

Table 6. Adding formulae

	Tables	Chairs	Both products
New York	546	2184	2730
Boston	762	3048	3810
Both Cities	1308	5232	6540

And if the above is a forecast rather than a report of actual sales, and if sales in Boston are typically one half of the sales in New York, then we can build a ***Forecasting Model*** where stating one number (the number of tables we expect to sell in New York) will give us the sales for tables and chairs in both New York and Boston (see Table 7).

Table 7. Forecasting model

	Tables	Chairs	Both products
New York	**546**	2184	2730
Boston	273	1092	1365
Both Cities	819	3276	4095

In summary, the formulae contained in the above mini-model are:

```
Chairs = Tables * 4;
```

```
Boston = New York * 0.5;
Both Cities = New York + Boston;
Both Products = Tables + Chairs;
```

Notice that contrary to the way formulae are stored in spreadsheets, in DB2 OLAP they are stored once only and applied wherever they apply. Notice also that the above model has only one datum. The rest of the numbers are all derived data. (And if we were to add products and cities and establish similar logical relationships we could end up with a model that computes sales for all the products in all the cities based on one number! The purpose of the example, though, is to show the advantages of applying and extending a well-known programming paradigm: the spreadsheet).

Position can be further exploited through the use of relative and absolute addresses as in Table 8.

Table 8. Using relative and absolute addresses

	A	**B**	**C**	**D**	**E**	**F**
1		1	2	3	4	5
2	1	**1**	**2**	**3**	**4**	**5**
3	2	**2**	**4**	**6**	**8**	**10**
4	3	**3**	**6**	**9**	**12**	**15**
5	4	**4**	**8**	**12**	**16**	**20**

In Table 8, the bold numbers are calculated by formulae generated by entering the formula =$A2*$B1 into cell B2 and then copying this formula into cells B2:F5. We only had to write the formula once and the computer was able to make the necessary adjustments to correctly apply different versions of the same formula to other cells. Without getting into the details of how it is done we can already point out that DB2 OLAP, exploits position similarly with the added feature that cell formulae do not have to be restated for every cell. They are adjusted dynamically as they are computed, and only one generic version of the formula is actually stored. And the dynamic adjustment is not limited to a two-dimensional space but can be extended to as many dimensions as may be present in the DB2 OLAP database.

1.5 The issue of data redundancy

One of the difficulties to overcome when proposing the implementation of an OLAP solution is to justify the duplication of data storage mechanisms as well as of that of the data contained therein. The data warehousing movement is evidence of the improvements in data availability that can be obtained from restating data in a different format. And Inmon (who has written *“Building the Data Warehouse”*, John Wiley&Sons, Inc. (1996)) points out very clearly that there is no redundancy in data warehouse and datamart contents because it is restated data, data at a different level of detail than their sources.

1.6 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 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.

Consider the business example of a two dimensional array having three members of a Measures dimension called sales, cost of sales, and profit (sales minus cost of sales), recorded daily across Time (the second dimension) for one year. It is usually intuitive that as time progresses (that is every day) we are able to measure sales and the cost of sales and generate our profit. This model would have 1,095 intersection points (365 days by 3 measures — see Figure 1). The description that Day and Measures are densely populated with respect to each other is quite readily comprehended. For a given day, if you have a sales number, you probably also have cost of sales information, and you can therefore compute the profit measure. Unless there is extreme season fluctuation, sales information exists for most days, and thus the matrix is likely to be “densely” populated for Measures across Time.

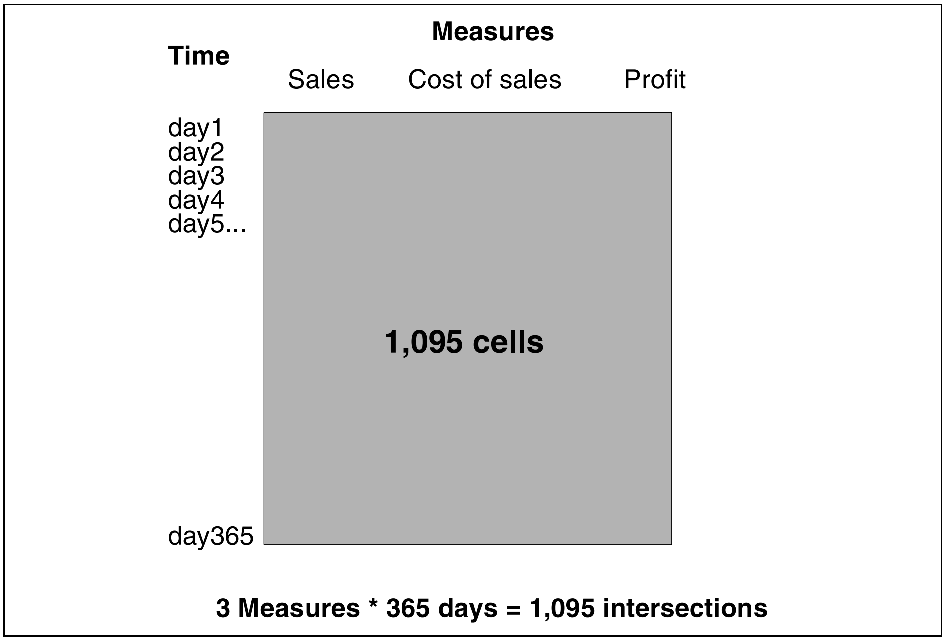


Figure 1. Number of intersection points — example

The notion of sparseness enters the discussion as we add more dimensions. For example, consider adding the dimensions of Customer (numbering 100,000) and Product (numbering 50) to this model. It is generally easy to conceive that there will be numerous intersections across these dimensions that will not contain data. All customers purchase not all products every day of the year.

The number of intersection points has increased as the Cartesian product of all members from each dimension. Our model has just increased from 1095 cells to 5,475,000,000 (365*3*50*100,000) cells.

Transactional relational databases handle data sparseness by only storing data as the result of a sales event. They would record actual sales. The Structured Query Language (SQL) can then be used to interrogate and analyze the data to answer questions like which products have the fewest sales across time. This answer is derived programmatically. Relational databases generally do not have tables that store the fact that (many) products were *not* purchased by (many) customers. An array does.

By declaration, a matrix *would* house an intersection point for every possible combination of sales and cost of sales across time for every product and every customer. Most of these cells do not contain data. The answer to which products have the fewest sales across time is contained in the database and is revealed simply by retrieving the intersection points in question, and examining their contents.

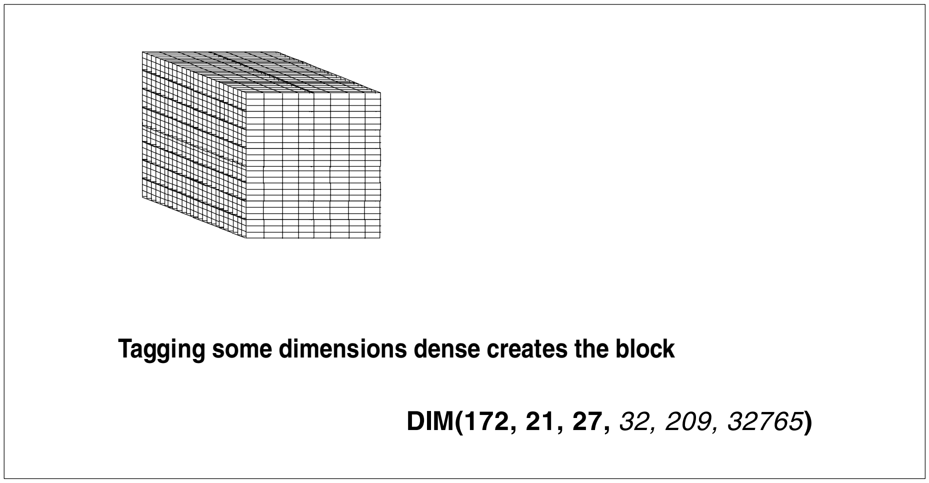
It is apparently easy to understand the concept of sparseness. Unfortunately it is apparently not as easy to conclude that the DB2 OLAP storage structures are the central design mechanism that can be used to minimize the impact of sparseness. The two storage structures of the data block and the index were indeed developed by the designers of Essbase to contend with the reality of sparseness.

1.7 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 (see Figure 2).



Tagging some dimensions dense creates the block

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

Figure 2. Dense dimensions form the block

Every data block that is created in the database has an identical structure. In this 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. By enabling these two definitions of array dimensions, the creators of Arbor Essbase enabled the matrix to be 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.

1.7.1 Block creation explored

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

- **dense** dimension #1 containing 172 members
- **dense** dimension #2 containing 21 members
- **dense** dimension #3 containing 27 members
- sparse dimension #1 containing 32 members
- sparse dimension #2 containing 209 members
- sparse dimension #3 containing 32,765 members

Which data blocks are actually created depends upon unique sparse combinations that contain data (see Figure 3).

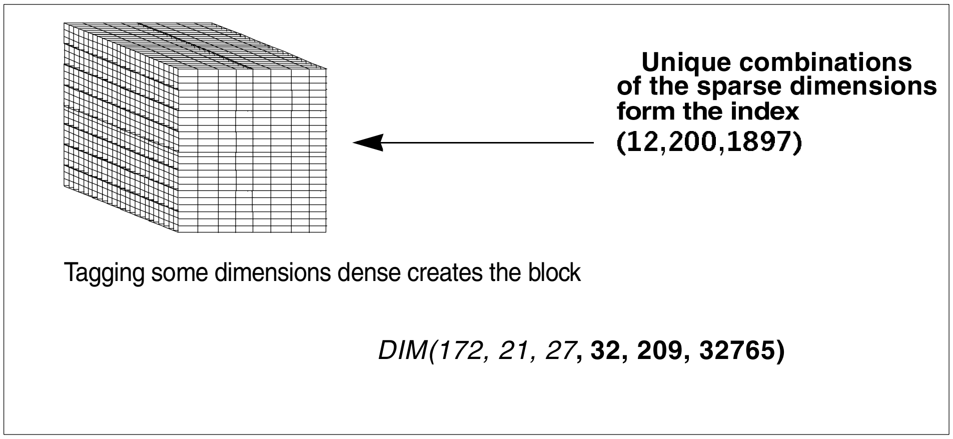

Unique combinations
of the sparse dimensions
form the index
(12,200,1897)
Tagging some dimensions dense creates the block
DIM(172, 21, 27, **32, 209, 32765)**

Figure 3. Sparse dimensions form the index

In this 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 interpret the above snapshot Figure 3 to something like:

```
colas (member 12 of sparse dimension #1) were sold in
New York (member 200 of sparse dimension #2) by
A&P (Customer 1897 of sparse dimension #3).
```

In its most simple form, DB2 OLAP lets organizations represent their business elements as members of an array, and store numeric values at intersection points within the array.

1.7.2 Matrix explosion

The three defining characteristics of a DB2 OLAP 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 21,370,660,375,680 to 3,339,165,683,700,000! In similar fashion, adding a completely new dimension will explode the number of potential intersection points and we can do both at once; adding more members to one dimension while adding a completely new dimension. The discussion that follows simulates a DB2 OLAP implementation and uses sparse dimensions as examples and assumes some familiarity with basic DB2 OLAP concept.

Consider, for example, the differences between DIM (172, 21, 27, 32, 209, 32765) and DIM (172, 21, 27, **5000**, 209, 32765, **450**.)

The first rule of design in OLAP modeling is to minimize the number of dimensions in a database design.

While minimizing dimensionality is our first guiding principle, there is a dynamic tension between it and the reporting needs of the enterprise. The point is that the possibility of block explosion needs to be bounded in such a way as to produce intersections that are actually used for business analysis.

The third way a matrix can explode is not as apparent as the first two. It is the effect of increasing the complexity of intra-dimensional hierarchical relationships. In array DIM (172, 21, 27, 5000, 209, 32765, 450) we will take the new sparse dimension #4 (with 450 members) as our case study.

Keeping in mind the other sparse dimensions, let us assume that the dimension is entirely flat and does not contain any hierarchical relationship between members except at the parent of all members. The dimension would look graphically something like Figure 4.

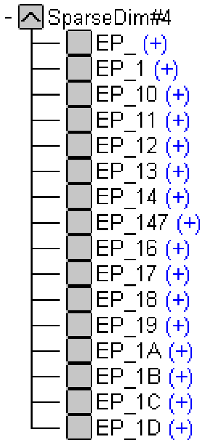


Figure 4. Dimension without any hierarchy

The dimension would have a total of 449 members aggregating to the sole parent, giving us 450 members in total. Consider the implications in terms of data point creation between the dimension in Figure 4 and the dimension in Figure 5.

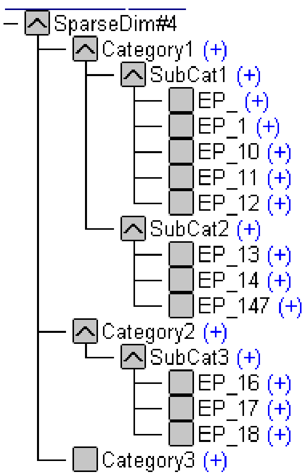


Figure 5. Dimension with embedded hierarchy

In this declaration we have hierarchies that groups of members are organized along, and this has important storage implications for our database.

If we have business transactions for member EP_11 but do not have any business transactions along EP_12 in the flat sparse dimension #4, no EP_12 blocks are created. The impact of the sparseness of the data has been effectively handled (eliminated) by the design.

However, when we introduce hierarchy to the dimension, we are declaring the existence of upper data points when data exists only for one descendant. In the hierarchical version of SparseDim#4 a transaction at EP_11 will generate data at SubCat 1, Category 1, at the SparseDim#4 parent member, *and data block creation will result as a Cartesian product across each member of every other sparse dimension for which data exists!* This effect has enormous cross-dimensional implications.

The point here is not that flat dimensions are good, and hierarchical ones are bad. On the contrary, hierarchies accurately reflect the reality of many business organizations and a matrix is extremely efficient at modeling these relationships. The point is rather that hierarchies add complexity to the array and are a component that contributes to data explosion in multidimensional databases. Their impact must be well understood.

1.7.3 The data block and index revisited

The data block reserves one cell for every intersection point (the Cartesian product) for all of the members from each dimension that is tagged dense. Sparse member combinations that contain data drive data block creation. So, if we were to implement a database design that tagged as dense the dimensions of Time, Measures and Products (assuming that, after all, we do sell all of our products every day), only those customers that actually make purchases will cause a data block to be created.

Thinking a little more deeply about the above implementation should reveal that because the data block contains products, and every customer does not purchase every product (even during the course of the entire year), our block will contain mostly empty cells. By removing the Product dimension from the block (that is tagging Product as a sparse dimension) reduces our block size. Now the real combinations of specific customers who in fact purchase specific products will create data blocks.

The new database design has eliminated from the storage structure intersection points that do not contain data. We have, in essence, implemented a design that creates the fewest data blocks that have the highest data density. It is this empirical relationship or ratio between data block contents (the block density) and the number of blocks created that can be used to detect optimal database configurations.

1.8 Other multidimensional design guidelines

There are a small number of design guidelines that follow from the previous discussion of sparse matrix handling in DB2 OLAP Server. These guidelines were originally articulated in the original Hyperion Solutions' OLAP Fundamentals course, and we are expanding on them considerably here, hopefully providing a compelling theoretical context.

1.8.1 Avoid inter-dimensional irrelevance

An irrelevant dimension is a dimension where the majority of intersections of its members have no business meaning in combination with the majority of members of other dimensions. Consider the example where a Profit and Loss database contains the expense metric of employee salaries. The option to include the employee master file as a dimension in the model presents itself as a way to derive the total salary expense metric. Indeed, the ability to produce detailed employee analytic within this model is seductive. However, the impact of doing this is profound. It forces the creation of intersections where the vast majority of contains no relevant business data (for example, rent expense by employee is a near meaningless metric.)

It might be tempting to think that if these irrelevant dimensions are tagged as sparse, then the effect of data explosion will be effectively offset by the database configuration. But this is not true, and their presence will still drive intersection creation in the OLAP model.

Examine again the characteristics of the dimension SparseDim#4 in Figure 5 on page 16 and consider this dimension in combination with another sparse dimension as configured in Figure 6:

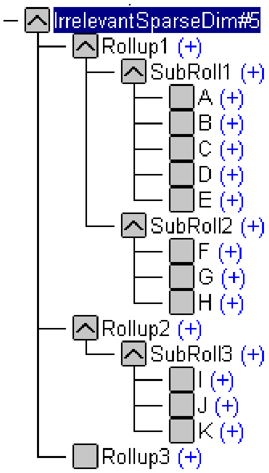


Figure 6. Irrelevant sparse dimension

Let us assume that Intersection point EP_11 only has business meaning across the members from the otherwise irrelevant dimension. Every member from the irrelevant dimension (that are members A, B, C, etc.) will be loaded in combination with EP_11. The effect within the DB2 OLAP storage structure will be to generate a storage point for upper members across both of these dimensions. For example, an intersection point at SubCat 1->A will be generated by this array. This data point contains absolutely no business value but clearly has hardware and software infrastructure overhead.

Inter-dimensional irrelevance leads to inefficiencies in production causing unnecessary data block creation. These useless blocks need to be indexed and calculated and serve ultimately only to reduce analytic efficiencies both within batch processing and for user at query time. The presence of inter-dimensional irrelevance is usually accompanied by the existence of measures that make sense in terms of some but not all dimensions in the model. In the final analysis, inter-dimensional irrelevance infers the existence of more than one model.

1.8.2 Combine dimensions where possible

The method for determining when to combine dimensions is to investigate dimensions combinations to locate where one-to-one correspondence exists between members of different dimensions. One to many relationships are modeled using separate dimensions. One-to-one correlations between dimensions require a single dimension be implemented.

Consider the 2 database outlines in Figure 7.

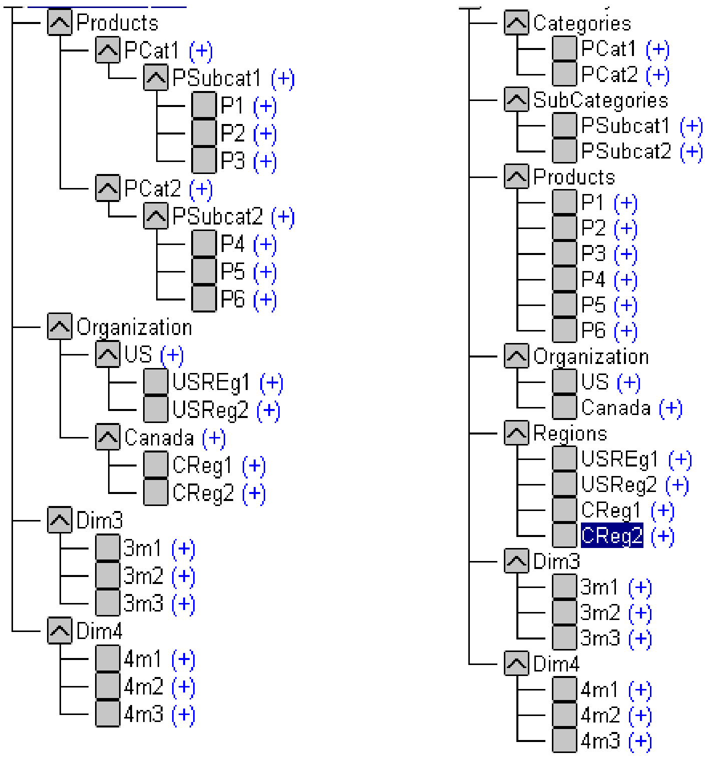


Figure 7. Combining dimensions

The outline on the left in Figure 7 has its one-to-one members arranged hierarchically. The outline on the right arranges the hierarchies as independent dimensions.

The total number of intersection points generated by left outline is 1,232. The total number of intersection points generated by the right outline is 15,120!

Data point explosion occurs because the array schema will generate intersection points containing absolutely no business value (for example, the intersection Products->Psubcat2.) These irrelevant intersections do have significant storage structure overhead. So, simply by combining dimensions (it2) we implement a more efficient model.

1.9 Summary

In this chapter we presented an overview of the major components of the DB2 OLAP multidimensional storage structures. The chapter began by discussing the inherent importance of the matrix-oriented application server as a viable information systems database. It concluded by suggesting very general rules that can be used in the practical task of modeling business processes for implementation on a multidimensional database management system (MDDBMS).

Implementing a business model using DB2 OLAP Server is not a science and can only be called an art to the extent any human endeavor can be performed in more or less artful fashion. Success of design is directly related, though it is not too much an exaggeration to say exclusively related, to the depth of the designer’s understanding of the DB2 OLAP storage structures, though this is not more true of an MDDBMS than of an RDBMS.

Every component of the database implementation owes the degree of its efficiency to how it impacts the data blocks and index entries. Even the efficient use of the 120+ functions of DB2 OLAP, which are not discussed within this text, is determined by how they are implemented in respect of these structures.

As readers page through the various sections outlining practices and theory, they are asked to keep this observation in mind to see if it does not, for them also, become a self-evident truth.

Chapter 2. OLAP model development checklist

This chapter presents an OLAP model development checklist. The original intention of this chapter was to provide a comprehensive outline of steps required to implement a DB2 OLAP model.

2.1 Introduction

What you will find below is a distillation of an OLAP design methodology that has been developed and polished over at least 1/2 decade by experienced DB2 OLAP Server and Hyperion Essbase Server practitioners - primarily William Sterling. The outline can really be considered comprehensive from the perspective of database creation and design. In itself, this outline is a significant contribution to DB2 OLAP implementation practices.

This chapter does not include any significant discussion of topics presented. Most topics are attended by no commentary at all. This has been done on purpose. Wherever possible we include references to contents and practices contained elsewhere in this redbook. In Table 9, we have provided readers with an outline they can use as an OLAP checklist for quick reference. Further discussion at this point would only serve to obscure the main points that the brevity of an outline conveys almost at a glance.

In referencing contents and practices we have provided an OLAP cookbook of sorts. Readers are able to refer to relevant sections to learn what to do, what order to do it in, sometimes how to do it and whenever possible why the practice is done. In the eyes of the authors, it would not be considered at all a bad strategy to read this section only after digesting the rest of the contents contained in the other chapters and appendices.

We suggest that you photocopy the checklist below and use it as a project checklist. The last column is left blank to let you validate the project progression step-by-step. An abstract of this checklist is also provided in Appendix F, “OLAP model development short checklist” on page 237.

2.2 The OLAP checklist

Table 9 provides an OLAP checklist to be used as a project checklist to help readers to develop OLAP models.

Table 9. Project checklist

Activity	Reference?	Check?
1. Plan for OLAP		
a. Recognize an OLAP opportunity.		
• OLAP is aggregated analytic.	Chapter 1., “Introducing OLAP” on page 1	
• We should not cube the warehouse	Appendix A., “OLAP datamart design approaches” on page 155	
b. Evaluate the OLAP opportunity.	Chapter 3., “Project management for OLAP” on page 29	
• Define subject matter databases		
• Technical people have to understand the business matter and the business issues that the OLAP application will solve		
2. Begin high level modeling of the OLAP database structure		
a. Look at current and desired reporting requirements		
• Evaluate the dimensions required		
• Do it for each report by asking the following questions: “Who, What, When, Where, How”	4.1, “Prototyping an outline” on page 39	

Activity	Reference?	Check?
• From this, begin sketching dimensions - Avoid designing on paper - Use the application Manager as a Rapid Application Design tool (RAD) - Create the OLAP outlines in the development stage to be able to save it, to display it to users and to modify it quickly - Check that the model has the right scope • Combine dimensions 2 by 2 and check if the analysis of dimension A by dimension B makes sense	4.1, “Prototyping an outline” on page 39	
b. Do modeling development in a group		
• Use Application Manager as a Rapid Application Design tool	4.1, “Prototyping an outline” on page 39	
• Project the outline on screen for very fast progress	4.1, “Prototyping an outline” on page 39	
c. Evaluate the data using SQL to know data volumes involved and to check the relevance of data	4.1.2.3, “Use SQL wherever possible” on page 42	
• If data is relational, use SQL - To scope dimension hierarchies: do a “select count distinct” across each dimension” - To check the cardinality of the fact table or the table that will be the source for the measures dimension		
• If data is not relational, make it so		
- Perhaps using desktop database software		
• Check for nulls in data used for building dimensions		
d. Estimate development software hardware required		

Activity	Reference?	Check?
• Prefer a dedicated platform for development, similar to production if possible	3.3.2, “Choosing an environment” on page 32 and 5.2, “Interview results” on page 98	
• The main configuration issues concern mainly memory and disk space	4.6, “Performance tuning: the buffers” on page 73 and 4.8.4.1, “Estimating database sizing” on page 83	
3. Build a prototype for size testing		
a. Build the dimensions		
• Type in the dimension names manually, using directly Application Manager		
• Check if partitioning is required	4.10.1, “Partitioning tips and strategies” on page 86	
• Set preliminary dense/sparse - Use Application Manager Database/Settings dialog box suggestion for sparse/dense as a first cut	4.3.1, “The concept of sparseness: dimension tags” on page 45	
• Always use esscmd.exe scripts to build large dimensions		
b. Let the outline do the work		
• Use Member Tags - Use Label only. - Use Dynamic Calc. - Put Formulae in data blocks	4.4.2, “Considerations on use of member tags” on page 52 and 4.5, “Considerations on database calculation” on page 66	
c. Load test data		
• Sort the data by sparse dimensions	4.10.3, “Data load optimization” on page 91	

Activity	Reference?	Check?
• If SQL sources, make transformations in SQL. - Prepare upstream the data by doing transformations in SQL rather than the rules file - SQL is self-documenting - rules files are not	5.2, "Interview results" on page 98	
d. Calculating and tuning		
• Calculating - Move the formulae into the outline and do a CALC ALL. - Configure your database so that it does a two pass calculation in one pass. - FIX and IF to focus calculations. - Use Intelligent calculation.	4.5, "Considerations on database calculation" on page 66 and 4.5.3, "Focusing calculations" on page 70 and 4.9, "Intelligent calculation" on page 84	
• Tuning - Use compression - Set up caches	4.7, "Data compression" on page 77 and 4.6, "Performance tuning: the buffers" on page 73	
e. Test sizing, calculation performance and query as you go		
• Evaluate the model - Check the model for size Use the SET MSGS ONLY method Use the sparse/dense methodology - Evaluate the model for calculation time Use the SET MSGS ONLY method Use the sparse/dense methodology	4.8, "Using SET MSG ONLY" on page 80 and 4.4.4, "Sparse/dense methodology" on page 60	

Activity	Reference?	Check?
• Evaluate the query response time - Check the application log for the spreadsheet retrieval factor - Verification by users and acceptance by users - Adjust sparse/dense dimensions to accommodate user data requests	4.10.2, "The application log" on page 89	
f. Refine dense/sparse to optimize the outline		
• Use the sparse/dense methodology • Use the configuration Wizard	4.4.4, "Sparse/dense methodology" on page 60	
g. Adjust dimensions and members based on prototype size testing		
4. Build and load the final model		
a. User acceptance testing: the data		
• Use a spreadsheet tool - It is important to ensure that the OLAP database satisfies the goals of the user requirements. Do the calculations give them the information they need? Are they satisfied with consolidation times? Does the database work for them?	3.4, "Acceptance testing" on page 36	
b. User acceptance testing: the access user tool		
• Writing reports - Create standard user interface if required - If you plan to provide predefined reports to users, design the reports layout and run the reports using the final end-user tool		
c. Building and setting up a security model	4.10.4, "Building a security model" on page 93	
• Check the security model with users		
d. Train users		

Activity	Reference?	Check?
• Users should feel comfortable in accessing data with the reporting tool chosen		
5. Migrate to production		
a. Setting up system administration		
• Develop and write the production procedures: - To maintain the outline - To refresh/update the data - To validate the data - To backup and restore the OLAP database - To maintain the software level and set up the patches - To trap errors	5.2, “Interview results” on page 98	

Chapter 3. Project management for OLAP

This chapter highlights the project management issues for OLAP solutions. It is based on Bob Wallace and Dave Nolby's experience who both are working as OLAP specialists for LumenSoft Corporation, dedicated to delivering value added OLAP solutions through consulting services, product development and training. LumenSoft's services division delivers superior project management, OLAP solutions and a full complement of custom OLAP courseware.

3.1 Importance of project management for OLAP

The key to any successful OLAP implementation is good project management discipline. By exercising proven and tried project management methodologies, your OLAP implementation can enjoy on-time and on-budget results that meet the most challenging business requirements.

A rigorous project management practice is essential to the success of any IT project, and OLAP implementations are no different. OLAP implementations are typically more closely tied to and driven by the business side than traditional systems development projects.

More often than not, this means that there must be a great deal of collaboration between business side staff and technical staff. Technical staff involved in the development of an OLAP project often find themselves spending a great deal of time interacting with business side staff extracting complicated business rules and nuances. This type of interaction is quite different than traditional systems development and certainly requires a unique skill set, one with sound business and technical experience.

In fact, the deep and inherent link between business and technical staff that is usually required to implement an OLAP solution often means that sound project management techniques take a very pronounced role in ensuring successful completion of the project. This means that there must be a conscious and significant effort applied to traditional project management practices such as requirements gathering, system design, change control, implementation and ultimately project acceptance.

If we look at a traditional technology project, we can agree that the same common project stages exist. These basic stages may resemble the following:

1. Initiation Phase
2. Construction Phase

3. Implementation Phase

Each of these stages has an objective and purpose, and a good project manager will plan and schedule the completion of each of these stages, knowing that their careful execution will increase his chances of success, and simplify the undertaking.

An OLAP implementation does not differ significantly from these traditional stages, an experienced project manager may be able to “squeak-by” without properly understanding OLAP implementation techniques. However, to improve a given project manager’s ability for success and to help minimize unnecessary risk to the project, several factors should be incorporated into the traditional project plan. But before we discuss these factors, let’s first overview the components for developing a complete OLAP system.

In the following sections, we will discuss these topics:

- OLAP project issues
- Key points to implement an OLAP solution
- User acceptance issues

3.2 OLAP project issues

There are two primary concerns when conducting an OLAP project; these are the design and development of the OLAP database, and the design and development of the end-user application.

3.2.1 OLAP database

The life-cycle for developing a successful OLAP database is cyclical in nature. Many of the tasks, with experience, evolve into more of an art than a science.

Since the OLAP database is the foundation from which all other project activities are based upon, it can easily be described as the most important, and the most difficult, activity of the project. Listed below are the major areas of the OLAP database development process:

1. Modeling of the OLAP database structure
2. Outline construction and the applying of attributes
3. Formulae and calculation construction
4. Data loading and verification
5. Optimization of database performance

3.2.2 OLAP end-user application

Because the OLAP database is esteemed as the most important activity of the project, the end-user application may be trivialized in its importance to the over-all OLAP implementation. This is unfortunately too often the case. As a result, many well-intentioned and well-designed OLAP databases fail miserably because they do not meet the expectations and demands of the end-user community.

The entire OLAP implementation can be likened to a fine automobile. The OLAP database is the engine, and the body with all of its accessories is the end-user application. Just as with the automobile, the database (engine) is the most important component of the auto that helps it perform its designed task. But the engine cannot accomplish anything alone — it requires carefully designed and implemented components that work together with the engine to accomplish a given task.The driver (end-user) will be able to accomplish their task with relative ease and efficiency if the components are well designed.

The considerations and implementations of a successful end-user application are too numerous and varied to effectively discuss in this chapter. You may find some additional information regarding end-user development topics in the sections below.

3.2.3 OLAP project team structure

In that OLAP implementations are highly user driven, it is important that roles be identified. These generally include the following roles and responsibilities described in Table 10.

Table 10. Project team roles

Team title	Roles and responsibilities
Project Sponsor	Sign-off responsibility for project deliverable, change requests and final acceptance. Project support of goals and objectives.
User Representative	Will provide system functionality and requirement information.
Project Manager	Responsible for support of project team and direction of project initiatives.
Team Leader	Responsible for resource and schedule management. Will guide and direct project activities at the task level.
Contractor	Will aid in the project initiatives. Will assist in direction and development of these activities.

3.3 Implementing an OLAP project

Planning for an OLAP project involves project assessment and feasibility studies, choosing an environment, planning and analysis, and sponsorship.

3.3.1 Project assessment and feasibility

Certainly every project manager would like to ascertain feasibility before beginning a project. Can the project succeed, or is it destined for failure? If it can succeed, can it be completed in a reasonable amount of time and expense?

Assessing an OLAP project abides by standard management practices and principles that help assess practicality and worth for the understood needs. The difficult part is to complete an accurate and credible assessment. A common mistake in many OLAP implementations is to over analyze through forums, discussions, surveys, and reports. The result is inaccurate or unclear results, divided team members, and frustrated management all at the expense of time and money.

A recommended plan for conducting a successful and effective assessment is to get OLAP and Project Management experience and to start with a Proof of Concept (POC) project or prototype.

Proof of Concept: A vital element is to identify a small, manageable portion of the business that will provide adequate context so the business side can easily extrapolate the concept to the larger picture. In many cases, the multidimensional concept is so far removed from these people, that the biggest hurdle is often achieving a basic understanding of what can be accomplished. Multidimensional solutions are frequently so much more powerful and flexible than traditional reporting and analysis systems, that it can be a challenge to get the business side to fully grasp the possibilities. One typical outcome of a successful proof of concept is the “jaw on the table” effect when the true power of the solution finally sinks in.

3.3.2 Choosing an environment

The DB2 OLAP environment can vary widely, depending on factors such as company standards relating to hardware and operating systems and the actual system requirements of the DB2 OLAP project.

Ideally, a DB2 OLAP environment will contain at least two and possibly three tiers of servers: development, test, and production. It is not advisable to operate with only one server for both development and production. Operating in a single server environment can not only hamper the developer's efforts, it also introduces risk to any production applications running on the server.

Environment decisions also depend on whether or not a DB2 OLAP server already exists. In either case, care should be taken to follow recommended sizing techniques for both disk and memory requirements (see "*OLAP Database Administrator's Guide*"). In a situation with an existing server, a complete impact assessment must be done taking into account the current applications on the server. Questions such as the following must be asked:

- What are the approximate disk space requirements?
- What are the approximate memory requirements?
- How many users will potentially be added to the system?
- Will there be a need for additional ports?

Situations that require the purchase of a new server take a slightly different approach. Obviously, consideration must be given to any corporate standards in place regarding operating system and hardware. Aside from that, the process of assessing potential disk and memory requirements must proceed as in the previous case. Additionally, care should be taken to predict future growth of the system. Is there expected growth in the number of applications loaded on the server or the amount of data loaded into existing applications? A recognition of the growth factor therefore frequently leads to purchasing a server with more capacity than the immediate application requires or at a minimum one that will reasonably support hardware upgrades in the future to accommodate expected growth.

3.3.3 Planning and analysis (requirements and design)

In this section we will consider the planning and analysis phases of a project.

3.3.3.1 Guiding requirements for the project

The requirements phase is a critical component of the system development cycle that will ultimately determine the success and business side acceptance of the project. The goal of requirements gathering is to thoroughly define what the ultimate system users must have to satisfy their business needs. This means that defining not only what will be in the project, but also what will not be in the project. An outcome of a successful requirements gathering phase is that all parties involved will have realistic expectations of what the system will ultimately deliver. Some common OLAP development requirements may be:

- How many end users must have 24x7 access to the system?
- Will users have the ability to directly update data on the system? This leads to additional security requirements and possible client application requirements.
- Will some users be restricted as to what they may view? This again leads to security requirements.
- Is the definition of the cube relatively static or does it change frequently? That is, do the business requirements dictate frequent additions, deletions or movements of members in one or more of the dimensions in the definition of the cube? This leads to ongoing maintenance requirements that must be considered.
- Do the business requirements indicate that one or more dimensions in the cube will be defined with many members and with many levels? This may require that partitioning options be investigated.
- Are the business reasons for the project basically retrieval and analysis of information or is there also a process that is being implemented (such as budgeting). If a process is being implemented, the use of VBA applications or other client tools to control the process (as automated running of calc scripts, controlled input screens, and so forth) should be investigated.
- Is the reporting from the project completely ad-hoc or are there some fixed reporting format requirements? If fixed report formats are being used, VBA applications or other client tools should be investigated to control consistency.

Some keys to success also include formalizing the requirements gathering approach rather than loosely gathering ideas, ensuring that all key system users are represented and iteratively refining and verifying the requirements during the stages of design and development.

3.3.3.2 Model the OLAP database (outline)

This is the step where detailed user requirements for structuring the OLAP database are developed to satisfy the reporting and analysis requirements.

Modeling involves the creation of a prototype outline that models the dimensionality of the user requirements. This outline should be kept to as few dimensions and members as possible but must adequately reflect the dimensionality and the depth within the dimensions being communicated by the user.

Next the database will be populated with test data. Once test data is available, use a client access tool such as the spreadsheet add-in and work with the user to ensure that the model will provide the reporting and analysis information required.

Cycle through these steps with the user until a satisfactory model is obtained.

3.3.3.3 Develop the OLAP database (outline)

This step is where the actual outline structure for the database cube is developed.

The outline will be based on the structure defined from the modeling above but will include actual members with the necessary attributes and formulae. Some members may be entered manually and others can be loaded from files of information available from other systems.

If dimension members are being created from files from other systems, dimension building load rules will be created at this stage.

3.3.3.4 Calculate strategy

If a full consolidation calculation of the data is all that is required, no calc scripts have to be developed, as the default calc performs this function. However, if computational requirements exist that are not satisfied by the formulae in the outline or consolidation requirements exist which are best satisfied by not calculating the full database, calc scripts must be developed.

In such instances it is recommended that scripts embrace modular concepts rather than building large complex monolithic scripts even though this may result in multiple calc scripts. There are situations in process implementations where rather than creating a series of conditional calc scripts, an API application can be written to simplify the process by dynamically generating a script, sending it to the server and executing the script.

3.3.3.5 Data loading

Data can be loaded directly into the database via the spreadsheet add-in or from files of data being sent from other systems.

If data is being loaded from files of data being sent from other systems, the file structures for the data interface must be defined and the data load rules will be developed during this step.

If data is directly entered through the spreadsheet add-in, no load rules are required. However, to ensure integrity, increased security requirements may be required.

3.3.3.6 Reporting

Ad-hoc analysis and reporting can now be accomplished through the use of the spreadsheet add-in or other client reporting applications. Generally users should acquire training in the use of the spreadsheet add-in to become proficient in this tool.

If common report format sets are to be defined and used, VBA applications or other client tools should be investigated to facilitate consistency and useability.

Tools like Analyzer are also available to create presentation level information from the base (see D.2, “What about DB2 OLAP Server Analyzer?” on page 218).

3.3.3.7 Client applications

In situations where consistency, control and ease of use are a goal, the creation of VBA applications, API applications or other client applications should be considered. This is even more important if the project is developed to manage a process.

3.3.3.8 Performance

Performance tuning is an empirical process that usually is best addressed after a representative set of data has been loaded and used. This cyclical procedure that involves the tuning of cache settings, the use of calculation scripts that selectively calculate sections of the base, the use of dynamic calculation member storage setting, and so forth.

3.4 Acceptance testing

The newly developed OLAP application must be accepted by the user community prior to deployment. This involves validation of the data, verification of the application design and acceptance of the client reporting tool.

Acceptance testing is an important part of the project’s quality assurance cycle, and should be driven by a formal test plan that covers quality assurance of the data, the application and the client tool. The test plan should be developed and executed by representative members of the OLAP development team and the business users.

Any test plan should be well thought out and cover a wide range of the data and functionality provided by the application. This comes down to developing a list of actions that can be taken within the application, noting the expected result and having the tester record the actual result. For example, a new reporting system built on the Analyzer platform may provide a portfolio of ten financial reports to end users. A test plan item may be as simple as noted in Table 11.

Table 11. Test plan item example

Step	Action	Expected result	Actual result	Pass /fail
14	Execute Financial Summary Report	System generates and displays Financial Summary Report with accurate numbers that balance to provided control numbers	Report generated successfully; all financial data matched control data	P

Special consideration should be given to data validation, which often becomes part of the regular cube building cycle. Both input and derived data should be validated. If the OLAP application is replacing or augmenting existing reports, then validation can be as simple as building the same reports from the OLAP application and comparing the two. If reports are not available, then a tool or program will be required which extracts data from the source systems and independently calculates aggregates for comparison. If the extraction, calculation and comparisons can be automated, then this step can easily be implemented as part of the batch process which builds the cubes, resulting in quality assurance as an ongoing part of the OLAP process.

Chapter 4. Tuning, good design practices, and useful tips

This chapter is a collection of tuning techniques, design practices, and useful tips. It is not intended to be an exhaustive chronological presentation of DB2 OLAP application development practices for new developers. The tips and techniques described here are augmented by many database parameters (like database isolation level, aggregate missing values.) Please refer to the use of these parameters as described in DB2 OLAP documentation and the System Administration course offered by IBM and Hyperion Solutions.

Although the current chapter is not structured for reading from beginning to end, it *can* be. In doing so, new developers will become acquainted with its contents and this familiarity will enable them more effectively to refer back to the details and methodologies contained as issues arise for them in the development process. The ability to use this chapter as a sort of cookbook reference is enhanced by reading it page by page. It is also hoped that experienced developers will also derive benefit from reading the contents contained below.

4.1 Prototyping an outline

Prototyping is an excellent next step once the decision has been made to implement a DB2 OLAP solution. The chief objective is to arrive at an accurate picture of the analytic and infrastructure requirements early.

When prototyping an outline, the most important steps to address are:

- Training and team building
- High level modeling

4.1.1 Training and team building

The OLAP team should be a complementary blending of technical information systems and business personnel. To exclude either side from this process is to court disaster, and really, the first step in prototyping after selecting team members is to invest in DB2 OLAP training for both groups.

Technical personnel will devote time investigating the software and hardware infrastructure requirements of DB2 OLAP. They will be keenly aware of client-server and Web architectures as well as issues of sparse matrix management as they pertain to DB2 OLAP Server storage structures. The training process will enable them to become aware of issues for modeling business processes in a DB2 OLAP outline.

The essential role of business personnel will be to define the end-user analytic and reporting requirements *and* ensure that the specific business logic of the application being developed is appropriately reflected in the DB2 OLAP outline. Perhaps the latter is their most important function.

It is extremely important for business personnel to have devoted time investigating the technical issues of embedding business logic within a DB2 OLAP outline. The training process will enable them to become aware of the software and hardware infrastructure issues for modeling a business process in DB2 OLAP.

Ultimately the two groupings of individuals on the OLAP development team will acquire skill sets that complement and complete one another. The classic DB2 OLAP training class is composed of Information Systems professionals and the business professionals that they support.

4.1.2 High level modeling

One way to begin high-level OLAP modeling is to assemble the above team to work through a more or less complete listing of reporting and analytic requirements. By systematically de-constructing current sample reports, the team will be able quickly to develop a list of reporting attributes each of which could eventually be reflected as an OLAP dimension and its hierarchy. All of the generic OLAP rules of thumb mentioned in Chapter 1, “Introducing OLAP” on page 1, should be adhered to, for example, by looking for one-to-one correlations between dimensions, minimizing the number of dimensions, and so forth.

4.1.2.1 Create a report and analytic grid

Generate a list of example report templates that itemizes all of the dimensions and business metrics.

The example grids shown in Table 12 illustrate the results.

First, notice that Reason Closed, Status and Assigned_To originally identified as important dimensionality do not appear on any reports. They can be eliminated. Second, observe that the Process Analysis dimension never appears on the same report the Product and Received_Via dimensions. From this chart we can conclude that one 8 dimensional and one 9 dimensional model will produce all reporting requirements. We have averted the desire to deliver analytics using a single 13 dimensional model by eliminating unused dimensions and clustering used dimensions in two more efficient configurations.

Table 12. Analytic grid

	USER REPORTS										
		1	2	3	4	5	6	7	8	9	10
LIST OF DIMENSIONS	Case Type	x			x			x	x	x	x
	Call Center	x		x	x	x	x	x	x		
	Time	x		x	x	x	x	x			
	Time of Day				x						
	Product	x		x	x			x			
	Customer	x	x	x	x		x	x			
	Severity	x		x	x		x	x	x	x	x
	Reason Closed										
	Received Via	x		x				x			
	Status										
	Assigned to										
	Process Analysis		x			x	x				x

By examining the dimension combinations, you can much more easily determine how many DB2 OLAP database models will need to be developed to deliver the analytic requirements. Such a grid will be useful when determining whether a database scale strategy (like partitioning) might be necessary.

4.1.2.2 Design as a team: Use the Application Manager as a RAD

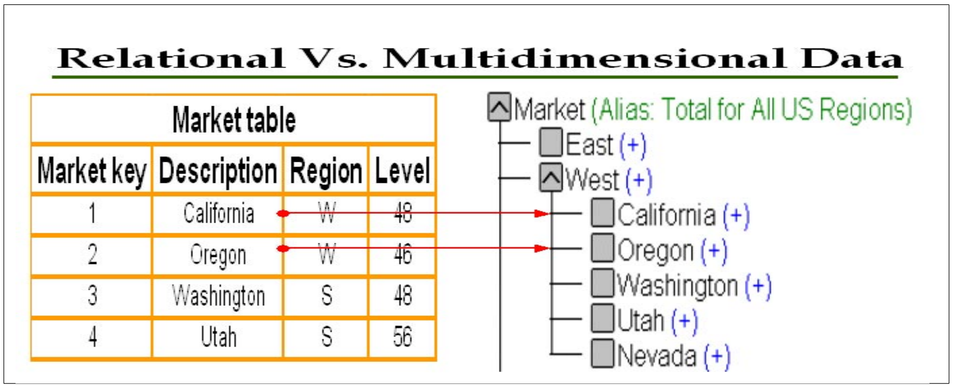
Use the Application Manager Outliner as a rapid application development (RAD) tool in design sessions. Do not design on paper, since this leads to headings and categories and metaoutlines that are not OLAP outlines, and defers thinking about the actual data. The net result of using the Outliner is that you will have created DB2 OLAP prototype outlines in this early part of the development cycle.

Project the outline in progress from a desktop computer onto a viewing screen for all to see. Use the spreadsheet add-in to demonstrate OLAP functionality (pivot, drill-down,...) of the outline as you build it. You don’t need to load data to the model to do this. Witnessing the pivot and drill functionality via the spreadsheet add-in can streamline the design decision-making process.

4.1.2.3 Use SQL wherever possible

It is extremely useful to be able to access a relational database that contains data relevant to the dimensionality being discussed during the design session. Data elements of a relational database become part of the metadata in a multidimensional database as illustrated by Figure 8.

It is not a co-incidence that the star schema is the most effective relational model schema to support OLAP functionality. A star schema is precisely a *relational* attempt to reflect the *multidimensional* characteristics of data.



Relational Vs. Multidimensional Data

Market table

Market key	Description	Region	Level
1	California	W	48
2	Oregon	W	46
3	Washington	S	48
4	Utah	S	56

Figure 8. Relational versus multidimensional

Having access to source data in a relational database means that as each dimension is brought forward for consideration, the development team can quickly generate SQL queries that will give an indication of dimension membership and hierarchies. You can learn how clean the source data is by looking for duplicate entries, null values and so forth. Moreover, by retrieving counts of members, the team can even begin to discover generic but useful information about the potential size of OLAP models. Expectation management begins in this process.

If resources permit, consider converting the source data that will be used to perform the first database builds to a relational database, specifically build a relational star schema.

One of the most significant benefit will be the practical ability to investigate detail-level data drill-through. Having access to the source data in detail will also give you the ability to rebuild and transform source data efficiently as new and unexpected requirements arise.

In the final analysis, being in full control of source data significantly truncates the OLAP development process. An important bonus is that the development team can see first-hand the benefits of an architecture that relates OLAP multidimensional databases to a relational star schema datamart.

4.2 Database tuning introduction

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

1. How to handle the characteristics of the dimensionality and embed business logic
2. How to implement Member Tags
3. How to handle outline/database consolidation and business formulae
4. 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 environment the database outline is the database schema. 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. A discussion of the foundation of tuning the outline will always include a discussion of the foundation concepts of the DB2 OLAP storage structures.

4.3 Basic matrix database concepts

The distinction between relational and multidimensional storage structures is relevant only to the extent that it helps demonstrate what DB2 OLAP storage structures are (see Figure 9):

1. *Values* in relational tables can be understood to become part of the metadata *structure* of multidimensional databases.
2. Matrix data stores eliminate the need for a data manipulation language (like SQL) for data retrieval. Access to data is provided directly through cell addresses.
3. *In its most simple form, DB2 OLAP lets organizations represent their business elements as members of an array, and store numeric values at intersection points within the array.*

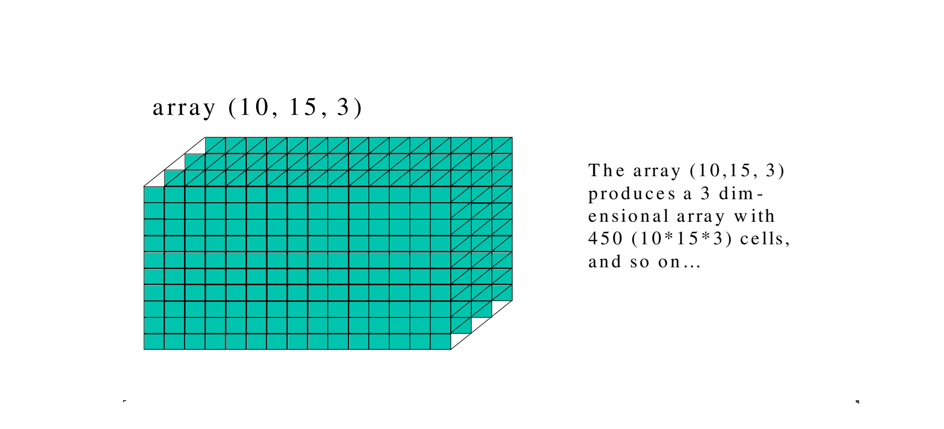

array (10, 15, 3)
The array (10,15, 3) produces a 3 dimensional array with 450 (10*15*3) cells, and so on...

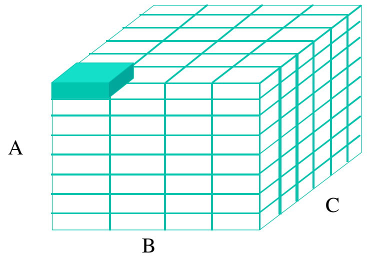
Figure 9. What is an array?

Programmers are familiar with the concept of an array as a reserved area of working memory that is used to temporarily store information. Part of the beauty of programmatically storing data in an array resides in the fact that when there is a requirement to retrieve data, the contents are *directly* accessed.

Declaring a three (3) dimensional array such as array(10,15,3) in Figure 9 would result in the creation of 450 intersection points as the cross-product, or Cartesian product, of all members from each dimension, in this case 10*15*3. Even within the context of this trivial example we can discuss the concept of sparseness.

Let this array store sales of our 10 products sold by 15 salespeople across 3 time periods. Because *not* every sales person sells every product in every period, there are cells that will *never* contain values. A normalized relational data store would only store the values from transactions of sales actually made. A matrix reserves space for every possible value, whether it makes sense to do so or not.

The referencing of contents in an array (see Figure 10) is achieved simply by supplying the appropriate coordinates of intersection points — that is, by supplying the cell address. Data retrieval of DB2 OLAP values is nothing other than the supplying of cell addresses to the DB2 OLAP Server.



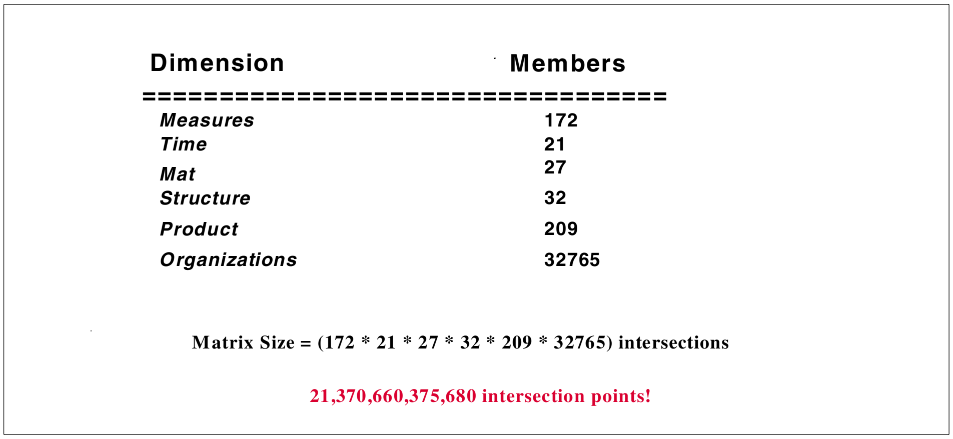
To address the ***contents*** of any cell in an array, supply the cell address *(A1,B1,C1)* in this example

Figure 10. Referencing contents

On a deeper level, however, DB2 OLAP enables users to reflect complex business rules (calculation scripts and formulae) even down to the level of an individual intersection point within a given multidimensional view of an organization. This permits real and complex business models to be computer developed and tested.

4.3.1 The concept of sparseness: dimension tags

The overall size of any array or matrix is the Cartesian product of all of the members across all dimensions. The next picture illustrates that even for a rather small multi-dimensional database, the number of possible intersection points can be daunting. It is self-evident to those experienced using multidimensional databases that most intersections will *not* contain data. This reflects the fact of the inherent sparseness of all business datasets (see Figure 11).



Dimension	Members
Measures	172
Time	21
Mat	27
Structure	32
Product	209
Organizations	32765

Matrix Size = (172 * 21 * 27 * 32 * 209 * 32765) intersections

21,370,660,375,680 intersection points!

Figure 11. Multidimensional databases are inherently sparse structures

However, an array generates an intersection point for every member from each dimension across every member of every other dimension, regardless of whether that intersection is reflected in business reality.

Recall that relational storage systems were designed to store only data generated by the business. For example, the idea of *storing* the fact that no picnic baskets were sold in the Yukon in January is absurd to a relational database schema design. In a multidimensional structure, the intersection of “picnic baskets with the Yukon with January” exists by declaration. In the final analysis the *DB2 OLAP storage structures were specifically designed to enable designers to deal effectively with this inherent sparseness,* and the inherent sparseness of DB2 OLAP arrays is managed by tagging dimensions as Sparse or Dense.

Dense/sparse settings

- Tag dimensions either sparse or dense
- This creates unique DB2 OLAP storage structures (BLOCK and INDEX)
  - ***Reflect density within the block***
  - ***Reflect sparseness within the index***

Dense dimensions comprise the data block. In general it is accurate to say that *dense* dimensions are implemented to reflect the *density* of the data set (in the *data block*) and *sparse* dimensions are implemented to eliminate the *sparseness* of the *data set*:

> ***A data block is created only where sparse member combinations contain data as illustrated in figure 12.***

But it must be understood that the block storage structure will, in all probability, retain some of the database sparseness (being defined by the Cartesian product of all stored members from each dense dimension.) That is, there will be empty cells within the data block. This is another way of saying that it is not possible to eliminate all sparseness from a DB2 OLAP database. The objective is thus to minimize the impact of sparseness.

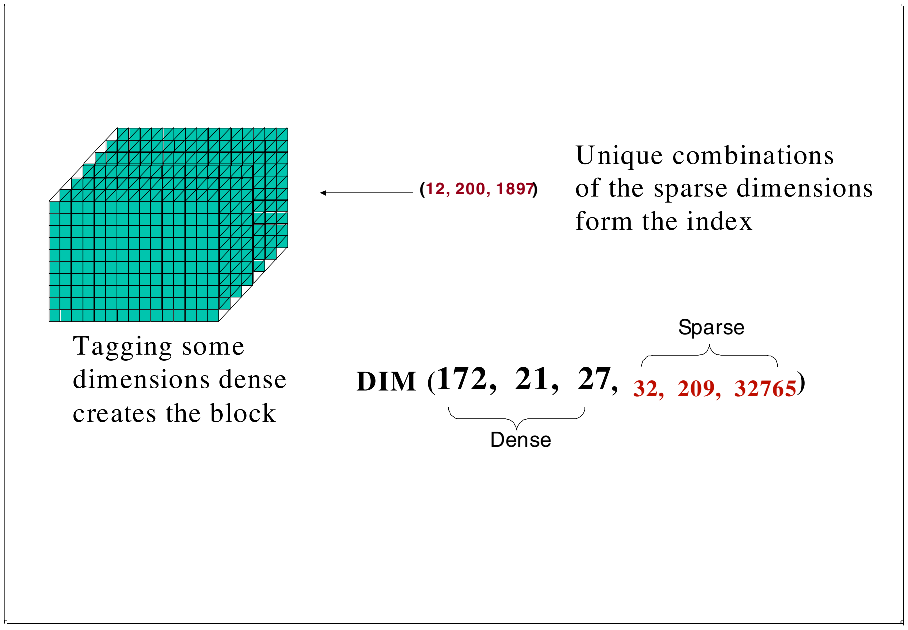

(12, 200, 1897)
Unique combinations of the sparse dimensions form the index
Tagging some dimensions dense creates the block
Sparse
DIM (172, 21, 27, 32, 209, 32765)
Dense

Figure 12. Index and block creation

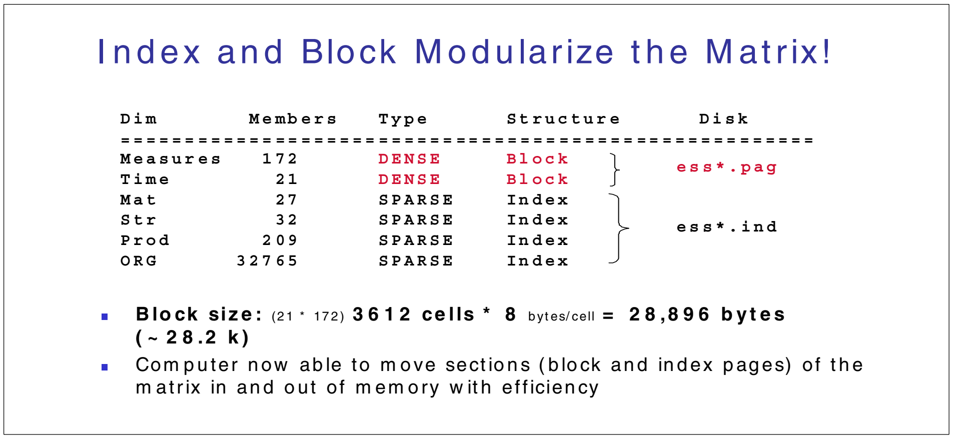
The minimization of sparseness is achieved by paying attention to the fact that in a DB2 OLAP environment, *business relevant* sparse member combinations drive data block creation. If a sparse combination does not exist, which means business data does not exist, a block is not generated.

The inherent sparseness of the matrix is dealt with by appropriately tagging dimensions. That is, dimensions that have the fewest intersection points with respect to each other are tagged as sparse. When sparse tags are correctly applied, the majority of the intersection points are eliminated from the array. This is how very large business models can be implemented using DB2 OLAP.

However, a *sparse* dimension that has been *tagged as dense* causes the proliferation of empty and perhaps meaningless cells within the database. There are times when no business activity is a meaningful business event, and analysts need to know about them. For example the fact that we had no sales of bathing suits in May might have tremendous importance to a Miami based clothing retailer.

Conversely, when the dense nature of a data set cannot be contained within the block structure, that is a dense dimension has been tagged sparse, then this results in an explosion of the number of data blocks that the configuration generates.

Multidimensional databases have a fixed number of dimensions. Thus the matrix can be implemented using one of a number of different database configurations (sparse/dense settings). Dense dimensions that form the data block structure are stored on disk within the ess*.pag file(s). Dimensions that are tagged as sparse form the index structure and are stored on disk within the ess*.ind file(s), see Figure 13. Index entries are pointers to data blocks and contain components of the (array) address to cells, and data blocks are correctly conceived of as where data is actually stored.



Index and Block Modularize the Matrix!

Dim	Members	Type	Structure	Disk
Measures	172	DENSE	Block	ess*.pag
Time	21	DENSE	Block	
Mat	27	SPARSE	Index	ess*.ind
Str	32	SPARSE	Index	
Prod	209	SPARSE	Index	
ORG	32765	SPARSE	Index	

- **Block size:** (21 * 172) **3612 cells * 8** bytes/cell **= 28,896 bytes (~ 28.2 k)**
- Computer now able to move sections (block and index pages) of the matrix in and out of memory with efficiency

Figure 13. How the matrix is built

Goal of the database designer:

The goal of the database designer is twofold:

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

To repeat, *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 matrix. Also recall that 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 blocks that the configuration generates will result. Conversely, largely empty data blocks indicate that sparseness has not been effectively eliminated from the data set.

4.4 Tuning the outline

Tuning the outline involves the following:

- Dimensionality and business logic
- Member tags
- Consolidation types
- Sparse/dense methodology

4.4.1 Dimensionality and business logic

The objective of every database schema is to reflect the business logic of the specific analytic problem at hand. Whereas it is a task beyond the scope of this redbook to discuss solutions to business-specific algorithms, certain generic notions can be brought to light to design the outline:

- DB2 OLAP values are persistent values

  a. How many dimensions?

  b. How large are the dimensions (total members)?

  c. How deep are the dimensions?

- How a model performs will reflect the interrelationship between the last 3 bullets.

4.4.1.1 Generic data storage considerations

DB2 OLAP optimizes run-time performance by making (the vast majority of) data set values persistent. This has obvious batch calculation and disk storage implications. To help defray the cost of 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*.

4.4.1.2 Generic outline considerations

The main considerations are:

1. **How many dimensions?** 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. Look for ability to change a dimension from a regular dimension to an attribute dimension. This greatly decreases size and calculation times since the attribute dimension is done on the fly, but is mostly effective on smaller database models.
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 must 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.

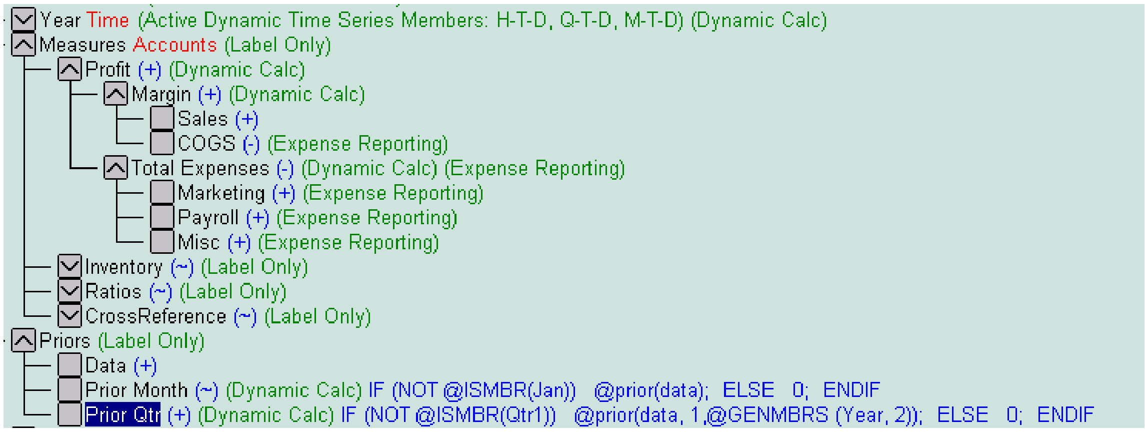
4.4.1.3 When to add a dimension

The idea of intentionally adding a dimension to a database appears counter-intuitive at first reading. It really needn't be. In a certain sense, it is simply a variation of the basic design technique that suggests splitting dimensions when you find repeating labels.

For example, it is suggested that when you encounter metrics like Actual Sales, Budget Sales, Forecast Sales an so forth, it is best to add a separate Scenario dimension containing 3 members: Actual, Budget and Forecast. Now because of the nature of multi-dimensional data set these 3 Scenario members intersect with every member from the metrics dimension, and this will occur across *all* members of *all* other dimensions. The addition of the Scenario as a dimension enables users to perform all of the wonderful magic that accrues to OLAP databases – slicing and dicing, drilling, pivoting and so forth. Let's take the example to another level.

Consider the requirement to see all of your current database metrics in relation to the identical metric of the prior period. This could mean creating a new metric for each existing metric to store the variance. For example the variance between Current Sales and Prior Sales. This effectively doubles the size of the Measures dimension.

On the other hand, Figure 14 shows the inclusion of the Priors dimension.



```
Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D, M-T-D) (Dynamic Calc)
Measures Accounts (Label Only)
    Profit (+) (Dynamic Calc)
        Margin (+) (Dynamic Calc)
            Sales (+)
            COGS (-) (Expense Reporting)
        Total Expenses (-) (Dynamic Calc) (Expense Reporting)
            Marketing (+) (Expense Reporting)
            Payroll (+) (Expense Reporting)
            Misc (+) (Expense Reporting)
    Inventory (~) (Label Only)
    Ratios (~) (Label Only)
    CrossReference (~) (Label Only)
Priors (Label Only)
    Data (+)
    Prior Month (~) (Dynamic Calc) IF (NOT @ISMBR(Jan))   @prior(data);  ELSE   0;  ENDIF
    Prior Qtr (+) (Dynamic Calc) IF (NOT @ISMBR(Qtr1))   @prior(data, 1,@GENMBRS (Year, 2));  ELSE   0;  ENDIF
```

Figure 14. Adding a dimension in the outline

Note the member tags for this dimension. The dimension name (Priors) is tagged Label Only. There is a single member called Data that is the point to which all of the data is loaded to for this dimension. Then there are 2 dynamically calculated members below Data: Prior Month and Prior Qtr. Each dynamic member uses a version of the @PRIOR() function. Because these members have been added to another dimension, they intersect with every other metric within the Measures dimension. This is the multidimensional paradigm in action.

Pay close attention to the configuration. We have added to this database a dimension that has a storage magnitude of 1. If we tag this as a dense dimension, we are multiplying the current block size by a factor of 1. In other words, the block size remains the same!

We have increased the complexity of our outline, true. We have also increased the size of the logical block, and this could have an impact on performance. But we have added a dimension that adds nothing to the storage structures but almost breathes new life into the functionality of the model. Indeed, adding more dynamic members to the Priors dimension can be done effortlessly and almost with impunity.

4.4.2 Considerations on use of member tags

Member considerations

- Store data only when necessary.
- Use tags to reduce block size:

  Label Only

  Dynamic Calc members

The use of member tags is reduced here to the good practice principle *of storing data only when there is a necessity of doing so*. For example, DB2 OLAP Dynamic Calc members are designed to return values only when requested by the client. They can reflect data density by demand. That is, a value is dynamically calculated at run-time because the user expects a value to be returned.

4.4.2.1 Label Only and Dynamic Calc tags

In the case where all data values in the matrix are pre-calculated, the only optimization tip would be the intelligent use of Label Only tags. Label Only tags reflect the business requirement to store values that make sense within the business paradigm. The label only tag reflects the business requirement to have members which function only as “headings” or “grouping” members. Their value is in providing a placeholder for grouping like members together, not in their quantitative value. “Scenario” for example, is a parent that holds all the planning members in the same group.

The block detailed in Figure 15 contains 6*4*3=72 cells.

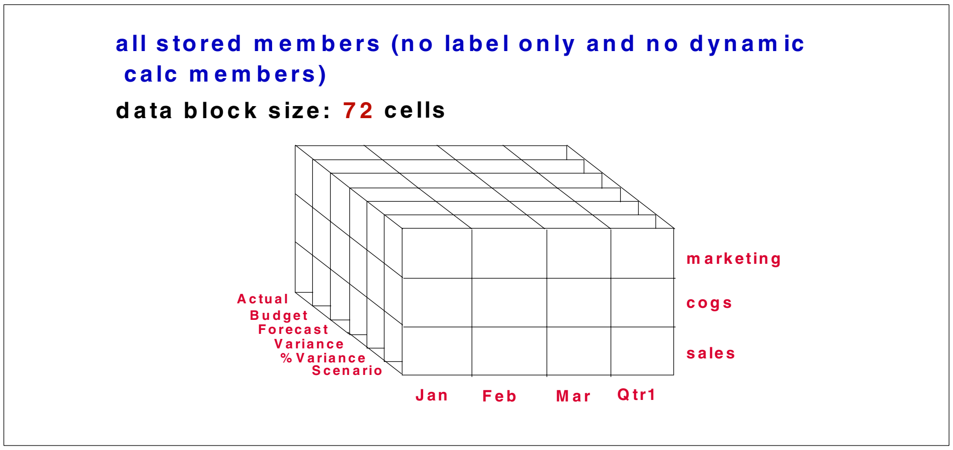

all stored members (no label only and no dynamic calc members)
data block size: 72 cells
marketing
cogs
sales
Actual
Budget
Forecast
Variance
% Variance
Scenario
Jan
Feb
Mar
Qtr1

Figure 15. Member considerations: the block

Effective use of the *Label Only* on the Scenario member in Figure 16 reduced the block size by 12/72 cells in the above example. The net result is a potential reduction in overall database size of16,7%. Because of data compression, the net reduction in overall database size will probably not be equal to 16.7%.

A 16.7% reduction of block size is not inconsiderable but use of Label Only is limited to navigational members in the outline. Using dynamically calculated members, on the other hand, can greatly increase the database designer’s ability to reduce database size (stored values).

Effect of using LABEL ONLY

data block size: 60 cells now

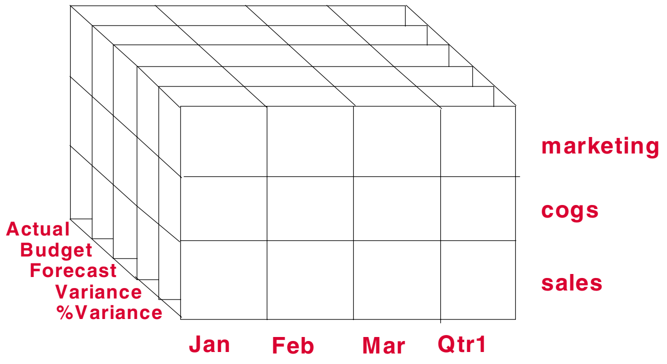


Figure 16. Member considerations: Label Only

Implementing dynamically calculated members in the sample data block on all parent members and members with a formula radically alters the storage requirements of the matrix. The block design in Figure 17 results in a data block that is 62.5% smaller than the original block and 56.5% smaller than the block size achieved by using *Label Only*!

Parent and formula members made dynamic calc and place holder members made label only

data block size: 27 cells now

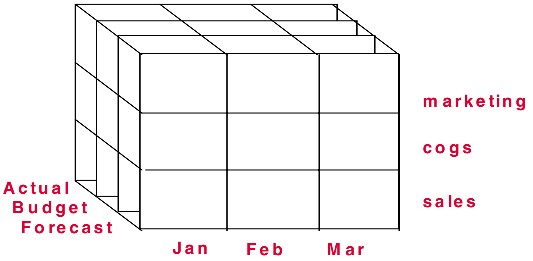


Figure 17. Member considerations: dynamic calculation

4.4.2.2 Physical and logical blocks

Implementers must learn think about the size of the block as being different under different circumstances. The physical block on Figure 18 is the block that has been optimized for storage. It is represented by the *Block size in bytes* statistic in Application Manager and the *Actual block size* in cells reported by GETDBSTATS. If the database is properly configured, the physical block is the block that DB2 OLAP retrieves during batch calculation.

The Logical Block in Figure 18 is the block that is fully expanded somewhere in working memory (***Not*** the DB2 OLAP caches) and includes storage space in RAM for all dynamically calculated cells. Remember, DB2 OLAP only calculates dynamic values that are requested. But it must reserve space somewhere else in working memory other than the caches for those values. However, you should be aware that, during batch calculations, the DB2 OLAP calculator might (inadvertently and inappropriately) have need to dynamically utilize the space in the data cache to fully expand data blocks.

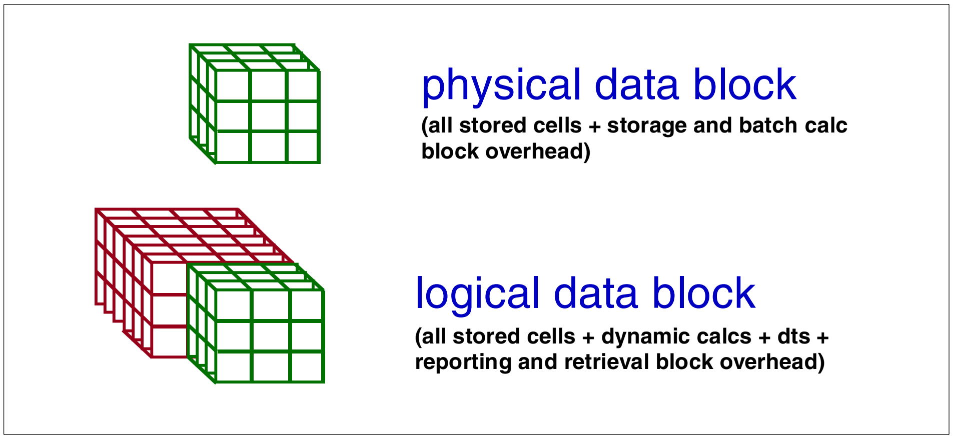
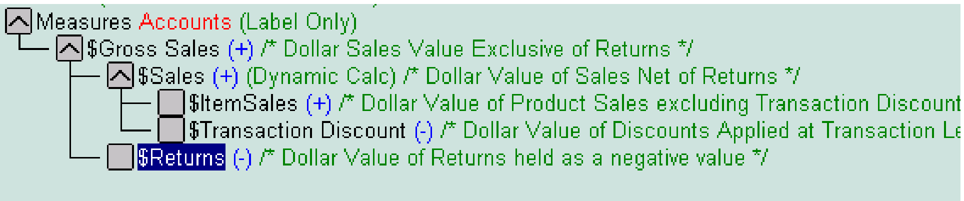

physical data block
(all stored cells + storage and batch calc block overhead)
logical data block
(all stored cells + dynamic calcs + dts + reporting and retrieval block overhead)

Figure 18. Two types of blocks? When will it end?!

Dynamic calculations are involved in batch calculations when, for example, a stored member value is dependent upon dynamically calculated ones.

Consider the outline fragment in Figure 19. Notice that the member *$Gross Sales* is a stored member and its value depends upon the dynamically calculated *$Sales* member. An outline of this configuration will cause the calculator to fully expand the data block to its logical size within the data cache during a batch calculation.

This results in an under-sizing of the data cache.



```
Measures Accounts (Label Only)
└─ $Gross Sales (+) /* Dollar Sales Value Exclusive of Returns */
   ├─ $Sales (+) (Dynamic Calc) /* Dollar Value of Sales Net of Returns */
   │  ├─ $ItemSales (+) /* Dollar Value of Product Sales excluding Transaction Discount
   │  └─ $Transaction Discount (-) /* Dollar Value of Discounts Applied at Transaction Le
   └─ $Returns (-) /* Dollar Value of Returns held as a negative value */
```

Figure 19. Outline fragment

The administrator may have configured the data cache to hold 100 physical data blocks for batch calculation, but because the calculator is fully expanding blocks in the data cache at runtime to perform these dynamic calculations, there might only be room for 2 or 3! This will needlessly protract the batch calculation process.

4.4.2.3 A word of caution

The data block is the fundamental data storage structure in DB2 OLAP and database configurations are ultimately constrained by block sizes. This is *not less true in version 7 and later release* than in earlier releases. That is, in Hyperion Essbase OLAP Server versions prior to release 5.

4.4.2.4 What block size is best?

There are well-tuned and responsive production databases with data block sizes ranging from a few hundred bytes to 1 megabyte. But in the final analysis, the optimal block size is the one that provides the best performance. And like it or not, database implementers alone understand sufficiently all of the criteria by which to determine that database performance is within acceptable limits.

One indication that the optimal size has been isolated might be that the fastest batch calculation time has been achieved. We believe that the database configuration that generates the fewest most dense data blocks will correlate highly with the fastest calculation time. We will provide a methodology for efficiently determining this configuration in the text that follows. Database designers need to be all be aware, however, that other (procedural) issues at times may take precedence over the sparse/dense configuration when tuning.

Some databases will have block configurations that are optimized according to user query requirements, or specific complex calculation requirements, or both. Database tuning and optimization may or may not be more art than science, but our conviction is strong that a good understanding of the nature of the DB2 OLAP storage structures is fundamental to database tuning.

Optimal block sizes, then, are model specific. So the best practice is not to provide to developers a mythical block size range objective, but rather a method for quickly being able to isolate the optimal block sizes for specific databases at hand.

Dynamic member practices

- Incorporate more data density within data block using dynamic members
- Rationale

  Dynamic members reduce the block size

  Creates opportunity to incorporate other dimensions into the block structure

  Number of different block configurations increases

Block configurations are still constrained by overall block size!

4.4.2.5 Dynamic Calc tagging conventions for optimization

It is possible to consider implementing a wider variety of database (block) configurations because of the implementation of dynamically calculated members. The ability to suspend the calculation of certain members from the batch calculation by using dynamic members enables the designer potentially to incorporate more data set density within the data block.

Be aware that there still exists the practical limit of the block size that is compounded by DB2 OLAP having to dynamically produce data values. In the final analysis the optimal database design will efficiently coordinate the relation that exists between:

1. Block size
2. Number and nature of dynamic retrievals
3. Hardware configuration

4.4.3 Considerations on consolidation types

We can conceive of at least three general types of DB2 OLAP database consolidation types. Each will have its own performance characteristics within the context of DB2 OLAP storage structures.

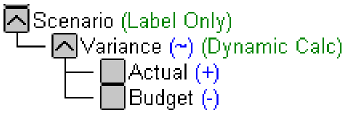
Three consolidation types

- Natural consolidations:
  - Without formulae
  - With dense member formulae
- Special case consolidations
  - Outlines that require fixing or sparse member formulae

1. **Optimal configuration**
   - Unary operators reflect all business logic
   - Database is calculated in outline order according to the default calculation algorithm
   - Block creation and consolidation is regular and predictable
2. **Nearly-optimal configuration**
   - Formulae on dense members reflect some business logic
   - Cross-dimensional references resolved within the block performed as they appear in outline
   - Database is calculated in outline order according to the default calculation algorithm
   - Block creation and consolidation is regular and predictable
3. **Sub-optimal configuration**
   - Formulae on sparse members
   - Contain cross-block references to sparse members
   - Block creation is not regular and predictable

An optimal consolidation takes full advantage of the programming logic implemented by DB2 OLAP engineers. All business logic is reflected by the (unary) hierarchical relationships in the outline. Block creation is regular and predictable and can easily be coordinated with DB2 OLAP buffers for optimal hardware utilization. CALC ALL initiates the consolidation and the internal algorithms built into the calculator engine efficiently generate data values.

Figure 20 shows an example of natural consolidation with unary operators.

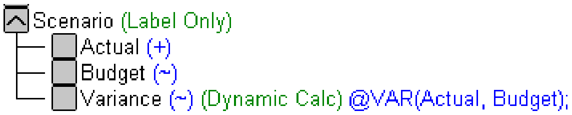


```
Scenario (Label Only)
└ Variance (~) (Dynamic Calc)
   ├ Actual (+)
   └ Budget (-)
```

Figure 20. Example of natural consolidation with unary operators

A nearly-optimal consolidation reflects more complex scenarios where business logic cannot be imbedded in the unary hierarchies and business formulae are required. All formulae are able to be resolved within the data block. So the main difference the optimal and nearly-optimal consolidation resides in the additional overhead to process member formulae. Because all formulae are intra-block formulae (do not involve multiple blocks), block creation is still regular, predictable and efficient.

Figure 21 shows an example of natural consolidation with formulae.

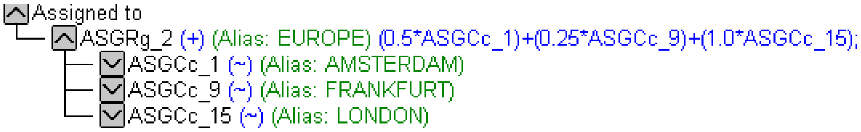


```
Scenario (Label Only)
├ Actual (+)
├ Budget (~)
└ Variance (~) (Dynamic Calc) @VAR(Actual, Budget);
```

Figure 21. Example of natural consolidation with formulae

Sub-optimal configurations involve the FIX statement or inter-block calculations, or both. The formulae that initiate sparse cross-dimensional references can be contained within the outline or within a calculation script but performance is directly related to the number and degree of sparse member references.

Figure 22 shows an example of formulae used for the sparse dimension Assigned to.



```
Assigned to
└ ASGRg_2 (+) (Alias: EUROPE) (0.5*ASGCc_1)+(0.25*ASGCc_9)+(1.0*ASGCc_15);
   ├ ASGCc_1 (~) (Alias: AMSTERDAM)
   ├ ASGCc_9 (~) (Alias: FRANKFURT)
   └ ASGCc_15 (~) (Alias: LONDON)
```

Figure 22. Example of sparse members with formulae

Attempt to ensure natural consolidation by bringing all formulae into the outline, placing all formulae within the data block and resolving cross-dimensional references within the block. Designers should take advantage of DB2 OLAP storage structures by implementing a natural (default calculation) database consolidation whenever possible.

4.4.4 Sparse/dense methodology

The twofold goal of the database designer (to make the fewest most dense data blocks) suggests that there is a relation between the density of the data block and the number of blocks generated. Indeed, within any DB2 OLAP database the relation between the density of the data block and the number of blocks created can be expressed mathematically as:

(data set density) = (block density)/(number of blocks) or

Dd = Bd/Nb

The twofold objective of the developer is *not* to create the fewest number of blocks, neither is it to create the densest blocks, but rather *to create the fewest number of blocks with the greatest density*. Using the data set density ratio that reflects this relationship we simplify the task of determining optimal sparse /dense settings to finding the highest ratio.

It should be evident that the ratios themselves are trivial numbers. A block density of 1% for a model that generates 35 million blocks yields a data set density metric of very small proportions indeed. These ratios are not relevant between databases. *They only have relevance to the same values derived for the same database using the same source data over different sparse/dense configurations!*

Reminder

- Reflect data density in the block, data sparseness in the index

  Create as few of the most dense blocks possible

- There is a relation (data set density Dd) between block density (Bd) and number of blocks (Nb) that can be expressed as the following ratio:

  Dd = Bd/Nb

Any given database instance has a fixed set of intersection points that contain data, and there is also a fixed set of ratios between block density and number of blocks for any database instance (By database instance is meant a populated database at a single point in time).

As the block density metric goes *up*, the overall ratio goes *up*; as the number of blocks metric goes *down*, the overall ratio goes *up*. *The highest ratio reflects the optimal relation between density and number of blocks across all database configurations for the same database instance*. Or, if you prefer, the highest ratio points to the configuration that generates the least number of most dense data blocks. (See Figure 23.)

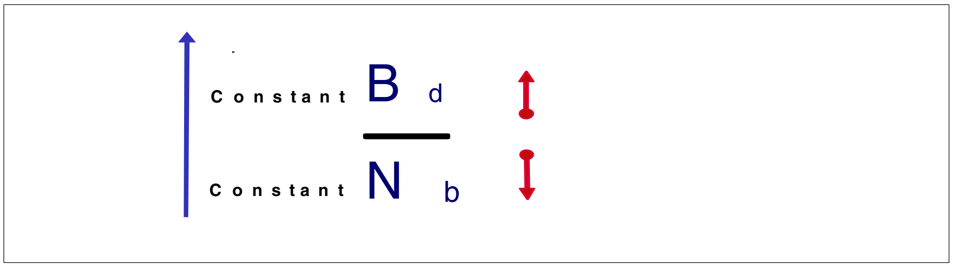


Figure 23. Observations and method

Combining concepts from the previous discussion, we can detail a sparse / dense methodology. The method assumes first and foremost that the database outline already had been designed to accurately reflects business logic.

Working through various block configurations:

1. Place all formulae within the outline.
2. Set all upper members / formulae within the data block Dynamic Calc Non-Store.
3. Load data to different sparse/dense configurations to create table of values for Bd/Nb.

Dense/Sparse settings

Once outline design has resolved issues of business logic and dimensionality

Reflect data density in the block, data sparseness in the index

- *Combine guiding principles to create a dense/sparse methodology*
  - Place formulae in outline
  - Tag members in block dynamic
  - Reflect density in block, sparseness in index
  - Seek high ratio Bd/Nb

4.4.4.1 The method in detail

DB2 OLAP developers have been implementing a version of this methodology all along. That is, they have repeatedly tested different configurations and, by measuring the calculation time, were able to determine an optimal configuration. Or perhaps not. Designers generally seek the configuration that meets the batch-processing window. There may, in fact, be more than one database configuration that enables the database to be calculated within the batch window. On the other hand, the methodology will reveal what the optimal database configuration is even if designer does not implement this configuration.

Note: A configuration utility has been developed that is freely available for download at the following Web sites:

```
www.essbase.com
www.ibm.com
www.olapunderground.com
```

The unfortunate pre-condition to determining the optimal configuration in this manner is that it is very time consuming has generally been predicated on the exclusive use of a (development) server.

A developer who wants to test for the most efficient model will run performance tests in a controlled environment. They want to eliminate as much resource contention as possible in order confidently to determine which configuration results in the calculation time. This not only usually means that for the duration of the test calculation in progress other server activity is restricted, but also can result in test iterations that are tremendously time consuming.

Each database configuration tested has to run at least as long as the shortest (or best to-date) test calculation time. Database configuration testing can be extremely resource intensive and time consuming. This explains why a developer will be satisfied with the first and not necessarily the most optimal configuration that fits within the batch processing window.

In order to truncate this process, a methodology was developed to produce an indicator of optimal database configuration in the shortest amount of time.

Set dense members with children or formula to Dynamic Calc
Tag these members Dynamic Calc Non-Store. The Dynamic Calc and Store tag does not help to reduce the storage requirements of the data block and is really reserved for sparse member tagging.

All DB2 OLAP databases are constrained by the size of the data block. We can see no reason to preclude the possibility that a data set exists whose dense dimensions define a data block that is simply too large to be implemented. The ratio that is computed relates the data to their storage structures. Block size is a hardware (and software) restriction and though critical to database configuration, does not pertain to the matrix itself.

By parsing the array into the two array components (sparse and dense), the creator of Arbor Essbase really did invent an ingenious way to modularize the matrix and enable huge arrays to be implemented on relatively small computer hardware. The design objective really can be thought of as determining which block size functions best to minimize I/O.

However, by looking for a specific block size based on assumptions of optimal size may end up missing something more than the point. The method computes the ratio over a set (subset) of possible database configurations. The end result is a table of values that can be easily represented graphically.

This graph (or table of values) *accurately* describes the dense/sparse characteristics of the model (array) and *metaphorically* describes the performance characteristics of a given database. In complex models, which configuration is finally selected will be determined by a consideration of all factors, block size, process and calculation requirements being perhaps the three most important ones.

By computing the ratio rather more freely we hope to let database designers learn something more about their data. Hopefully this information will have heuristic value. So, rather than imposing block size limits, we decided to enable the method to be used to determine the relative significance of almost any configuration.

The current version of the methodology physically instantiates database configurations, and thus lives within the world of available RAM. As a result, it will not be possible to test data block configurations whose memory requirements exceeds the maximum block size that DB2 OLAP is able to access on any given machine. The ability to determine block size without actually creating the data block will eventually free the method from this restriction, though the practical reasons for not doing so (large dimensions are inherently sparse) probably do not warrant the functionality.

The methodology for setting Dynamic Calc Non-Store members in the block has become the *de facto* standard for block design. Setting all possible members to dynamic reduces the block size. This can have dramatic impact on the batch calculation time. However, possibly the most significant benefit from this use of dynamic members is that it enables designers to incorporate dimensions within the block that would not otherwise be able to be incorporated because of block size limitations. In the final analysis, this method enables more data density to be incorporated within the data block.

Load real and completely representative data to the database

The accuracy of the method is critically dependent upon being able to load real production data across a full range of the database dimensions. We need to have all cells populated across the dense dimensions to be confident that the average block density that DB2 OLAP reports is representative. Similarly, we need all points populated across the sparse dimensions to be confident that total block creation is also represented. Because we iterate through a range of configurations, we really need to have completely representative data.

Note, however, that we can fudge within certain limits. If (and only if) it is known that a dimension will be either sparse or dense and not tested for its opposite, then the data need not be representative.

Consider a block that has 12 months, and 4 quarters. If it is known for certain that time will be tagged *dense*, then loading data to only a subset of members (or 1 member) across time will have the same impact on block density across all tested configurations. So, if time will always be part of the block, loading only one month will yield reliable metrics for block density across all other test configurations. On the other hand, if data is only loaded to one month and time is tagged *sparse*, then the number of generated data blocks would be greatly underestimated and skew results

Retrieve average block density metric from database statistics

The computation of average block density reported by database information statistics is as follows: the total number of blocks is divided into 100 equal parts. Each 100th block is sampled for density and the average density is computed from these 100 blocks.

For the purpose of minimizing the time to compute the ratio we are assuming that the average block density at data load time will be representative of the block density of the fully calculated model. We are assuming this for a specific database instance. That is, we make this assumption based on the requirement that the same data set is loaded into the same outline across possible sparse/dense configurations for that outline.

Block density assumption

The average block density at data load time is representative of overall block density after calculation in relation to the other sparse-dense configurations

Retrieve from database statistics the number of blocks generated

For the purpose of minimizing the time to generate the ratio we are assuming that the number of data blocks created at data load time will be representative of the total number of blocks of the fully calculated model. We are assuming this for a specific database instance. That is, we make this assumption based on the requirement that the same data set is loaded into the same outline across possible sparse/dense configurations for that outline.

Block count assumption

The number of blocks generate at data load time is representative of the overall number of blocks that would be generated after calculation in relation to the other sparse-dense configurations

We do not want to convey the notion that *every* database application created in DB2 OLAP will conform to these assumptions. There is an error rate that has not, as yet, been determined for this methodology. We do want to suggest, however, that the vast majority of models will conform. This is what testing has revealed to-date.

4.4.4.2 Non-optimal implementations?

The metric described above really is an indicator of how data is displaced across the array within a given model data. The fact that it also can be used as an indicator of the performance characteristics of the database is not coincidental. A design that eliminates the most amount of sparseness from the schema results in a design that has the smallest amount of disk storage requirements. And from this point of view, this is the configuration that involves the least amount of I/O.

There is no question in our minds that optimal array configurations will not always be implemented. Some of these are discussed in brief detail as follows.

Dense restructures and incremental dataloads

There are processes that require incremental data loading. The classic example is best illustrated by considering incremental data loads that occur across time. For example, it is likely conceivable that whereas the method reports that the time dimension should be considered part of the dense dimensions, the developer will implement a design tagging time as sparse to support the incremental loading of data across time.

Indeed, dimensions that undergo regular member insertions are strong candidates for the sparse tag even though they may be part of the density of the data set. If the refresh process does not enable a complete re-load and re-calculation of the database, then member additions to dense dimensions would require a full dense restructure. The designer would seriously want to consider setting such dimensions sparse to avoid having to undergo dense restructures on a regular basis.

Overall block size

It is not inconceivable that the method will point to a configuration that would necessitate implementing a block size that is simply not supportable on the hardware infrastructure being used. In fact, this was precisely the case for Hyperion Essbase OLAP Server versions prior to the release of Dynamic Calc members in version 5.

4.5 Considerations on database calculation

In this section we discuss the main considerations when calculating the DB2 OLAP database:

1. Using the Set NOTICE command
2. Using dynamic calculations
3. Focusing calculations

4.5.1 Use the SET NOTICE command

Use the SET NOTICE command when running a development or test environment calculation. It gives a way to compare performance when making a change to a script or database setting without having to wait for the entire calculation to run. By keeping track of the time that the calculation takes to reach the same notification point, you will be able to infer whether the change has succeeded in shortening or lengthening the calculation process. For information on using the SET NOTICE command please refer to the DB2 OLAP documentation.

4.5.2 Dynamic calculations reviewed

In this section we discuss various considerations involving dynamic calculations (Dynamic Calc).

Dynamic Calc review

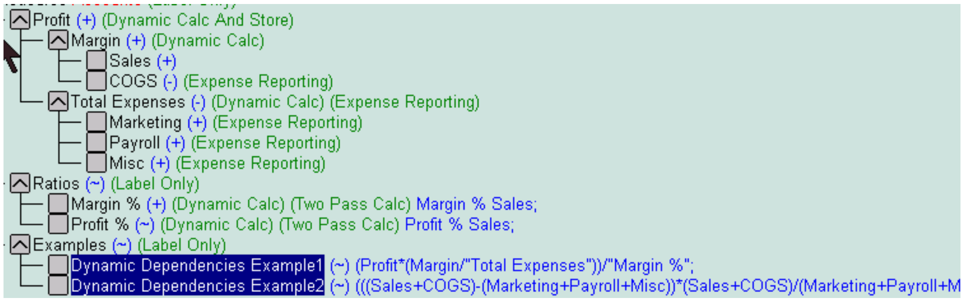
- Dynamic Calc
  - *Upper level dense members*
  - *Dense members with formula*
- Dynamic Calc (and Store) and sparse members
  - *Sparse member with small fan-out (<7)*
  - *Remember cross-products (cartesian) effects of tagging multiple members of multiple (sparse)dimensions dynamic*
    - Exercice caution!

4.5.2.1 Dynamic Calc on dense dimensions

It has become standard practice to tag every member within the data block Dynamic Calc Non-Store. This includes all members with a formula as well as any members with children. This practice ensures that you are only storing those values that you have to.

However, sometimes because of the complexity of Dynamic Calc member cross-references, the ultimate Dynamic Calc value does not get correctly resolved. If you are not getting the correct results in these cases, remove the Dynamic Calc member name from the formula and actually hard code the reference in the member formula itself.

Ignoring the erroneous logic, consider the example of the highlighted example members in Figure 24.



```
Profit (+) (Dynamic Calc And Store)
    Margin (+) (Dynamic Calc)
        Sales (+)
        COGS (-) (Expense Reporting)
    Total Expenses (-) (Dynamic Calc) (Expense Reporting)
        Marketing (+) (Expense Reporting)
        Payroll (+) (Expense Reporting)
        Misc (+) (Expense Reporting)
Ratios (~) (Label Only)
    Margin % (+) (Dynamic Calc) (Two Pass Calc) Margin % Sales;
    Profit % (~) (Dynamic Calc) (Two Pass Calc) Profit % Sales;
Examples (~) (Label Only)
    Dynamic Dependencies Example1 (~) (Profit*(Margin/"Total Expenses"))/"Margin %";
    Dynamic Dependencies Example2 (~) (((Sales+COGS)-(Marketing+Payroll+Misc))*(Sales+COGS)/(Marketing+Payroll+M
```

Figure 24. Outline example with dynamic dependencies

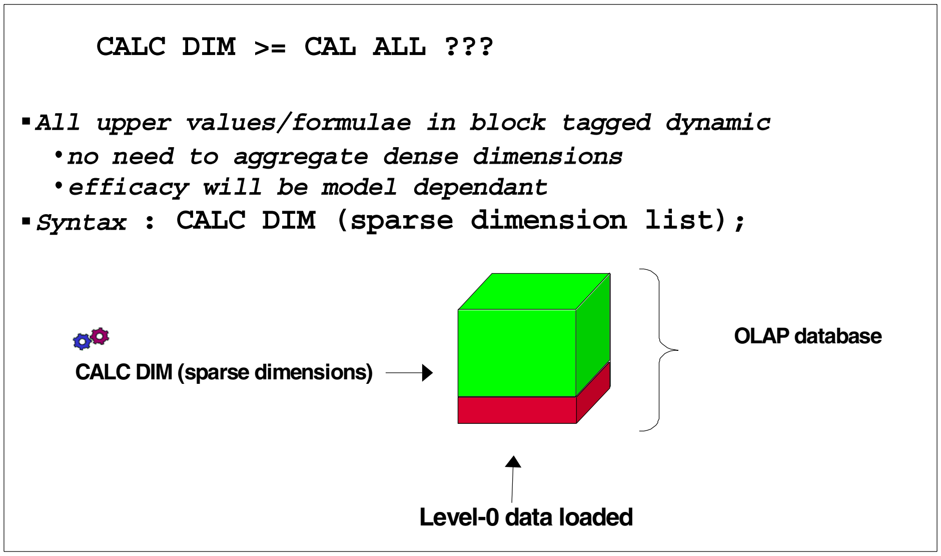
Each dynamic member in the formula on *DynamicDependenciesExample1* has been fully resolved in the formula on *DynamicDependenciesExample2.* Compare the concise formula **(Profit*(Margin%”Total Expenses”))/”Margin %)** to the explicit formula **(((Sales+COGS)-Marketing+Payroll+Misc))*(Sales+COGS)/(Marketing+Payroll+Misc))/(Margin%Sales)**. Though not as concise, the explicit references will certainly resolve the value correctly.

CALC DIM and AGG versus CALC ALL

Databases that implement the Dynamic Calc Non-Store strategy for the data block might also benefit from the following. To recap the methodology, all upper members and members with formulae are tagged as dynamic.

Issuing a CALC DIM (AllSparseDimensionsList) or AGG (AllSparseDimensionsList) command as opposed to the CALC ALL command should be a more efficient method of generating upper data blocks. The CALC DIM (AllSparseDimensionsList) command will ignore the dense dimensions and simply populate the database with all data blocks. Alternatively AGG (AllSparseDimensionsList) will produce the same result but will ignore having to check sparse member formulae. Theoretically AGG should be the most efficient means of producing upper blocks.

Consider that member calculations are only performed on level-0 data and the degree of performance boost will be directly related to the number of level-0 versus upper data blocks (Figure 25).



CALC DIM >= CAL ALL ???

- *All upper values/formulae in block tagged dynamic*
  - *no need to aggregate dense dimensions*
  - *efficacy will be model dependant*
- *Syntax* : CALC DIM (sparse dimension list);

Figure 25. CALC DIM versus CAL ALL

Review the application log entries for two different methods of calculating the same database. The following entry occurred as a result of a CALC ALL:

```
Total Block Created: [0.0000e+000] Blocks
Sparse Calculations: [1.9700e+002] Writes and [8.2900e+002] Reads
Dense Calculations: [1.7700e+002] Writes and [1.7700e+002] Reads
Sparse Calculations: [3.3096e+004] Cells
Dense Calculations: [0.0000e+000] Cells
```

Now compare this to the application log entry for an AGG of the same database:

```
Total Block Created: [1.9700e+002] Blocks
Sparse Calculations: [1.9700e+002] Writes and [6.3200e+002] Reads
Dense Calculations: [0.0000e+000] Writes and [0.0000e+000] Reads
Sparse Calculations: [3.3096e+004] Cells
Dense Calculations: [0.0000e+000] Cells
```

By virtue of the implementation of Dynamic Calc tags, the numeric results of the AGG are identical to the CALC ALL. However (and *still* by virtue of the implementation of Dynamic Calc tags) the calculation time was ~17% faster for the AGG! The application log excerpt highlighted above appears to reflect that the AGG had smaller overhead than did the CALC ALL.

4.5.2.2 Dynamic Calc on sparse dimensions

You can consider using Dynamic Calc tags on upper level sparse dimension members that have only a few children. You will only substantially reduce your calc times and database size if you use these on the upper levels. Beware of using too many. For example, if you tag the top of three sparse dimensions Dynamic Calc and each had 10 children, then to retrieve the grand total of the database, the server has to retrieve 10x10x10 = 1000 blocks and add them up. It can greatly slow retrieval times. You can tag such members as Dynamic Calc and Store with the effect of defraying the cost of subsequent retrievals of the same member.

As a final word, in an array a dimension is a dimension, and a member is a member. In DB2 OLAP on the other hand, dimensions are not all the same. They are tagged either sparse or dense. Dynamic calculations are optimally configured when they reflect this DB2 OLAP reality.

4.5.3 Focusing calculations

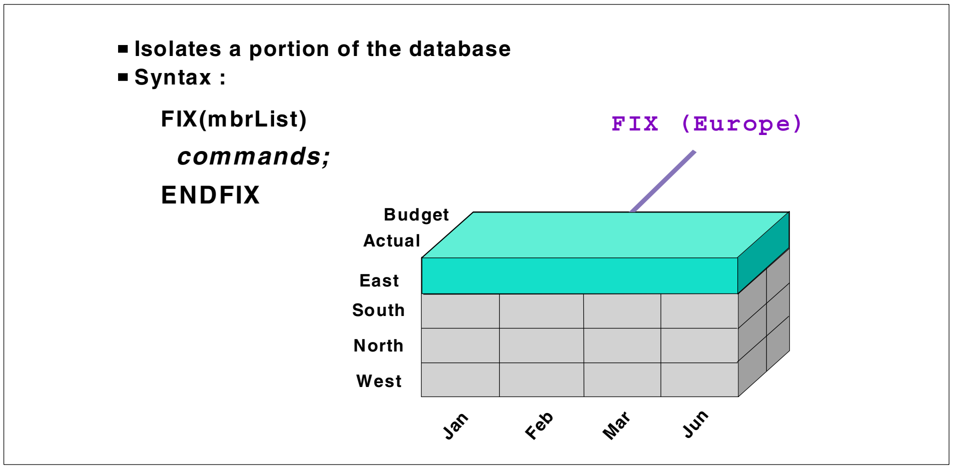
Often it is a database requirement that only a subset of the database be calculated at one time. Focusing calculations is achieved by three means – the FIX command, IF logical construct or the use of the cross-dimensional operator.

The main recommendations should be to ***isolate specific sections of database to increase calculation efficiency***:

- FIX ... ENDFIX: focus on block subsets
- IF...ELSE...ELSEIF...ENDIF: fine tunes logic
- CROSS DIM OPERATOR: granularity to cell level

4.5.3.1 FIX and IF

FIXing a calculation can cause only a subset of the database to be affected by a calculation script (see Figure 26).

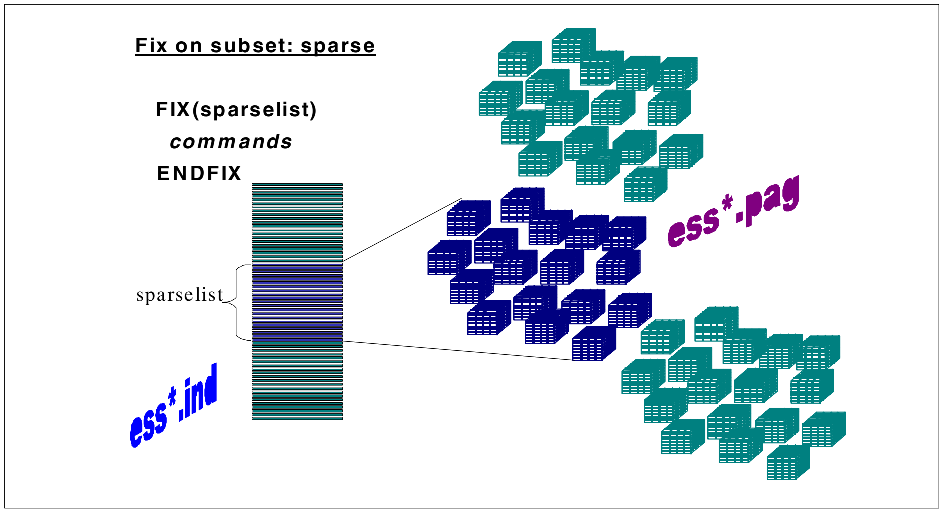


- Isolates a portion of the database
- Syntax :

```
FIX(mbrList)
  commands;
ENDFIX
```

Figure 26. Focusing calculations using FIX,ENDFIX

FIX is best employed as a means of isolating data blocks from the index. FIXING on sparse members (as Figure 27 illustrates) enables the calculator to isolate which blocks need will be required for processing by making a (single) pass through the index.



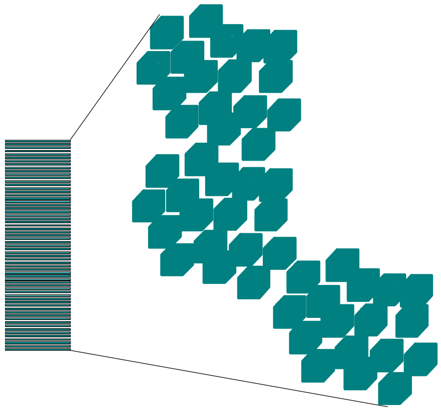
Fix on subset: sparse

```
FIX(sparselist)
  commands
ENDFIX
```

Figure 27. Fixing on sparse dimensions

Fixing on *only* a dense dimension (as Figure 28 illustrates) does not subset the database because every block in the database contains the requisite members. Sometimes the activity of fixing only on a dense member is a necessary calculation requirement.

A dense fix can touch every block !



Assuming that Time is Dense and Region is Sparse, consider:

```
FIX(January)
    IF (Boston)
        commands;
    ENDIF
ENDFIX
```

Figure 28. Fixing on dense dimensions

The logical IF…ELSE… construct can also be used to isolate members for calculation. It should be used only when it is not possible to FIX on data. That is, use IF in scenarios only where you would have to otherwise use multiple FIX statements (which would cause multiple passes across the index.)

The calculation objective in an array database is no different than in a relational one, we want to make as few passes over the storage structures as possible. So, in the final analysis, whichever method proves the most effective for data retrieval during calculation is the one that should be employed.

4.5.3.2 Focusing calculations summary

Use FIX whenever possible:

- FIX to focus on a subset within database
- IF to resolve logic where FIX cannot be used
  - You can FIX in the block
  - You can IF on a subset of blocks

- Use combinations of FIX and IF to most efficiently isolate data for calculation

4.5.3.3 Special case calculation requirements

There are database scenarios that preclude implementing Dynamic Calc Non-Store within the data block as outlined above. In models where the complexity of data calculation (where dependencies between metrics) preclude the ability to make all formulae Dynamic Calc Non-Store, the rules regarding the default calculation order should be followed to ensure optimal database calculation. These rules are made explicit in the *OLAP Database Administrator's Guide* under the section titled *Defining the Calculation Order*.

General calculation script notions are:

- Dense calculations before sparse calculations
  - Generally outdated- all upper values are dynamic but when necessary populate cells with values that aggregate
- Two-pass calculations
  - Dynamic calculation
  - Ensure single pass when not dynamic

4.6 Performance tuning: the buffers

The overall performance of the DB2 OLAP server is related to a number of different factors. A DB2 OLAP database administrator does not have to be an operating system guru in order to optimally configure DB2 OLAP buffers. It should, however, be kept in mind that operating systems have their own memory requirements and DB2 OLAP buffers must be set within these limits.

Buffer management: basics

- Overall performance related to memory
  - *DB2 OLAP buffer requirements*
    - multiple applications
    - multiple databases
    - logical block
  - *Operating System memory requirements*
    - virtual memory

How to allocate resources in a multi-application environment where total resource requirements exceed total resource availability is not easy. Perhaps under-resourced DB2 OLAP environments defy tuning but at the very least they demonstrate the need to manage expectations as much as the need to manage memory.

4.6.1 Guidelines for configuring DB2 OLAP caches

There is no replacement for a systematic and scientific determining of the size of the DB2 OLAP buffers. What follows are some guidelines that will assist in developing an overall strategy for their configuration.

4.6.1.1 The index, data and physical (file system) caches

The data cache is a repository of expanded physical block data pages in RAM and the physical (file system) cache is a repository of compressed physical data block pages in RAM. The index cache is a repository of index address pages in RAM. The DB2 OLAP Storage Manager (or Kernel) manages the index and data caches by providing addressing to data blocks for server functional layers such as the calculator and the reporting module (See Figure 29).

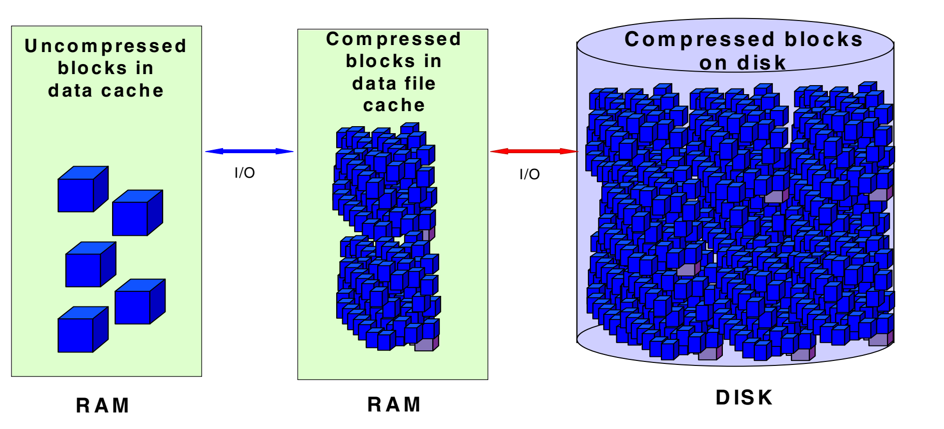

Uncompressed blocks in data cache
I/O
Compressed blocks in data file cache
I/O
Compressed blocks on disk
RAM
RAM
DISK

Figure 29. Data, OS caches and data on disk

Data blocks in the DB2 OLAP data cache are *not* compressed, whereas those blocks in the physical data cache (or OS file system cache) are. As a result of the lack of data block compression in the data cache, increasing data cache buffer will eventually increase likelihood of resolving data fetch from disk because it reduces the total number of blocks that can be held in memory.

In other words, the data cache optimizes at relatively low values. To reflect this reality, favor the physical data cache (or OS file system cache) over the data cache.

Every block fetch involves accessing the appropriate index page. Index entries (112 bytes each in Version 7.1) are generally much smaller than data blocks. Each index page contains multiple block addresses. Taking a similar approach to the index cache as we did with the data caches above, we suggest that the index cache be favored over the physical data cache (or OS file system cache) because proportionally more block addresses than blocks can be fit on a memory page.

It is a balancing act to co-ordinate memory allocations between these three caches. It is not rocket science. Rather the science involved to systematically determine these settings is more like that we learned in our first year of high school: keep all variables but one constant and observe the effect.

4.6.1.2 Batch versus production settings

Buffer memory requirements can be different during the batch calculation than during user query time. For example, a ‘natural’ or optimal batch calculation which regularly and predictably populates a database with data blocks will require much less memory across all three buffers than the same database would after calculation when 500 users are randomly requesting data blocks through ad hoc analysis. In the former case, the hit ratio on index cache can be expected to be very close to 100%, in the latter it will not be anywhere near that number.

4.6.2 The calculator cache

The calculator cache is not a repository of data in RAM. It is rather a roadmap that the calculator engine uses to keep track of block creation and consolidation (Please refer to documentation regarding the configuration of the calculator cache in the *OLAP Database Administrator’s Guide*).

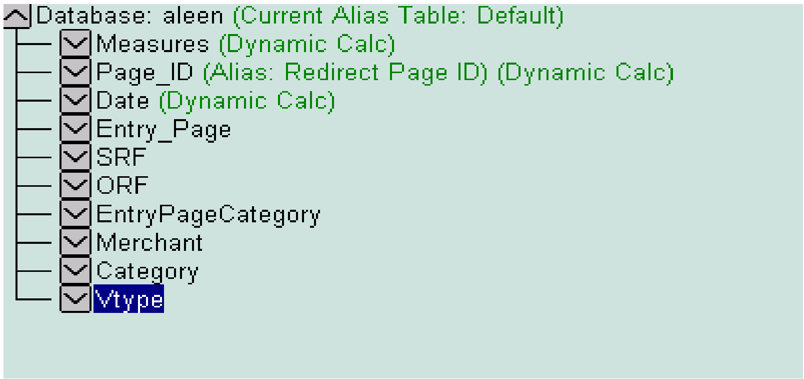
DB2 OLAP documentation reports that achieving multiple bitmaps is desirable and correlates highly with decreased calculation times.

However, the search mechanism through the calculator cache is sequential and not indexed. Although in general achieving a multiple bitmap representation within this cache is desired, there are certain instances where the sparseness of the anchor dimension is so great that searching through the calculator cache is of higher overhead to the system than making a call to the kernel.

4.6.2.1 Single bitmap over multiple bitmaps

In such instances it is recommended good practice to alter the declaration of sparse dimensions (that is to alter their order) in the outline to achieve a single bitmap representation across multiple sparse dimensions as a test to see if the batch calculation time is decreased as a result.

Consider the outline fragment in Figure 30. Measures, Page_ID and Date are the dense dimensions.



```
Database: aleen (Current Alias Table: Default)
├── Measures (Dynamic Calc)
├── Page_ID (Alias: Redirect Page ID) (Dynamic Calc)
├── Date (Dynamic Calc)
├── Entry_Page
├── SRF
├── ORF
├── EntryPageCategory
├── Merchant
├── Category
└── Vtype
```

Figure 30. Outline fragment

The following log fragment indicates that DB2 OLAP established a single bitmap anchor on the last 4 sparse dimensions (EntryPageCategory, Merchant, Category and Vtype):

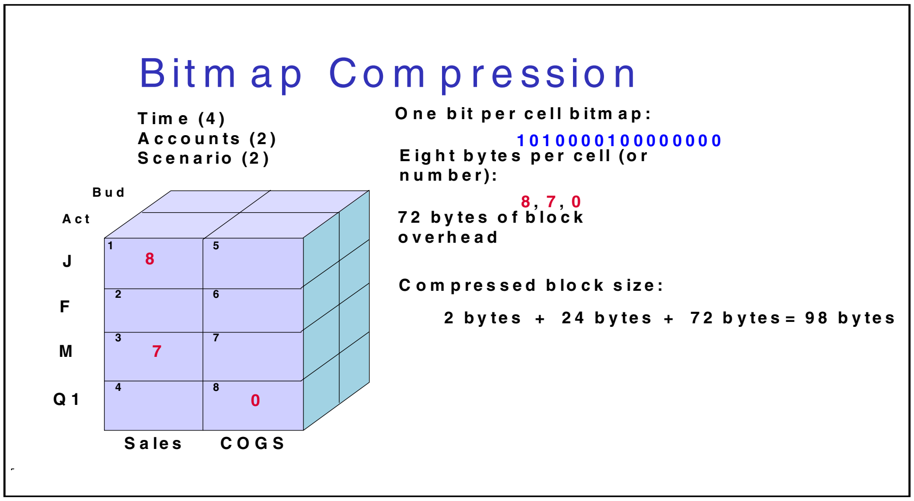
```
Maximum Number of Lock Blocks: [100] Blocks
Completion Notice Messages: [Disabled]
Calculations On Updated Blocks Only: [Enabled]
Clear Update Status After Full Calculations: [Enabled]
Calculator Cache With Single Bitmap For: [EntryPageCategory]
```

The idea here being that making a call to the OLAP Server kernel (across the 4 dimension anchor dimension) would be more efficient than searching sequentially through a large and very sparsely populated multiple bitmap calc cache image.

4.7 Data compression

The data compression type can increase overall system performance. Smaller compressed data blocks are more efficiently moved between disk and memory than larger ones are. Choice of which compression algorithm to use is co-related with block density. DB2 OLAP documentation suggests that when the average block density is around or below 2-3% Run Length Encoding (RLE) compression will be more effective than the default bitmap compression. Smaller blocks on disk would mean that more blocks are read per disk read, and that more compressed blocks can be retained in RAM.

Data compression type can be set for each database. In general, bitmap compression reserves one bit for every cell in the data block and RLE compression maps repeating block values. Figure 31 and Figure 32 provide some of the details regarding how each algorithm would compress the same data block. The point, however, is that the most effective data compression co-relates with increased I/O performance.



Bitmap Compression

Time (4)
Accounts (2)
Scenario (2)

One bit per cell bitmap:

1010000100000000

Eight bytes per cell (or number):

8, 7, 0

72 bytes of block overhead

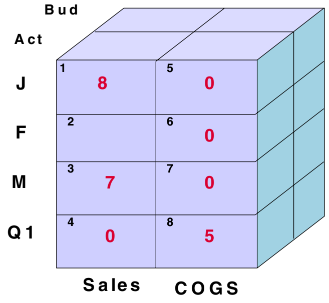
Compressed block size:

2 bytes + 24 bytes + 72 bytes = 98 bytes

Figure 31. Bitmap compression

RLE Compression

Time (4)
Accounts (2)
Scenario (2)



8,#MI,7, 0, 0, 0, 0, 5,#MI, #MI, #MI, #MI, #MI, #MI,#MI, #MI

Compresses to 10 cells:
8, #MI, 7, three cells, 5, three cells

Eight bytes per cell
72 bytes of block overhead

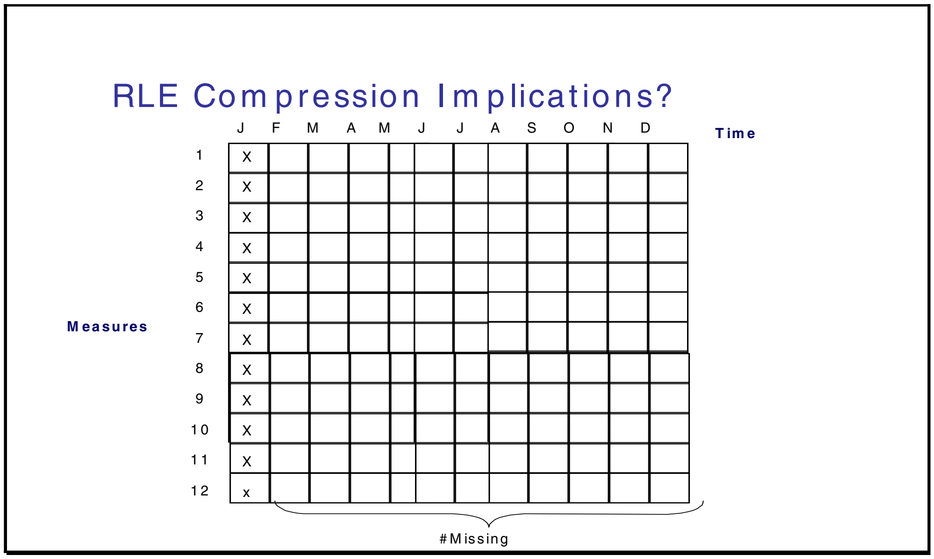
Compressed block size:
80 bytes + 72 bytes = 152 bytes

Sparse data blocks are filled with #Missing and RLE compression is more effective than bitmap compressing blocks with repeating values.

Figure 32. RLE compression

4.7.1 RLE compression and array declaration

Figure 33 and Figure 34 attempt to demonstrate graphically how the order of dense dimensions in the array might affect RLE compression. In the next diagram you can visualize why every individual row has to be mapped by the algorithm. Each row is identically mapped by the value (X) and #Missing across subsequent months.

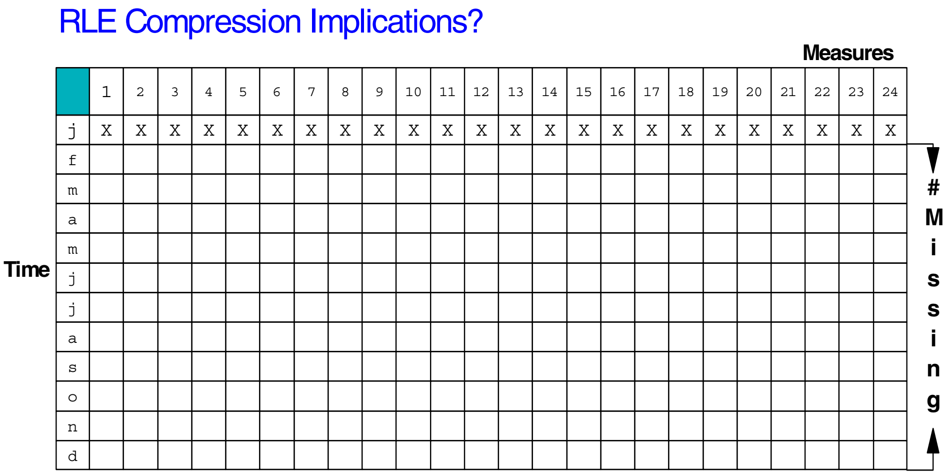


RLE Compression Implications?

	J	F	M	A	M	J	J	A	S	O	N	D
1	X											
2	X											
3	X											
4	X											
5	X											
6	X											
7	X											
8	X											
9	X											
10	X											
11	X											
12	x											

Figure 33. RLE compression Implications (1)

Note that altering the order of the dense dimensions creates an opportunity to capitalize on the nature of RLE compression. In the next declaration only the first row of cells needs to be mapped and all subsequent rows can be compressed to a RLE single statement: Beginning address, ending address and #Missing.



RLE Compression Implications?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
j	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
f																								
m																								
a																								
m																								
j																								
j																								
a																								
s																								
o																								
n																								
d																								

Figure 34. RLE compression implications (2)

In the final analysis, the compression algorithm that achieves the most compression should be utilized. How the dense dimensions are declared in the outline will potentially contribute to the effectiveness of RLE compression.

4.8 Using SET MSG ONLY

SET MSG ONLY is a completely *undocumented and unsupported* command that has grown to have a number of very significant uses within the consulting community that including a discussion of it here is almost mandatory.

4.8.1 How is SET MSG ONLY used?

Level-0 data is loaded to a completely empty database. Then a calculation of the database is initiated using the following script:

```
SET MSG ONLY;
CALC ALL;
```

Note that this command will generate a syntax error if embedded within a calc script. This is normal and can be completely ignored.

Also please pay attention to the fact that its use involves a representative data load at level-0. *Reliable statistics reported after a SET MSG ONLY calculation are predicated on using data that is essentially identical to production data in terms of the number of data blocks that the data load generates***.** Read on to see why.

4.8.2 What does SET MSG ONLY do?

This command, when specified in a calculation script or as part of the database default calc, will initiate a false-calculation of the database. By false-calculation is meant the following: the DB2 OLAP Server engine will perform a calculation run through the database keeping track of the calculation process in terms of I/O (block creation and updates.) This information will be reported in the application log in similar fashion to that set by SET MSG SUMMARY, and from it you can deduce approximate number of data blocks that the database would have created as if you had actually calculated it.

SET MSG ONLY reports upper block creation *based on the nature of the data that was loaded*. So if you only load some data, or fabricated data, you will only generate some or fabricated upper blocks, or both! To be reliable and accurate, you need to load real data across all of the dimensions in the model.

Thus, SET MSG ONLY will enable you to get a very accurate estimation of the total number of blocks that a particular database (configuration) will generate in a very short amount of time compared to a real calculation. We write estimation because the algorithm will not take into consideration member formulae (or by calc scripts) at all. If the model has formulae that create data blocks, then SET MSG ONLY will not report these, so the effective use of SET MSG ONLY on such models is limited by this factor.

4.8.3 SET MSG ONLY application log example using Sample::Basic

The following database statistics were registered in ESSCMD using the GETDBSTATS command:

```
Number of dimensions                                   : 11
Declared Block Size                                    : 12000
Actual Block Size                                      : 168
Declared Maximum Blocks                                : 594
Actual Maximum Blocks                                  : 513
Number of Non Missing Leaf Blocks                      : 177
Number of Non Missing Non Leaf Blocks                  : 0
Number of Total Blocks                                 : 177
Index Type                                             : B+ TREE
Average Block Density                                  : 92.57143
Average Sparse Density                                 : 34.50292
Block Compression Ratio                                : 0.9464407
```

Notice that the database has undergone a level-0 data load and has only 177 blocks.

A CALC ALL command was issued for this database preceded by a SET MSG SUMMARY statement. Here are the same statistics after a full calculation of the database:

```
   Number of dimensions                                   : 11
   Declared Block Size                                    : 12000
   Actual Block Size                                      : 168
   Declared Maximum Blocks                                : 594
   Actual Maximum Blocks                                  : 513
   Number of Non Missing Leaf Blocks                      : 177
   Number of Non Missing Non Leaf Blocks                  : 197
   Number of Total Blocks                                 : 374
   Index Type                                             : B+ TREE
   Average Block Density                                  : 92.85714
   Average Sparse Density                                 : 72.90448
   Block Compression Ratio                                : 0.9491525
```

The resulting application SET MSG SUMMARY log entry shows the results of the calculation:

```
Total Block Created: [1.9700e+002] Blocks
Sparse Calculations: [1.9700e+002] Writes and [6.3200e+002] Reads
Dense Calculations: [1.7700e+002] Writes and [1.7700e+002] Reads
Sparse Calculations: [3.3096e+004] Cells
Dense Calculations: [0.0000e+000] Cells
```

Notice that the Total Block Created: [1.9700e+002] Blocks is precisely the number of blocks reported in the database statistics as the Number of Non Missing Non Leaf Blocks as 197. Compare this log extract with that generated by a false-calculation (SET MSG ONLY) of the same database with the same data loaded. The database was reset and reloaded with level-0 data prior to running the calc script:

```
Total Potential blocks: [3.7400e+002] Blocks
Sparse Calculations: [1.9700e+002] Writes and [0.0000e+000] Reads
Dense Calculations: [1.7700e+002] Writes and [0.0000e+000] Reads
Sparse Calculations: [0.0000e+000] Cells
Dense Calculations: [0.0000e+000] Cells
```

It is important to note that the Total Potential blocks statistic in the application log is ***not*** the same as the Declared Maximum Blocks statistic returned by the GETDBSTATS command. Declared Maximum Blocks returns the total number of potential blocks as a result of taking the cross product of all sparse members of all sparse dimensions.

Notice the differences in output. The first line of the false-calculation log entry reads ‘Total Potential Blocks’ (374) and the real calculation log entry reads ‘Total Blocks Created’ (197.) Of significance is the fact that the number of *Sparse Calculations: Writes + Dense Calculation: Writes* (374) from this false-calculation is exactly equivalent to the total number of blocks generated by the real calculation (374.) Similarly, the Total Potential blocks metric is also identical to the actual number of data blocks generated by the database calculation (374.)

4.8.4 What can you do with SET MSG ONLY?

If you can generate a reliable statistic regarding the total number of blocks that a model will create without having to create them, you can do two really terrific things! In fact, initial development of the sparse/dense methodology and utility used the SET MSG ONLY calc to generate the total block count metric.

4.8.4.1 Estimating database sizing

Record the following statistics directly after a level-0 data load to your database: total number of blocks and database size (combined ess*.pag and ess*.ind files in bytes.) then divide the (total number of bytes) by the (number of loaded blocks) to get an estimate of the average number of bytes per compressed block on disk. Multiplying that figure by the total number of blocks gives you a very good approximation of the total size of you compressed database in bytes!

4.8.4.2 Estimating database sizing with limited data

By taking a sparse slice of the database (loading production data to the data across all sparse combinations) you can run a false-calculation to derive an approximation for total block-count. This is useful when access to your dense information is limited (because of data being time related and future information isn't available.)

Even with limited access to dense data, you can then generate a dense slice of your data by choosing a narrow set of sparse combinations (for example all data for 1 customer) and populating the database with all dense data for that sparse combination. We do this to ensure that the average block density is a more accurate estimation of real block density. From this you generate a much more accurate average block density and your average bytes per block will be more reflective of a production environment. Now you can create another accurate estimation of total database size.

You will have to factor in the fact that in version 7.1 the .pag files are extended in 8 megabyte increments. Additionally if you select to use the sparse/dense slice method, it will be necessary to factor in both the data block header and index entry overhead.

4.8.4.3 Estimating batch calculation time

Assuming that you now have a very good approximation of the total size of you database (from Database Sizing above) you can divide that number by the DB2 OLAP throughput metric of your system to generate an estimation of the total time to calculate the model. Note that we are assuming here that batch calculation throughput is linear. Empirical observations repeatedly show that as calculations approach their completion, throughput on the DB2 OLAP Server diminishes. This is especially evident when calculating very large databases containing many tens of millions of blocks

4.8.5 What you need to know about SET MSG ONLY

The SET MSG ONLY algorithm ideally works in conjunction with the calculator cache. So, if your database configuration achieves multiple bitmaps, the entire process will occur in RAM and will be very, very fast. However, if multiple bitmaps are not achieved during the false-calculation then the algorithm will keep track of block creation by writing completely empty data blocks out to disk.

This process is much slower than when multiple bitmaps are achieved, though still much faster than a real calculation. But this means that you will see both ess*.pag and ess*.ind files grow during (what has been described to as) a false-calculation! Do not despair. Ensure that you have the database compression set to RLE (remember, these are empty blocks – filled with #Missing, as it were) and the total disk space occupied will rapidly diminish as will the time the false-calculation takes to complete.

4.9 Intelligent calculation

In the scenario of periodic database updates, it remains a very good practice to make use of the intelligent calculation feature of the calculator engine.

Intelligent calculation takes advantage of the fact that dirty data blocks (that means data blocks that have been updated and need to be re-calculated) are tracked as part of the index structure. Periodic updates to the database can rely on the fact that the calculator engine will be able to isolate only those data blocks impacted by the update needs to be recalculated. Thus the calculator can *intelligently* select dirty data blocks to refresh.

Make note, however, that the use effectiveness of intelligent calculation is model dependent. Databases where periodic updates have an impact across the majority of *upper* blocks that already exist will not benefit from the use of intelligent calculation for very long. Keep in mind that existing upper data block are updated and a block update is twice as expensive, in terms of I/O, as data block creation.

Intelligent Calc needs to be managed. (For detailed information on intelligent calculation, see the *OLAP Database Administrator's Guide.*) It is not always intuitive or self-evident which blocks have been made dirty as a result of update activity. Additionally, the behavior of intelligent calc is different when used in conjunction with CALC ALL versus CALC DIM or AGG.

You can determine in advance of an actual calculation approximately how many data blocks will be impacted by additions to the database by using the undocumented SET MSG ONLY command.

The following is an log excerpt from a slightly modified version of the SAMPLE::BASIC database. New data blocks were created when the database was updated from a spreadsheet. The update creates 2 new data blocks and updates 10 upper blocks. The database prior to the lock and send was set clean using a calc script that contained the SET CLEARUPDATESTATUS ONLY command in combination with a CALC ALL command. A false-calculation was initiated using the following script:

```
SET MSG ONLY;
CALC ALL;
```

Here is what the OLAP Server recorded in the application log:

```
Total Potential blocks: [1.2000e+001] Blocks
Sparse Calculations: [1.0000e+001] Writes and [0.0000e+000] Reads
Dense Calculations: [2.0000e+000] Writes and [0.0000e+000] Reads
Sparse Calculations: [0.0000e+000] Cells
Dense Calculations: [0.0000e+000] Cells
```

Notice that the *Total Potential Blocks* figure is precisely the sum of *Sparse Calculations…Writes and Dense Calculations…Writes*. A full calculation (CALC ALL) of this database would result in a total of 12 blocks being updated or created. This is the sum of 10 upper blocks being created/updated plus the 2 blocks that were created from the lock and send being updated during the calculation.

From this example we can deduce that a CALC DIM or AGG of the sparse dimensions of this database would update/create 10 data blocks (i.e. Sparse Calculations: [1.0000e+001] Writes.) In this way you can relate the significance (total number of impacted data blocks) of a periodic update to the database. What you unfortunately cannot infer from this information is the number of new blocks that will be created versus the number of existing upper blocks that will be updated.

Actually performing the calculation results in the following log entry:

```
Total Block Created: [4.0000e+000] Blocks
Sparse Calculations: [1.0000e+001] Writes and [3.7000e+001] Reads
Dense Calculations: [2.0000e+000] Writes and [2.0000e+000] Reads
Sparse Calculations: [1.6800e+003] Cells
Dense Calculations: [0.0000e+000] Cells
```

The first line relates the number of blocks that were created (=4.) By subtracting this number from SET MSG ONLY, *Total Potential Blocks* figure (=12-4) you can determine (approximately) how many data blocks are updated (=8) as a result of the database incremental update and calculation. When the number of updated blocks approaches 50% of existing data blocks you can expect Intelligent Calculation to take at least as long as a complete level-0 data load and recalculation to the (empty) database. This is so because data block updates are twice as expensive in terms of I/O as data block creation, it would take the same amount of time to update 50 blocks as it would to create 100).

4.10 Miscellaneous issues

This section covers some additional implementation design issues not previously discussed:

- Partitioning tips and strategies
- The application log
- Data load optimization

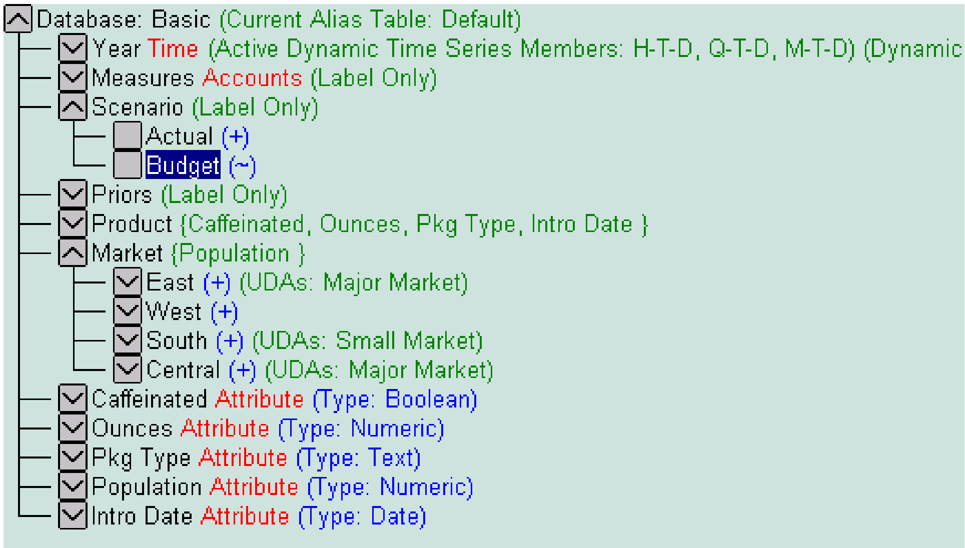
4.10.1 Partitioning tips and strategies

Database partitioning can be a most effective strategy to help scale very large OLAP models. In general, you partition to isolate into separate databases those parts of the large model that have different calculation/processing requirements. These different parts could be differentiated by how often calculation takes place, or how complex the calculation is, or even whether a part of the model is set read-write versus read-only. Thus partitioning will prevent your entire database from calculating at the lowest common denominator.

Not only does partitioning portion database size, it enables the spanning of database calculation requirements in parallel across multiple CPUs. In the final analysis, the success of a partitioning strategy resides chiefly in the architecture of the partition definitions. This is another way of saying that you must consider the database configuration carefully when implementing partitions.

4.10.1.1 Partition across a non-additive dimension

The main challenge that partitioning can be used to resolve is that of overall database size, or database scaling. When choosing a dimension to partition across, it is always recommended that a sparse dimension be chosen. However, if there are values that will then exist across the partition definition, the solution begs the question once again. Let us explore this in detail.



```
Database: Basic (Current Alias Table: Default)
  Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D, M-T-D) (Dynamic
  Measures Accounts (Label Only)
  Scenario (Label Only)
    Actual (+)
    Budget (~)
  Priors (Label Only)
  Product {Caffeinated, Ounces, Pkg Type, Intro Date }
  Market {Population }
    East (+) (UDAs: Major Market)
    West (+)
    South (+) (UDAs: Small Market)
    Central (+) (UDAs: Major Market)
  Caffeinated Attribute (Type: Boolean)
  Ounces Attribute (Type: Numeric)
  Pkg Type Attribute (Type: Text)
  Population Attribute (Type: Numeric)
  Intro Date Attribute (Type: Date)
```

Figure 35. Partitioning on Market and Scenario

Two dimensions are being presented as partitioning strategies: Market and Scenario (See Figure 35).

If I chose Market, I would create four source databases, one for each of my Regions (East, West, North, South) and implement a transparent partition on Market at a target database. I will have 4 small databases to update and calculate and I can now achieve this in parallel across multiple CPUs. The dilemma then exists how to generate data for total Market, which values exist only across the partition definition as the sum of East, West, and so on?

These values have to be generated. Recall that the number of blocks potentially involved during retrieval is potentially the Cartesian product of data blocks across all sparse dimensions from the source databases. Thus it is not always feasible, as might be suggested above, simply to make the Market member Dynamic in the target partition because of the potential I/O that would be generated on the server. This is usually the case in large databases (the reader is asked to imagine that Sample::Basic is a large database!)

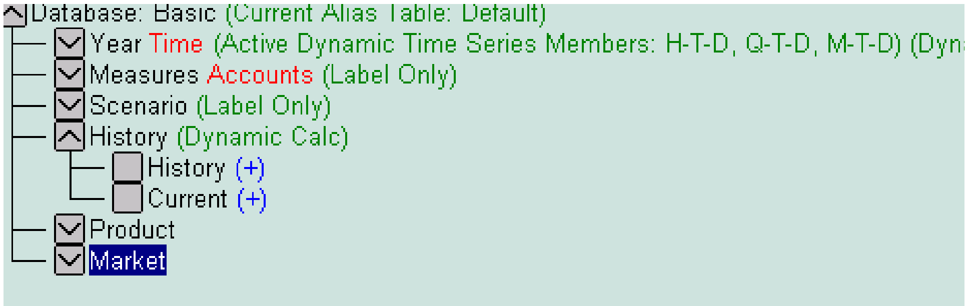
If we consider partitioning across the Scenario dimension, a much better picture emerges. The concept of Total Scenario does not make any business sense. Hence there are no values that need to be generated across this partition definition. The source database infrastructure can now be configured to seamlessly and effortlessly feed data to the target.

4.10.1.2 Partitioning on a dense dimension

If we consider more closely the previous hypothetical example, we will realize that the Scenario dimension properly adheres to the dense nature of the dataset. So the partitioning strategy overrides our database sparse/dense optimized configuration and we tag the Scenario dimension as sparse to accommodate. But, what happens when we (apparently) absolutely have to partition across a dense dimension?

Consider the situation where a model is periodically updated across time and where tagging time as sparse explodes the number of data blocks to forbidding proportions. This makes it impossible to update the database periodically and calculate it in time for user interaction. The dilemma can be resolved by the following strategy.

Consider the outline in Figure 36.



```
Database: Basic (Current Alias Table: Default)
├── Year Time (Active Dynamic Time Series Members: H-T-D, Q-T-D, M-T-D) (Dyn
├── Measures Accounts (Label Only)
├── Scenario (Label Only)
├── History (Dynamic Calc)
│   ├── History (+)
│   └── Current (+)
├── Product
└── Market
```

Figure 36. Outline example: partitioning on History

The History dimension has been inserted as a sparse dimension whose parent member is tagged dynamic (or Dynamic Calc and Store.) Now we can create two source databases each with identical dimensionality. The idea to add a dimension to enhance other dimensions is a clever idea and another example of such a technique is discussed in Section 4.4.1.3, “When to add a dimension” on page 50.

The partition definition is created across the History dimension. Periodic data is loaded to the Current member for the database. Values for total History are dynamically derived at query time. Procedurally the data from the Current database needs eventually to be moved to the larger History database prior to the next periodic update.

This strategy effectively eliminates the need to tag the time dimension sparse and enables the periodic updates to be available to users in a timely fashion. Moreover and just as significant is the fact that this strategy effectively increases the batch window for moving the data from the Current to the History cube to be just shy of the time in between periodic updates. If this were a weekly refresh model, the administrator would have almost seven days to move the last update of the Current database to the History database.

Consider a variation of this strategy where the periodic refreshes to the Current database occur on an hourly basis. Well, a Current model could be created the upper values of which are all dynamic! A transparent partition between it and the History database would make these values available. This would enable very nearly real-time data analysis for users. In order to keep the performance characteristics of the partitioning strategy optimal, the data from the Current model would need to be ported to the History model almost certainly on a nightly basis.

Dynamics on sparse members?

The above hypothetical example breaks the rule of setting dynamic members established above as best practice. It sets dynamic member(s) across a partition for crying out loud!

Well, this observation begs the recollection of the statement also made earlier that ‘database implementers alone understand sufficiently all of the criteria by which to determine that database performance is within acceptable limits.’ Whether the strategies described above will work as a practical partitioning implementation will depend upon a weighing of the differences between the overhead of batch processing periodic updates across databases where time has been tagged sparse, and that of performing dynamic calculations across a database partition at runtime.

The rules for tuning and optimization are not cut and dry, they really are conformed to the demands of the particular environment and user community that they are in service of.

4.10.2 The application log

It does bear repeating the importance of regularly reviewing the application log. For example, the event log records how the DB2 OLAP server is responding to user requests (through extraction or spreadsheet retrieval factor entries.) Comparing the retrieval performance characteristics of the spreadsheet add-in to those generated by a different query tool can quickly help to pinpoint trouble spots.

Many events that often go unnoticed are recorded in the application event log. Database performance problems can be isolated in an instant as a result of reviewing activity recorded there. Consider these examples:

```
[Fri Jan 05 14:54:28 2001]Local/Sample///Info(1012710)
Essbase needs to retrieve [1066] Essbase Kernel blocks in order to
calculate the top dynamically-calculated block.
```

This log entry tells me that, as a result of Dynamic Calc member tags on sparse members, the server will need to create 1066 blocks in order to compute the top block of the cube. From this I can infer the amount of block creation I/O that might be generated by users at retrieval time and I might want to adjust these tags accordingly.

A different log entry indicates that I have set dynamic calculations on every dimension, and that I have a Dynamic Calc and Store tag on a dense member:

```
[Fri Jan 05 14:59:31 2001]Local/Sample/Basicx/me/Info(1007125)
The number of Dynamic Calc Non-Store Members = [8 5 2 19 25 ]

[Fri Jan 05 14:59:31 2001]Local/Sample/Basicx/me/Info(1007126)
The number of Dynamic Calc Store Members = [0 1 0 0 0 ]
```

Whereas the former piece of information may be a design decision, the latter is very bad, and needs immediate correction. Dynamic Calc and Store tags on data block members do not reduce block size, but do force DB2 OLAP Server to ***update*** every block in the database that is retrieved when that particular member is requested.

Unless this is a tag made in error, the use of Dynamic Calc and Store tag usually reflects flawed logic. The flawed logic reasons that member should be tagged this way to make the value persistent after first retrieval because the metric is so popular with analysts. This has the undesired effect of increasing radically the number data block updates and results in a fragmented database.

4.10.2.1 Recording application logs events

A cube that is currently performing well may develop performance problems in the future as the outline is modified and the amount of data grows. If a cube starts performing poorly it is useful to know when the performance became worse and what changes were made just prior to this. Keeping a record of changes made to the outline as well as the time required to perform key tasks will help diagnose performance problems. The application log lists the actual time that was required to perform each operation. Since application logs become large and tend to be pruned regularly, you should save the elapsed times for tasks in a different file. A log file might contain the following information:

```
[Tue Jun 27 18:39:04 2000]Local/sales/Main/admin/Info(1003024)
Data Load Elapsed Time : [335.141] seconds

[Wed Jun 28 01:14:22 2000]Local/sales/Main/admin/Info(1012550)
Total Calc Elapsed Time : [3718.276] seconds
```

Table 13 is an example of the type of information that could be recorded. This table will make it clear if and when performance degrades. If there is a large increase in the time required for an operation that does not necessarily mean there is a problem. Perhaps a change was done that requires a lot of processing. For example, adding members to a dense dimension requires all blocks to be rebuilt. If there is a slow degradation in performance this may be due to the increasing size of the cube. It might be necessary to check the cache hit ratios to ensure the cube hasn't outgrown the cache settings.

Table 13. Recording application logs information

Date	Time	Action	Elapsed Time
2000-06-27	18:32	Building dimension	45.469
2000-06-27	18:39	Load	335.141
2000-06-27	18:45	Calc	3718.276
2000-06-30	14:00	Added members to Products dimension	
2000-07-02	18:12	Building dimension	48.123
2000-07-02	18:29	Load	337.532
2000-07-02	18:35	Calc	3723.533

4.10.3 Data load optimization

There is only one method to increase data load performance, sort the input data by sparse dimensions. Why this works is because by sorting the data across the sparse dimensions ensures that each data block that is being created as a result of the data load will be visited only a single time.

Consider the implications of a dataload file of the following random order characteristics shown in Figure 37.

New York	Old Fashioned	Actual	Sales	61	61	63	66	69	72	77	78	68	69	61	66
Florida	Diet Cream	Budget	Sales	80	80	80	80	80	80	80	50	50	20	30	40
Florida	Grape	Actual	Sales	80	80	81	85	89	105	110	114	87	94	70	75
Florida	Grape	Budget	Sales	80	80	80	90	90	110	110	120	90	100	70	70
Florida	Strawberry	Budget	Sales	80	120	130	130	140	150	150	160	130	120	80	80
Massachusetts	Grape	Actual	Sales	80	80	79	75	71	60	57	55	72	66	88	82
Massachusetts	Grape	Budget	Sales	80	80	80	80	70	60	60	50	70	70	90	80
Massachusetts	Strawberry	Actual	Sales	80	56	53	53	49	44	44	41	50	53	80	73
Massachusetts	Strawberry	Budget	Sales	80	60	50	50	50	40	40	40	50	50	80	70
Florida	Strawberry	Actual	Sales	81	115	121	121	130	144	144	154	126	118	78	85
Florida	Dark Cream	Budget	Sales	90	90	90	90	100	100	100	100	110	60	80	100
Massachusetts	Dark Cream	Budget	Sales	100	100	100	90	90	80	80	80	70	60	90	70
Florida	Diet Cream	Actual	Sales	110	110	112	103	110	110	115	69	76	51	52	56
Florida	Vanilla Cream	Budget	Sales	110	120	120	110	130	130	140	170	140	90	110	120
Florida	Dark Cream	Actual	Sales	120	118	120	127	130	133	133	140	149	126	121	145
Massachusetts	Old Fashioned	Budget	Sales	120	120	120	110	110	100	100	100	90	100	110	90
Massachusetts	Old Fashioned	Actual	Sales	126	128	125	117	114	111	111	105	98	116	120	99
Massachusetts	Dark Cream	Actual	Sales	130	132	129	121	118	115	115	109	102	120	124	103
New York	Old Fashioned	Actual	Sales	134	189	198	198	210	230	230	245	199	187	123	133

Figure 37. Load file with the first two 2 columns tagged sparse

The load file in Figure 37 is for the Sample::Basic database which has the first two columns (Market and Product) tagged sparse. The first data block that is created as a result of loading this data to an empty database is New York->Old Fashioned. Notice that this is also the address of the last (visible) record. Because the New York->Old Fashioned data block will already have been created during the load process, the OLAP Server will have to fetch from disk the existing block and update it. This results in two undesirable events. First, an update block is occurring during the data load in addition to the create block. For that data block, I/O has been trebled! Also, because DSB2 OLAP Server never does an update-in-place of data blocks, the database is being fragmented.

Now consider the implications of loading this same data file that has been sorted across the sparse dimensions as in Figure 38.

Florida	Cola	Actual	Sales	210	200	210	222	235	278	286	286	249	205	197	202
Florida	Cola	Budget	Sales	190	190	190	210	220	260	270	270	230	170	170	190
Florida	Dark Cream	Actual	Sales	120	118	120	127	130	133	133	140	149	126	121	145
Florida	Dark Cream	Budget	Sales	900	90	90	90	100	100	100	100	110	60	80	100
Florida	Diet Cola	Actual	Sales	200	206	214	267	273	282	336	277	230	218	245	262
Florida	Diet Cola	Budget	Sales	190	190	200	250	250	260	310	260	210	180	220	230
Florida	Diet Cream	Actual	Sales	110	110	112	103	110	110	115	69	76	51	52	56
Florida	Diet Cream	Budget	Sales	80	80	80	80	80	80	80	50	50	20	30	40
Florida	Grape	Actual	Sales	80	80	81	85	89	105	110	114	87	94	70	75
Florida	Grape	Budget	Sales	80	80	80	90	90	110	110	120	90	100	70	70
Florida	Strawberry	Actual	Sales	81	115	121	121	130	144	144	154	126	118	78	85
Florida	Strawberry	Budget	Sales	80	120	130	130	140	150	150	160	130	120	80	80
Florida	VanillaCream	Actual	Sales	150	154	155	151	170	177	189	226	182	174	164	177
Florida	VanillaCream	Budget	Sales	110	120	120	110	130	130	140	170	140	90	110	120
Massachusetts	Dark Cream	Actual	Sales	130	132	129	121	118	115	115	109	102	120	124	103
Massachusetts	Dark Cream	Budget	Sales	100	100	100	90	90	80	80	80	70	60	90	70
Massachusetts	Grape	Actual	Sales	80	80	79	75	71	60	57	55	72	66	88	82
Massachusetts	Grape	Budget	Sales	80	80	80	80	70	60	60	50	70	70	90	80
Massachusetts	Strawberry	Actual	Sales	80	56	53	53	49	44	44	41	50	53	80	73
Massachusetts	Strawberry	Budget	Sales	80	60	50	50	50	40	40	40	50	50	80	70
Massachusetts	Old Fashioned	Actual	Sales	126	128	125	117	114	111	111	105	98	116	120	99
Massachusetts	Old Fashioned	Budget	Sales	120	120	120	110	110	100	100	100	90	100	110	90
New York	Old Fashioned	Actual	Sales	134	189	198	198	210	230	230	245	199	187	123	133
New York	Old Fashioned	Actual	Sales	61	61	63	66	69	72	77	78	68	69	61	66

Figure 38. Load file sorted across sparse dimensions

Notice that now every sparse combination is loaded successively to the database meaning that each data block is touched only one time. This completely eliminates data block updates and database fragmentation!

4.10.4 Building a security model

IBM DB2 OLAP provides a comprehensive, multi-layered security system that administrators can use to control access to applications, databases and related objects.

A successful implementation can be measured in many ways including:

- Does the system fulfill the basic business requirements?
- Does the system grant appropriate access to the appropriate individuals?
- Is the system seamless and unobtrusive to the end user?
- Is the system maintainable?
- Is the security system portable in a multiple server environment?

The initial implementation of a security system or model is only a starting point. Throughout the life of a system, considerations relating to maintenance, automation and continued integrity take on much greater importance. Further still, a properly designed security system can be leveraged beyond its basic intent to help drive client application development in a secure and seamless manner.

The security model definition phase should not only seek to define objective access requirements, but can also take into account future custom application development. All security components can be accessed using the DB2 OLAP API within client applications developed in Visual Basic, C++ or Microsoft Excel. This means that custom applications can be developed that interpret and follow the defined security components, allowing the developer to tailor the contents of a client application to each individual user. This depends on the focus of the custom application, but may include things such as generating custom reports based on members as specified in a user's security filter profile.

4.10.4.1 Maintaining the security system

Security model definition and implementation are only part of the process of managing the security system. Security system maintenance is something that must be considered carefully to ensure that the original intent of the security model is sustained and that the maintenance of the individual security items does not become a burdensome and time consuming task.

All security items can obviously be managed in a manual fashion. This often is the typical course of action and may certainly be a viable option if the overall security system is relatively small. However, if an installation contains hundreds of users, dozens of applications and databases and requires many complex filters, the maintenance may become unmanageable. Additionally, manual maintenance of security items may likely introduce human error into the equation. Users may accidentally be granted the wrong access, or terminated employees may be left on the system.

The API is a viable alternative for developing maintenance applications that reduce the overall maintenance effort as well as increase the accuracy and integrity of the maintenance process.

The following are two examples that depict ideal situations for security maintenance automation using the API.

Keeping DB2 OLAP user list current

In this example, the system administrator implements an automated system to keep the list of DB2 OLAP system users current. We assume that the administrator has access to a master list of user IDs from the local area network.

Accessing central corporate resources such as the local area network user list will help the administrator keep current with such events as employee terminations, new hires or promotions to other departments. Terminated employees are automatically removed from the DB2 OLAP security system in accordance with the valid user profile list on the company's local area network system. It is not important what central corporate system or resource is used to validate the list of user profiles on the DB2 OLAP system. What matters is that the API can be used to accurately merge DB2 OLAP user profiles with other systems containing user profiles.

One important pre-condition is that the DB2 OLAP security model is developed with the same naming conventions as the master system that is being used for a comparison or that a mapping mechanism between the two lists be developed. Ideally the administrator would create user profiles in DB2 OLAP with the same user names as in the master system. This helps with automation in relation to keeping the DB2 OLAP user list current as well as simplifying the process for the end user who will not have to remember a separate user name for DB2 OLAP. To elevate the integrity of the security system, a verification log file would be generated to inform the administrator of deleted users.

Automating filter generation example

It is possible to leverage other systems within a corporation to automatically generate and maintain filters. For example, a security model that requires specified filter read or write access to a particular market requires both an initial setup as well as continued maintenance as employees change positions, leave the company or start new. We assume the requirement that employees in question are allowed to only view data within their particular market.

This information would be maintained in another corporate system, perhaps a DB2 table or some other department table. An API application would be developed to automate the process of reading the relationship between employee and market and implement this relationship using security filters within DB2 OLAP. The process can include creating the original filter assigned to user ids, as well as maintaining filters to respond to security events as they arise in the organization. This may include such actions as removing a filter when an employee no longer has the authority to view a market or changing a filter's contents to reflect an employee's reassignment to a different market. The ability to add and delete filters as well as change existing filter content is all provided through the API.

4.10.4.2 Managing security in multiple server environments

Multiple server environments are commonplace in the DB2 OLAP realm. Currently, the Application Manager does not administer or migrate security information across servers. This can be a significant problem with an extensive security system in place. Consider for example a multiple server installation having both a test server and a production server and the challenge of how to replicate the production server security system from the test server?

The automation of this requirement is accomplished utilizing the DB2 OLAP API. The API offers complete access to all DB2 OLAP security system components. User and group profiles as well as filter details can all be read and subsequently written. An API driven process can be written to read any and all security components from the production server and written to the test server.

4.11 Final comment

This chapter discussed at some length the concept of the data block and index storage structures. Every other section and sub-section in the chapter is only properly conceived when understood in the context of these storage structures. Only after the notion of the data block and index structures are well understood can effective database tuning be performed.

Chapter 5. Interviews and experiences

Most of the expertise and valuable knowledge on DB2 OLAP server belongs to people who implement DB2 OLAP server.

Our best implementers are customers and partners who are instrumental in figuring out the best way to design OLAP databases to satisfy user requirements and to put in production OLAP databases facing exploitation constraints.

When implementing OLAP applications, multiple manners often exist. To reflect practicality and multiplicity of knowledge, we address implementation and deployment of OLAP applications through different interviews we carried out by customers and partners.

5.1 Introduction

We asked our various customers and partner organizations to highlight their best practices on designing, implementing and deploying OLAP applications. This chapter details our findings.

The following interview questions are cross-referenced to the different individual interviews presented as a whole.

These interviews are based on the same frame and consist of the following questions:

- Describe your environment.
- What design approaches and design options did you experiment?
- What client tools do you use?
- What are the administrative processes you use?
- How do you manage the operations?

Each interview focuses mostly on a specific area, as shown in Table 14.

Table 14. Interviews

	Environment	Design	Client tool	Administration	Operation
Steve Beier’s interview	x	x		x	x
George Trudel’s interview	x	x	x		x
Mark Rich’s interview	x	x		x	
Joe Scovell’s and Jacques Chenot’s interviews	x	x	x	x	x
Anonymous person’s interview	x	x			x
Rich Semetulskis’ and Alan Farkas’ interview	x				x
Aster Hupkes’ interview		x			

5.2 Interview results

In this section we describe the results of our interviews.

5.2.1 Steve Beier's interview

Steve Beier (**SB**) is senior IT specialist and he has been working with IBM Global Services for 5 years. He currently works for IBM Storage Division on an enterprise information system. They utilize DB2 UDB, Essbase, DB2 OLAP, Visual Warehouse, AIX and some "home-grown" tools to build and maintain the project. His primary responsibilities include project architecture, technical lead and "renaissance" support.

5.2.1.1 Environment

1. Describe your OLAP environment.

   SB: "Our OLAP systems are all AIX running on RS/6000s. We chose AIX because of its flexibility, scalability, availability and manageability. All the applications we need, including DB2 for our data warehouse, run on AIX. Our choice for production would always be UNIX.

   Our production systems run on 2 RS/6000s. We run DB2 and DB2 OLAP on an S7A with 16G of RAM and 12 262 MHz processors with 8 MB of L2 cache. The extra level 2 cache is an important consideration. DB2 OLAP runs on the other production system, an S80 with 32G of RAM and 6 450 MHz processors. This is a very powerful machine, which we plan to upgrade to 12 processors in the future. The extra memory in this system allows us to tune the OLAP Server caches for best calculation performance. We try to allocate enough memory to the index caches to get an index cache hit rate of 1, which means we have the whole index cache in memory, which reduces I/O. We also tune the calculator cache using the algorithm described in the DBA guide. We allocate enough memory to allow the calculator to use multiple bitmaps, and we verify this by running the calculation with SET MSG SUMMARY. The latest model we tuned in this way went from a 4 hour calculation time to 40 minutes".

2. What about development and test?

   SB: "Alongside our production system we also run development systems and test systems. Ideally the development systems would match our production servers, but to keep costs down, we cascade old production servers to development, and old development servers to test. Getting the right hardware in place for OLAP is critical. Our production servers were under-powered for some time, which made our jobs very difficult. Obviously you don't want development and production to get in the way of each other. To do their jobs right, our customers need uninterrupted access to the production server, and our developers need an environment they control. We decided to include test servers so we can also test new versions of the product, run Beta code and try out new ideas without getting in the way of development or our users".

3. Where does your source data come from?

   SB: “A lot of our data now comes from SAP, but we also source data from other DB2 systems and from flat files. Until recently we also had to extract from VM. All this data goes through a sophisticated and highly automated Extract Transform and Load (ETL) process to get it into our DB2 Data Warehouse on the S7A. These processes are run using Visual Warehouse[1] and a custom system we developed ourselves. We then load directly from the warehouse into our OLAP cubes, using the SQL interface. All data transformation is done by DB2 before loading. We use very simple load rules, each rules file loads from a ‘load view’ in the Data Warehouse. This greatly improves maintenance and reuse because we know exactly where the data is from, and exactly what it looks like. The data definition language (DDL) for creating the ‘load views’ is stored and managed, so we can recreate the views very quickly. The SQL statements that select from the ‘load view’ always include an ORDER BY clause based on the sparse dimensions of the model. This loads the data block by block, reducing I/O and improving performance”.

4. What storage devices do you use, and how do you use them?

   SB: “We use SCSI devices for the operating system, and SSA devices for OLAP storage. We found RAID-1 mirroring is most effective and performs best for us. We allocate each set of page and index files to their own I/O channel”.

5. How many OLAP cubes are you supporting?

   SB: “We currently have 120 cubes in production. These range from a few MB to 8 GB in size, depending upon the users requirements.”

6. What applications are the cubes supporting?

   SB: “We have a diverse range of applications covering supply and demand management, finance, planning and forecasting. A lot of our users are using lock and send to update models and run scenarios.”

7. How many users do you support?

   SB: “We have a total of 250 users. Our average server load is about 20 concurrent users. At peak times the server will be handling 50 concurrent users”.

8. What is the makeup of your team?

   SB: “We have 20 people in total managing the entire data warehouse and the OLAP applications. Some of the critical skills are database administration, Essbase and DB2 OLAP Server specialists, support and

[1] Visual Warehouse is an IBM ETML tool now included in DB2 UDB V7 in its Data Warehouse Center function and its DB2 Warehouse Manager feature

helpdesk staff. A major part of our ongoing job is managing the nightly extract, warehouse load and OLAP build process. Two people are responsible for managing and monitoring this process every night. We have a number of programmers skilled in Java, ‘C’ and SQL who work on a variety of custom programs including the warehouse management processes and OLAP utilities. We also have several sophisticated power users that help with the design process.”

5.2.1.2 Design

1. Where do your OLAP requirements come from?

   SB: “Our business users drive the requirements. Our job is to meet their requirements, and to work with them to balance the requirements against the available resources.”

2. When developing a new OLAP model what techniques do you use?

   SB: “Planning and project management are very important. OLAP can be implemented very quickly when the requirements are understood and the project is well managed. The implementation can go so quickly that you start to think you don’t need planning and project management, but we have learned not to ignore these aspects of application development. Our power users are central to the design process. They understand the technology very well, so they usually take a first cut at the requirements using Application Manager to put together a sample outline. We work with them to refine the outline and determine the input data requirements. We build a prototype, and go through a number of iterations working closely with the users as we develop the cube. It’s important to have power users who understand the business in detail, and the technology in some depth. It’s also important that your OLAP specialists understand the technology in detail, and have some understanding of the business. Having your OLAP specialists and power users working well together makes a big difference.”

3. How do you validate your cubes?

   SB: “When a new cube goes into production, verification is part of the users responsibility prior to sign off. When cubes are in production we extract from the source system using SQL, and from the cubes using the APIs and automatically validate as much as possible. Some manual automation is also carried out using SQL programs and the spreadsheet. We validate every night as part of our batch process.”

4. What tuning steps do you go through as you develop an application?

   SB: “As we refine the prototype and move toward production we carry out load and calculation performance tests. We carry out final tuning when the cube goes into production, but there are design defaults and standards that we always use. These include: careful attention to data storage settings for dense and sparse dimensions, using the ‘hourglass’ outline layout, getting a block size between 20K and 60K, full dynamic calculation for dense members, and use of a calculation script that excludes dense dimensions from the calculation. We do not have formal checklists for tuning, but this is something I think we should do. We often adjust the model based on the user requirements. For example, we might remove some dynamic calculations in areas of the cube that get heavy use. We sometimes move formulae out of the outline and into a calculation script so we can get better control of calculation order. We have many cubes where we have had to make Time a sparse dimension because we have to clear and load time slices through the cube.”

5.2.1.3 Client tools

1. What client tools do you use?

   SB: “We use the spreadsheet add-in for interactive analysis, but we also run reports and charts that we capture as bitmaps and publish through Lotus Notes. The reports and charts are generated automatically using our own programs driving the spreadsheet program. We are looking at Web delivery in the near future, probably using Alphablocks.”

2. Who designs the reports that you deliver to your clients?

   SB: “Users request reports and changes to existing reports through our control system. Our team is responsible for generating the reports and making the changes.”

5.2.1.4 Administration

1. How do you manage outline updates?

   SB: “We incrementally build our dimensions from the data warehouse using load rules against load views. New members are added in a bucket that is part of the dimension. Our users are then responsible for moving the new members to the right place within the hierarchies.”

2. How do you manage load updates

   SB: “We carry out both complete cube rebuilds and incremental data loads. Cubes that are small, where the overhead is low, we clear the data and completely reload. For our larger cubes, and those where our users have entered data using lock and send, we use incremental loads. Some cubes provide a rolling time window, to manage these we have custom programs that are parameterized so we can drive them to clear the appropriate slice and load the new data into that part of the cube.”

3. How do you manage database fragmentation?

   SB: “We find it best to run a level 0 export, then reload and carry out a full calculation. We have tried dense restructuring, but we find it’s more efficient to export, run a simple default data load and calculation.

4. Do you use Application Partitioning?

   SB: “Yes, we have a number of applications that are partitioned. The best use we have found for partitioning is for high availability systems. We can load and calculate partitions in the background, and then use replication to refresh the main cube. We also use transparent partitions. We have not found a requirement for linked partitions yet. We do not use outline synchronization.”

5. How do you manage security?

   SB: “We have one person who is responsible for all aspects of security. As you can imagine, security is taken very seriously within IBM, so this is an important job, and you need the right person doing it. We have a custom support system written using Lotus Notes that our users access in order to request an ID, access to a database, or a change in security filters. This system routes the request to the user’s manager for sign off, and then to our security coordinator to be implemented. The system also helps us conduct audits and keep user IDs and authorities under control. We use all the available functions of the OLAP security system, and this is an area where we would really like the product to offer more such as improved granularity in the administration and management roles, password expiration rules, auditing and statistics. We are also using LumenSoft ServerManager to help with a number of security related activities that would otherwise be very time consuming and error prone manual processes.”

5.2.1.5 Operations

1. What steps to put a new application into production?

   SB: “We move the cube from the development system to the production system and test it in this environment. We then integrate the new application into the automation processes and test these. When we have the new application integrated, tuned and building as expected, we go through thorough end-to-end testing with the users. This includes validation of the input figures and calculations. When everything is right, we get sign off from the users”.

2. How do you manage backups?

   SB: “There are several strategies that can be adopted.

The first is to use the product’s ‘Begin Archive’ and ‘End Archive’ functions to take a ‘warm’ backup. These commands put an application into read only mode so the databases cannot be changed. Users can continue to query the cubes, but it’s safe to make a backup. This is the ideal solution, but we have found that the ‘Begin Archive’ command starts the application, even if it wasn’t in use. ‘End Archive’ doesn’t stop previously idle applications, so an automated process using these commands results in every application running, which uses up all our server resources. We could make the automatic process stop the applications, but if it’s an application that a user is busy with, and we stop it, then we have lost any advantages that this warm backup facility might offer.

The second strategy is to take ‘cold’ backups by stopping the server, taking the backup and starting the server when the backup is finished. We cannot use this approach because we have users around the world, so there’s no window of time when we can bring the server down for long enough to reliably complete a backup.

The final strategy, the one we use, is an automated process that unloads an application and takes a backup. We plan the unload and backup for times when we know the application is not in use. We run the risk that in extraordinary cases, a user may be using the application that we take off-line, but so far this hasn’t been a problem. We can backup each application as efficiently as possible with minimal impact on users.

We use IBM’s ADSM backup application to backup the entire install directory and the application data. We take incremental backups based on changes, so we minimize the amount of time required, and we can always recover anything from a single file right the way through to a complete OLAP system. Our backups run every day as part of our batch process. We can usually recover from problems with 4 hours. Ideally, we would like to simply backup outlines and level 0 data, but there are many other files that need managing, and there are often relationships between files in an application and other files in the system such as the CFG and SEC files. It quickly becomes impossible to recovery a complete application with such a simple strategy”

3. What’s your strategy for applying fixpacks or patches?

   SB: “Ideally, we would like to be able to choose to apply service once a quarter, when it best fits our schedule, but we haven’t been able to achieve this yet. We find we are forced to apply service quite frequently in order to get fixes we require. If possible, we do not apply the latest service level, we prefer to wait 2 months so we know it’s stable, but we often need a fix so we have less choice than we would like. Because of the frequency with which we apply service, we do not have time for a test cycle, so we install service directly onto our production server. We are looking at a product from Scapa technology that will allow us to automatically run sophisticated simulations of query workloads against the server so we can quickly check out the server and make sure we haven't introduced functional or performance problems."

a. What's your strategy for applying new releases?

   SB: "We are involved in as many Beta programs as we have time for, and we start looking at the new features and functions of a release as soon as it is available, but we will not put it into production for 6 months.

   We will put the new release on our test server and start by building our most important cubes. We then sample some of our other cubes, those that are interesting in some way, either because of the features they use, or because of their size. We look at a number of characteristics of the new release including load and calculation performance, storage requirements, and automation. We also plan to use Scapa to drive query workloads. Our users help with validation. We also test the entire end-to-end automation process using the new release. When the test and validation cycle is complete, we usually manage to upgrade the production server and refresh all cubes in one weekend.

b. How do you manage upgrades to client tools?

   SB: "We put the new tools on IBM's corporate install site ISSI and send e-mail notification to our users telling them that an upgrade is available. When we apply service, the upgrade is optional, when we upgrade to a new version, all users must upgrade their client code. The users download and install the code using ISSI."

4. What makes up your batch process?

   SB: "Our goal each week is to have a complete week without problems. When the systems are stable and have been running for some time we do achieve this. Obviously the more changes that are being made, the less stable the entire process is. We recently made some major updates to our data sources, and at the moment we are still working toward a 100% week. We hold regular team meetings to review problems, brainstorm solutions and track progress. These meetings are managed by some of our users, so they are involved in the overall process. They appreciate the issues we are facing, we get a better understanding of the impact of the problems and we can work together to resolve them."

5. How do you monitor the batch process?

   SB: “The automation programs that drive our batch processes are central to the way we manage the whole system. The programs send e-mail notifications of success or failure. We are fortunate that part of our team is based in Argentina, which is 5 hours ahead of us, so they get a head start on any problems.”

6. When do you validate the cubes during the batch process?

   SB: “Validation is built into our batch process. We don’t consider the process complete unless the validation is complete.

7. What batch window are you working with today?

   SB: “We have approximately 8 hours each night to carry out the Data Warehouse load and build all our cubes. The batch process is approximately 20% to 30% ETML (Extract, Transform, Modify, Load) and Warehouse load, and 70% to 80% OLAP load and calculations. We do as much as we can in parallel, but dependencies and resources limit this to a degree.”

5.2.1.6 Conclusion

1. What do you consider to be the world class things you are doing?

   SB: “The things we do well include our end-to-ends automation, which is very sophisticated, our user and security management infrastructure based on Lotus Notes and the implementation of standards in our data structures and outlines.”

2. What is the most important change you will make to improve your OLAP installation?

   SB: “Improving performance and reliability are top of our list.”

5.2.2 George Trudel’s interview

George Trudel (**GT**) is Director of Information Services at Scrip Pharmacy Solutions. He started working with multi-dimensional databases in 1990 and was the first business user to Beta test DB2OLAP. Scrip Pharmacy Solutions summarizes prescription transaction data as part of its knowledge delivery to its customers.

5.2.2.1 Environment

1. Describe your OLAP environment.

   GT: “Our current production environment on the server side is a DEC Alpha 4100 with 2G of memory and a 1.3 terabyte EMC symmetric storage box.I believe that 3 years ago the DEC Alpha was considered to be one or the top end servers. Performance was one of the key factors to choose a platform because of the overall anticipated size of the cubes. Today, there are no particular advantages to running in that environment and we are actively replacing the DEC Alpha with RS/6000 servers running on AIX.One of the key reasons for the change, is the fact that DEC Alpha is no longer supporting Windows NT and, as a result, Hyperion is no longer going to support its products on DEC Alpha. Since we knew we had to move, we went out and scouted around for the same type of platform performance. The RS/6000 came out on top for performance driven reasons.We run just one production server for all OLAP cubes. Currently this single production server runs four 500 megahertz processors. At anytime we have maybe ten to fifteen models available out of a pool of forty.”

2. What about development and test?

   GT: “We have a development environment which we run on a Windows NT server, 450 MHz, 25 G hard drive, 256M RAM. One of the issues that we have with this server is that it cannot replicate our large production models. So we may do some model development on the test server, but normally we load data into non-production databases on our production box. We do some of the testing on the development box, and some on the production machine”

3. Where does your source data come from?

   GT: “Well, our source data actually comes from two different places. Primarily we run Pharmacy prescription processing systems. As a Pharmacy Claims Processor, we run a proprietary package on an AS400. Typically, on a daily basis we take the day’s activity and download it into a relational database. Then we execute SQL programs that summarize the data and provide us with input files for our cubes. The relational output text files are loaded into the cubes. This process is a result of the evolution of the OLAP warehouse. We were originally getting source files from the AS400. We replaced that process with the source files from the Data Warehouse, and we haven't had a business case to move beyond that, into something like Integration Server".

4. What storage devices do you use, and how do you use them?

   GT: "We have an EMC Symmetrics 1.3 TB (terabyte) storage array. Physically this array contains 96 18 GB (gigabyte) drives.

   We have logically segmented those drives into multiple volumes. In support of OLAP, we have three volumes, one is about 500 GB, and two 300 GB. We don't use RAID or mirroring on that particular box.

5. How are you exploiting the multiple processors on the server?

   GT: "We run multiple calculations and database loads in parallel. We have business applications that are drilling into the production databases and at any point in time we can have anywhere from 10 to 15 models being actively used."

6. How many cubes are you supporting?

   GT: "We recently went through a regeneration process. The original thought process from our CIO, was to build a single large database, that incorporated all the dimensionality and everything that he ever wanted to see. Our production environment has evolved from that vision based on the sheer size of databases which caused us to segment information. Currently our models are broken up by regions. We split them out initially into three regions and further by time. Our current infrastructure contains six months of data in models for three different regions. The average database size is approximately 18 GB. Previously, database were over 30 GB.

   We have up to seven dimensions that we do not fully use due to the sheer size of the model. As a result of a business analysis, we remove a dimension for manufacturers since it really wasn't being used. When necessary, we can build a specific manufacturer model to meet the unique needs for that information because it is different from the way we look at the main OLAP core data. By pulling that dimension out of the production models, we able to reduce the overall size requirement by about 60% of our space utilization. We realized significant savings, which allows us to do some other things. That's on the normal reporting and analysis part of our responsibility.

   We also do ratings, for potential clients. We have a format that we send out that identifies the dimensionality and the measures that we use to construct rating models. The potential clients can fill out any portion of these templates. We run these files through our relational back end process, which generates all of the source information that we need for the cube. We use programming routines that automatically build these cubes. We can turn them around in a day to support the analyses performed by our ratings folks. Typically, the potential customer will provide us with some information about medication costs. We'll take that and run it through our models and do some comparisons with what we already know. This establishes a base for the rating process."

7. What applications are the cubes supporting?

   GT: "While we actually run the full mix of planning, budgeting, query and forecasting, primarily we provide, summarized information about the prescription process for our carrier-account-groups. The client dimension contains about 40,000 groups nationally.

   We summarize our medications across almost 200 different available measures that includes counts, dollars, and percentages relative to the prescriptions for members. We don't track member level detail in the cube. We summarize at the physician level. We are tracking 100,000 physicians for 40,000 groups, for over 15,000 medications."

8. How many users do you support?

   GT: "I would say that total number of users is 50, but 30 are active. One reason for this is because we do a lot of reporting for our clients. We have 10 clients who subscribe to our on-line service. However, we also produce about 1000 executive summaries each quarter for our customers. Additionally, we provide many of our customers with detailed files from the AS400 and from the relational database.

   The executive summaries go though in an automated batch process that was recently implemented by Applied OLAP. They provide a front end for Essbase. This automated batch process is anticipated to save us about $30,000 in the first year. This is a considerable operational savings for us".

9. What is the makeup of the team?

   GT: "On the relational side, we have a developer and a DBA. The DBA actually splits between Oracle and the AS400. We have an 2 person OLAP team. I'm responsible for the overall management and OLAP design, while the Essbase developer manages all the maintenance and support processes along with some development such as ratings."

5.2.2.2 Design

1. Where do your OLAP requirements come from?

   GT: "We look at business first."

2. When developing a new OLAP model what techniques do you use?

   GT: “The first thing that we do in designing a new model is to sit down with the business users. I will conduct a facilitated session that is going to try to establish the business rules that will then in turn define the dimensionality.

   Typically we take a look at the measures and then determine the values of the measure. Then this measure is defined in terms of its business views and its dimensionality. Then we’ll get into a discussion on how dimensions interact, to see if there are any opportunities like the example I talked about before with the manufacturing dimension. In that analysis, we also found that manufacturers are linked specifically to low level drug numbers, so they can become attributes, which is another way of utilizing special features within the technology. We are trying to develop business ownership as we go through the analytical process. Once we’ve completed these analytical tasks, it’s a very straightforward process to generate sample files and use the Application Manager to build the dimensionality, load the model and provide the first proof of concept within a very short period of time. Since all the source data can either be put into some form of relational data or some form of text data, we have the expertise and know how to leverage the use of Essbase rules to take these source files and create models out of it very quickly

   Because we understood the business requirements and the dimensionality, we generate prototypes very quickly. We put theses prototypes in the hands of the business users and let them start using it.

3. How do you go from prototype to production model?

   GT: “There are two aspects of production that we take into account before releasing new models. First, we validate the accuracy of the information. We have extensive routines to ensure that our OLAP data matches other claims and financial data, so that our presentation to our customers is consistent and correct. The second aspect is administrative. We estimate files sizes and support resource requirements. Given our experience, our estimates are accurate.”

4. How do you validate your cubes?

   GT: “We do some automatic validation. We have balancing routines that start right at the AS/400 and go all the way down and finish up with VBA routines that go in and do daily and monthly validation. We balance to the penny across all dimensions before the cubes are release. This process is very stable. We don’t run into very many problems at all”

5. What tuning steps do you go through as you develop an application?

   GT: "Well, we do go through sparse/dense and block size, we set up for the index and data caches. We always make sure that the caches are configured appropriately for our server, so that we get the maximum benefit on calculations. Typically I follow the time honored DB2 OLAP construction for block size, that seems to work pretty well. One consideration in dealing with response time is the use of dynamic calculations. For example, we had a dynamic calculation that was taking a very long time because it was focused across subsections of each dimension. While there's the time and space savings on the one side, there are response time issues on the other side when you go to retrieve it. We have a trade off depending on the application

   One of the widely-accepted rules that we apply is to make all calculations within the dense blocks dynamic. Further, don't make any sparse dimension members dynamic.

   We also put formulae in the outline, which is more efficient than putting them into calc scripts."

6. Have you formalized any of this knowledge, in terms of standards and checklists or is it experienced based?

   GT: "I developed a knowledge base, on our intranet. I write up explanations of everything I know and everything that I learn. We are building up the knowledge base, with related documents and information."

7. How do you move on in a new model using the full dimensions?

   GT: "We have separate extracts for dimension builds and data loads. We do that in two different processes because the input source files for our monthly update total about 700 MB. Trying to run through 700 MB, to pick out and update the dimensionality is a very time consuming task. Instead s we built SQL routines that query the relational database to find new monthly entries that are put into a formatted input file. We use this file as part of the monthly process to update the hierarchies before we load the data. We are dealing with a much smaller hierarchy file, which is much more efficient.

   We make the relational database do as much of the data transformation work so as to keep our load rules very simple."

5.2.2.3 Client tools

1. What client tools do you use?

   GT: "We have done some VB/Essbase development with Applied OLAP. This is another one of our evolutionary processes that come out of these applications. We were generating reports for small group executive summaries and physician report cards. These are either a summary by client or by physician of the drugs that they've dispensed over a certain period of time, and on the physician report card, a comparison between that doctor and all the other doctors within that specialty field. These reports were being run off FoxPro.

One of the goals for the Warehouse Team was to move out of that structure. Consequently, about a year and a half ago, we started looking at a way to quickly develop an internal process. We put an application together using the spreadsheet toolkit and VBA macros. It took a couple of weeks to complete. However, it was costly to maintain and support because of the need for developers to hand-hold it, so we initiated another project to evolve it. At the same time, we were looking to see how we could get Excel-like functionality on the Web, including access to our cubes.

We found Applied OLAP could provide everything we needed. We started a project with them last March. In September, we started production delivery. The entire process was streamlined. Now, we have a front end tool that lets our users maintain the report specifications in an MS Access database. From there we generate the XML sheets, that run against the Applied OLAP custom reporting applications to produces the thousands of reports that we distribute. Our account managers enter account information about new clients, that automatically flow through the process. Production Control regenerates the XML to automatically update the retrieval list and then all the requests are automatically run in batch. To give our customers flexibility, we can print the reports, email them, or store them on a client-specific bulletin board. The electronic delivery consists of an Excel workbook. They can go into the workbook for additional functionality without having to re-key information from a printout.

Actually, customers have also a direct on-line access. We had contracted with Painted Word, in 1998. They constructed a Web-based, Java application that allowed external customers to come in and use an Excel look alike. It has limited Excel functionality, providing a spreadsheet grid. Once they navigate to a low enough level within the OLAP cube, they can pick a measure and drill down into the relational detail that generated the cube. With this application, we were quite a way ahead of the curve for this three years ago."

2. Who designs the reports that you deliver to your clients?

   GT: "Internal pharmacists and account managers designed the original reports. One of the ways we add value is to involve our clinical staff. They perform analysis, taking a look at the different views from a pharmacy or clinical perspective. In addition, we provide financial and claims processing reports. Some of these are standard outputs from our claims processing system, but generally the business users design the reports based on requests from the customers."

5.2.2.4 Administration

1. How do you manage security?

   GT: "I don't think it's trivial. Our issue is providing our clients with Web-based access through front end applications. We have to restrict their access within the OLAP cube.

   For example, some of our clients are third party administrators. They may have multiple accounts, who are competitors. We had to create a security scheme through our Web interface, that limits their ability to see other data. The issue within the OLAP Server is that if you go to an account and zoom out and go back to carrier and zoom in, you will see all accounts. Our solution within the Web interface was to introduce a relational table that handles security within the applications, limiting the database view by customer.

   Our OLAP security system is not integrated with other systems, but we are looking forward to the promised land of single sign-on."

5.2.2.5 Operations

1. What steps to put a new application into production?

   GT: "Within our application development process, we can deploy new applications within two weeks, and often, within one week. But, there are a lot of other issues that go into the project life cycle. Let's say we've gone through a two to three week development cycle.

   Most of the development is handled by our OLAP team. We are primarily responsible for the production side of the warehouse. We sit down and conduct a run-through. We will check it out first. Then, we have a couple of our business users look at it. We also update the automated balancing routines to verify the numbers, and check that the dollar amounts are what they should be. Usually, the only discernible way the users have of knowing that it's gone from prototype to production is that all of the data is there. We send out regular communications on the status of models and our production environment."

2. How do you manage backups?

   GT: "We use Veritas cartridge based tape backups. The backup process runs every weekend. Sometimes we overlap with the production run because some of our calculations take a long time. If a specific cube is calculating and is therefore active, it will not be backed up. We run that backup separately. This usually happens as we perform prior month updates in the first week of the month. We are conservative with our batch calculations at the moment, because of I/O issues we have with our current server.

Our systems testing on the AIX platform have been successful in running multiple calculations for large databases. In our current production environment, we run the big calculations sequentially and then run the smaller ones in parallel. If a model resides on a single logical drive we can calculate it in conjunction with a model that spans drives. Running multiple calculations or multiple models on spanned drives don’t work well with our Alpha server.”

3. Do you have any disaster recovery plans?

   GT: “We have offsite storage. We take periodic snapshots that are moved off site. The economics don’t work for full disaster recovery, but we should be able to get back into production in a couple of weeks if the worst happened.”

4. What’s your strategy for applying fixpacks or patches?

   GT: “Well, when forced to update, we do, but when we find a stable level that is working for us, we stay there. We run quite a way behind the latest patch level, when a certain level of software is working in production. There have to be some pretty dramatic enhancements for us to go through the effort of upgrading. We are converting at our current level, but will upgrade to the latest version and latest patch once the AIX production is stable.

   Unless there is some urgent call for function that we don’t have, we will not update until we see there is a real need.”

5. What’s your strategy for applying new releases?

   GT: “We always take a look at the new releases, and we usually find significant new function that makes the upgrade cost worthwhile. But, we do not rush out and upgrade. We wait for real requirements that demand some new function.”

6. How do you manage upgrades to client tools?

   GT: “Upgrading means delivering the Application Manager, delivering the spreadsheet client, and whatever else we have integrated in our front end applications and making sure that it all works, so for a small organization, it is a significant effort. We have lists of users and applications that support this process.”

7. What makes up your batch process?

   GT: "We have a two tier system, monthly and daily.

   We have segmented the detailed monthly models by region. Most of our internal processing doesn't require the same level of detail, so we built another model that contains all of the regions, all the clients, but doesn't have as much detail in other dimensions so that we can begin to do the monthly reporting. This database is usually ready on the second of the new month. Many times it's available on the first of the month. In addition, we have daily models that get consolidated into the monthly cubes.

   On the last day of the month, the daily processes run. Then the relational database runs a series of routines so that by mid morning on the first, we have all of the hierarchy files and data files delivered. We can then start our update process. Often, by mid afternoon we have delivered the internal model for review of monthly data.

   We are not getting a lot of pressure to update more frequently; between what we do in the monthly models and the daily models we are meeting demand. Any time that someone wants to go in and see if current month trend looks good compared to the previous month, they can do that with the daily models. Each day, claims that are processed through our system until midnight are moved into the relational database overnight. The cube source files are then formatted and delivered to us in the morning. We load them during the day and our calculations run the next night."

8. How do you automate the OLAP operations?

   GT: "The processing of the relational data, the extraction of the flat files, dimension builds, data loads are using automated software routines that get the data from the AS400, load it into the Data Warehouse and then deliver the flat files for the OLAP cubes. We have several files summarized different ways by the Data Warehouse. We have members, and within the member population are people who have prescriptions. We roll those up based on the number of people who use certain drugs, the number of people who utilize a certain place of service, for instance an ambulatory service, a long-term facility or a mail order facility. Then we also roll up by physician, so we can see which doctors are prescribing which drugs. That's another set of information, much smaller files, but it's another way that we have been able to incorporate another level of data into the OLAP model, another level of summary. We have update processes for the outline changes, that are largely manual. We have a product from Painted Word called Clockwork, an OLAP batch loader which runs the data loads and calculations. Within Clockwork, we have an automated start time for the nightly calculations."

5.2.2.6 Conclusion

1. What do you consider to be the world class things you are doing?

   GT: "First of all, the ability to bring in a number of diverse data sources and turn around and produce OLAP cubes that our internal folks can look at. Our Web interface, which gives customers the ability to get into the cubes and analyze the data using either a Java based user interface, with the drill into the relational database, or using our Excel based Web application. This gets us the point where the client only needs a browser to be able to get at the data that they need.

   On the operational side, we make it work every month. It's a complex production environment with large data volumes, but it's there, it's up, and we've had very few problems with down time. Our operational staff have realized that availability is very important. We are running in the high 90%'s."

2. What is the most important change you will make to improve your OLAP installation?

   GT: "We are going to provide the level of performance to the client that they are demanding in the Web world."

5.2.3 Mark Rich's interview

Mark Rich (**MR**) has been working as a financial and business analyst for 16 years in IBM. In 1998, he was brought on board to lead IBM's World Wide Planning System, currently based on IBM DB2 OLAP on IBM RS6000 technology (APEX project).

5.2.3.1 Environment

1. Describe your OLAP environment.

   MR: "We run AIX on SP2.

   AIX gives us scalability. On the RS/6000 platform we can keep adding processors, memory and disks, and the operating system supports high-end systems very well. Of course that is limited by the particular system that you have, which is where the SP2 cluster comes in.

   Being an IBM system, we have to follow very strict security guidelines and NT especially will not come close to being able to give us the security that we need, AIX has the support we need.

   Then we cluster the RS/6000 nodes into an SP2. With the SP2 we were able to take multiple nodes, machines that were being maintained around the world, and bring them into a single complex. This allows us to offer 24X7 support, with staff that control and manage the entire SP2 complex.

   We have a workstation that is capable of operating all the software, all security updates, all operating systems through a single point of entry within the SP2."

2. How are you exploiting the multiple processors?

   MR: "The SP2 itself is broken out into clusters of several RS/6000s for manageability. With this configuration we are able to run 24 X 7 X 365 for read capability, and unless we have problems, every one of our models is available for update just under 24 hours every day. We have a very small window, 10 or 15 minutes, when update isn't supported. This is dictated by the size of our cubes. We have to disengage one of the three mirrors to take a backup. Once a mirror is completely disengaged, we only have two mirrors, but we are writable again."

3. What do the nodes look like today, are they high-end nodes?

   MR: "The system has been in place about three years. The main cluster uses 604E nodes with 2-4G of memory with 215G of storage for OLAP cubes, which we triple mirror. We are building a brand new cluster using the latest technology with 375 or 400 MHz processors, which go up to 24 CPUs and up to 32G of memory and 400G of writable DASD per physical node."

4. How do you decide what runs on each node?

   MR: “We have a total of 28 nodes. Of these, 4 nodes are for backups, using HACMP capability. We have 2 nodes for development, which includes system integration testing by us and the development team and operations when we get new code. The other nodes are split onto 4 different production clusters, which are split out by IBM business group — a node for the Software Group, a node for the Server Group, and a node for the PC company. Then we have nodes for the geographies; for example, EMEA has 2 nodes. And for the corporate CHQ models we have 3 nodes.

   We carry out benchmarks to decide how we further allocate the nodes based on how much RAM is used for particular models, how much CPU or how many CPUs, and how many different functions you have running within the cube itself.

   So we balance across line of business and then further balance based on resource requirements, but we haven’t been able to balance as much as we want on resources. The ideal would result in 2 or 3 SP2s as opposed to what we have right now.

5. What about development and test?

   MR: “We now have 2 nodes in the SP2 complex, which we use for a variety of development and testing activities. Ideally we should not be developing within the production complex so we also have cascaded machines from older system. These are not managed within the SP2 complex but are used by the development teams, and we do most of our development there, building new models, and testing new function Beta programs”.

6. Where does your source data come from?

   MR: “The input data is measurements, matched with accounting data, which come from a Data Warehouse, but our models are largely for planning and budgeting so most of the data is created in the models by the analysts.”

7. How many cubes are you supporting?

   MR: “We have 490 cubes.

   The smallest cubes are in the 200 M to 300 M range. These are small and easy to work with. The largest cube right now is about 70 GB. We brought that down drastically from over 100 GB.”

a. Presumably bringing down the number of cubes will make the whole thing more manageable?

   MR: "Yes. As many CPUs as we have in the SP2, our biggest problem is still performance. If we reduce the number of cubes we can make better use of the available resources.

8. What applications are the cubes supporting?

   MR: "Every one of our cubes is financial. They are used for measurement reporting, the majority for IBM Financial Planning.

   One of our major tasks next year is to understand why we have so many cubes. We think we have a large number of models that are very similar that can be merged across divisions. We hope we can use Application Partitioning to have source cubes with four or five dimensions and have four or five cubes connected together through the exact same four or five dimensions and then have an additional dimension, which is unique.

   We want to align common cubes and hopefully reduce the number to something like 200.

9. How many users are you supporting?

   MR: "We put a limit on each node as to how many people can connect. On 10 of our nodes we have a limit of 100 concurrent users. On another 10 nodes the limit is from 100 to 175 concurrent users. We have one node that goes up to 250 concurrent users. We are not limiting users per model, we are limiting per node.

   Our user population is 6000, worldwide. Within the cluster we can support 1675 concurrent users."

10. What is the makeup of your team?

    MR: "Our development staff is currently about 10 people. The core development team is gradually getting smaller as our base of power users increases. Some of the satellite areas like Software Group and Global Services have their development staff of 2 or 3 people each. They do development for their own models and only come to us when there is an issue.

    Of our 10 developers, we have 2 people that have very little OLAP knowledge. They are developing tools around the product. They understand the APIs and programming, but they do not build models.

    We have 4 people working on system maintenance and support, 2 that I mentioned before for OLAP API development, 1 for associated development, and 3 really good OLAP modeling people.

So we have a strong core team, but there are also model designers and strong model developers distributed across the various business units.”

a. Is the data warehouse your responsibility as well?

   MR: “The data warehouse is another groups responsibility, but once it’s in our models, it’s our responsibility.”

b. How’s your relationship with the warehousing group, does it work well?

   MR: “It works well, but the reality is, as with any group, they have totally different priorities than we do. They work with us to help us work efficiently with the data, but the priorities differ.”

5.2.3.2 Design

1. Where do your OLAP requirements come from?

   MR: “A lot of the OLAP requirements are coming in from the business users, from the line of business itself. Presumably my team is often involved with people from the line of business who are new to OLAP who want 27 dimensions and the nth level of detail. The toughest thing to deal with is getting the person to think multi-dimensionally, even the power users and the developers now, when they start thinking about a model like an end user would, it gets very tricky to understand what the dimensions are. Once we have found the right set of dimensions, it’s pretty easy.”

   a. Do you rely on your power users, your developers and designers to get the dimensionality right?

      MR: “A good starting point is an existing model. Then they come in with an additional request for information that they want added at a certain point within the model. That’s how the requests usually start.”

2. When developing a new OLAP model what techniques do you use?

   MR: “A good starting point is an existing model. Then power users come in with an additional request for information that they want added at a certain point within the model. That’s how the requests usually start.”

   We like to give them an understanding of what we are trying to convey to them and vice versa. What we spend most of the time on is what it will like in the Lotus spreadsheet and how they’ll pull their information.”

   a. So you very quickly build a prototype of the spreadsheet and let them see it in Lotus 1-2-3?

      MR: “Yes. From there we quickly build a prototype outline and let them start slicing and dicing.”

b. So if I want a new model, we work together and come up with a design that we think is reasonable. Do you then go through any sort of estimating process, or do you just go ahead and build the cube?

   MR: *"*Unfortunately we have not found a way to get good estimates. We use the standards that Hyperion has given us over time and the knowledge that we have evolved through development and we just take it from there."

c. When you are going through this first model build, what are the things at the top of your list in terms of technical settings. Are you most worried about block size then sparse/dense, how detailed does that list get?

   MR: *"*We worry about identifying the dimensions. Then we take a look at how this would change and how often and factor that into our choice of dense and sparse. We usually don't have too much flexibility. We have some models with block size over 300K."

d. Do you use dynamic calculation?

   MR: *"*We do, but we will only do dynamic within the block. We know you can also use dynamic calculation on sparse dimensions, but it impacts retrieval performance. The power users work with the end users to get the balance right and keep retrieval times acceptable to the individual end users."

e. Do you have this sort of thing formalized in a checklist?

   MR: *"*Unfortunately we have very little that is formalized. When a model is done, we don't review it to make sure it's efficient for the group. Whoever worked on the model is responsible for it. We really don't have anything in writing, other than what has been distributed from communications with the vendors and through technical meetings.

f. So you are relying on the body of knowledge that your people have built up?

   MR: *"*Yes. We still feel that building a cube is more of an art than it is a science, and we haven't been able to formalize very much.

g. When a new model has been through the design process do you do anything to test query response time or is that down to the users?

   MR: *"*We do accept this task when it's given to us, but our job is more about managing the production side. We just make sure that we can deliver the model, the rest is up to the power users. Remember that we have 490 models in production, and the models change. The power users have database administrator authority and there is a supervisor for each node. They modify the cube over time, we don't go back and review them on a regular basis. Once we have developed the models and put them into production, we don't go back to them."

3. How do you go from prototype to production model? On average how long does it take you to develop and deploy a new model and get it into production?

   MR: *"*With a new model that is based on an existing outline, say 75% reuse, we can do something as fast as a week. We prefer to take more time, but 4 weeks is probably the maximum."

4. How do you validate your cubes?

   MR: *"*For the models that we load data into, we validate the input data. For the pure planning models the input is from the analysts, and the power users are responsible for validating these models. Before any planning scenario or planning cycle can be closed it has to be validated. This is down to the planners, analysts and power users. Then the IBM financial community verifies the data within the model." We don't have any validation process, because it is managed by the analysts and the financial community within IBM."

5.2.3.3 Client tools

1. What client tools do you use?

   MR: *"*All of our users are running Lotus 1-2-3 with the spreadsheet add-in. We are also doing some work with Alphablocks and their new Spreadsheet Box. Moving to the Web is obviously very interesting, and the ability to assemble Web pages that shows multiple cubes and other data sources within a single page."

2. Do you publish reports from your data?

   MR: *"*We do not have an EIS type interface to our systems. Our user community is made up of financial analysts and planners rather than executives."

5.2.3.4 Administration

1. How do you manage load updates?

   MR: *"*Some of the cubes we work on are a rolling twelve month time period and we may have a major change over at the end of the year where we have to reset everything and start again. The good news is, other than 20 models or so, we have very little involvement from the development side, the supervisors manage the changeover process. In many cases, the best way to set up for the new period is to make a copy of the current model. This automatically copies all the latest changes. The latest inputs from the analysts, the latest security filters and so on. Then they can update the model and make it ready for the new financial period. With so many large models, this really puts our resources under pressure. We audit model use, and storage use and try to be as resource efficient as possible. During the switch we are under a lot of pressure because users want to run two sets of models for a while.

2. How do you manage security?

   MR: *"*With over 6000 users, security is a real issue for us. For example, when we open a new financial year we have to create many new models, and each of these should have the same complex security model as the previous year, including all the filters and rules that developed throughout the year. We have to move the whole thing from model to model. The power users and supervisors spend a lot of time on security and administration, which includes security.

   We are working on an internal system that should help. Each user will be uniquely identified within IBM, and we will maintain security for these IDs across the cubes, the report pool on the Web, for Application Manager functions and the spreadsheet add-in. They will have one password to manage for all access."

   a. So you are moving towards a single sign on?

      MR: *"*Yes, that's what we want to do."

   b. Building filters, adding people to groups removing people when they leave, all this security administration is done by the power users who own each of the nodes, is that right?

      MR: *"*Yes"

   c. Do you have any integration of this security system with other security systems?

      MR: *"*Yes. The CHQ system is integrated with the Data Warehouse for financial accounting data, so people request access through the center that manages access to all of IBM's accounting data worldwide. This is a highly secure system. Access to other models we can do ourselves, but through the proper channels."

   d. Are the actual updates to the OLAP security system automated?

      MR: *"*No, unfortunately it's a manual process for us."

5.2.3.5 Operations

1. How do you manage backups?

   MR: *"*We use triple mirroring, and we take one mirror off line to do a backup. We use ADSM. It runs on one node, so each cluster has its own ADSM setup. The backups run as part of the batch process, nightly, but because we have clusters that support different geographies around the world we backup according to geography. The backup for Asia Pacific runs through our lunchtime, the European backup runs around 7PM."

   We use the Begin Archive function to put the model in read only mode, then we take the mirror off-line, run the archive, put the mirror back on-line and issue the End Archive command.

   a. How granular is the back up, do you just backup the data?

      MR: *"*We backup everything that is written to disk other than the operating system. The system is We can override this too, for example we have some history models which don't change. We can take a one time backup for these, and that's pretty much it."

   b. Do you take incremental backups?

      MR: *"*No, we need to be able to recover very quickly even if we lose an entire node, so we have a complete backup that we could even reload to another node within the SP2 complex. If we lose a disk, we have triple mirroring. If we lost several disks at the same time in several places, we would have to go to the tape backup, but we haven't had that issue yet. With HACMP, if we lose the node because of a hardware failure then HACMP is ready to take over."

   c. So, if you loose a whole node, how quickly could you get the models back?

      MR: *"*If we loose the whole node and we didn't lose the DASD, your model could be back up and you wouldn't even know it. HACMP would simply take over. But, if we lost the DASD or if you have a database corruption and you have to reload that cube from tape it may take up to 3 or 4 hours for the very large models, 20 to 30 minutes for small model, tape speed is the limitation."

2. Do you have any disaster recovery plans in place for OLAP?

   MR: *"*We did investigate it, but it's not cost effective at this point in time."

3. What’s your strategy for applying fixpacks or patches?

   MR: “My ideal would be an upgrade every six months and no more. We haven’t applied service for over six months now, and we probably won’t until the next release. Six months ago we were upgrading every single patch because of fixes that we really needed. We get to the point where we understand the limitations of the current level and what we need to work around, and we stay there if possible. We aren’t exploiting some major pieces of functionality because of this, for example, we have tried to use Application Partitioning, but we found problems with it. The problems were fixed in a later patch, but by then we had a stable system which we didn’t want to put at risk. We look at major releases very early, including running the Beta releases. We have to decide if there’s enough new functionality to justify the upgrade including the system integration test cycle, which can take up to 2 months, and then updates for all the production nodes and 6000 users. If we decide the upgrade makes sense, then we go through the process of testing until we find a level that we can stabilize on. Ideally, we would have a new release in production in 3 months, but we have not managed this yet. Finding the right combination of functionality and stability has taken up to 8 months for us. “

4. What tools do you use to run your systems?

   MR: “It’s literally down to code we have developed around AIX facilities for scheduling and managing processes. We use Tivoli, and ADSM, and customer code to tie it all together.”

5. How do you monitor the batch process?

   MR: “We have 24x7 operational support, so there is always someone available who will receive a message if things start to go wrong. I get notified if the backup fails or aborts for the corporate models. I get daily status reports that let me know how everything is running, and what needed attention.”

   a. You said your goal was to be 24x7x365, what do you think you are actually achieving?

      MR: “For read capability, we’re there now. Other than problems, we are actually there.”

   b. How often are you losing time because of problems?

      MR: “I would say probably say we take a hit once a month. Sometimes it’s more because of particular software issues or design issues, for example when we tried to use Application Partitioning for one set of models we had to recycle a node every day for 8 days until we worked out a particular problem.”

5.2.3.6 Conclusion

1. What do you consider to be the world class things you are doing?

   MR: “Everything that we do with the SP2. The whole operation in my opinion is state of the art but it took us time to get there, and we had to build a lot of skills along the way. We are also a great example of OLAP large-scale enterprise OLAP for planning, forecasting and budgeting. Our models are not for reporting, they are for analysis, planning and forecasting for one of the largest corporations in the world. We have 6000 users, that’s not 6000 people looking at reports, it’s 6000 users who are updating the data and running scenarios.”

2. What is the most important change you will make to improve your OLAP installation?

   MR: “We will be deploying a new release of the OLAP Server, and we will be updating our hardware. We would also like to deploy Integration Server. It would help a lot with model building from our Data Warehouse.”

5.2.4 Joe Scovell's and Jacques Chenot's interviews

Joe Scovell and Jacques Chenot (**JS&JC**) are working at DST Systems Inc., of Kansas City, a Transfer Agent in the Mutual Fund Industry. Joe Scovell is a Client Services Manager. He has 13 years experience working with many diverse Mutual Fund clients. His area of expertise centers around providing clients value-added services that enable them to intelligently mine their data.

5.2.4.1 Environment

1. Describe your OLAP environment.

   JS&JC: "We are running Windows NT on our two servers. They are both 4 way 400 MHz Pentium. Each has over 1 GB of RAM.

   One is used for production and the other one is a test environment.

   We chose NT because that's where most of our expertise and knowledge is, so yes, we are comfortable operating in that environment. As we move forward, all the software that we use is scalable to UNIX and AIX, so there could be a possibility of migration to a larger platform."

2. What about development and test?

   JS&JC: "We do most of our development on the test server. We do have another machine that can function as an OLAP server from time to time, if we just want to play around with something."

3. Where does your source data come from?

   JS&JC: "We have a very strong data warehousing environment. We can source from our mainframe based systems, and from our Data Warehouse on DB2 Universal Database on NT, which is also available to our users for ad-hoc query and reporting.

   a. Are you pulling this data directly from the SQL interface or do you snapshot through files?

      JS&JC: "We can do both as required.

      We prefer to use the SQL interface. We haven't noticed any particular advantages or disadvantages either way, the SQL interface is just a more integrated way to source the data."

   b. How much work do you do in the data warehouse to get the data just right for OLAP? Are you building special tables, special views that you load from?

      JS&JC: "No, we are able to do most of our conversions in the SQL."

4. What storage devices do you use, and how do you use them?

JS&JC: “The primary drives are RAID 1 and the data and index drives are RAID 5. They both have over 100 GB of DASD.”

5. How do you optimize server resources allocation?

   JS&JC: “Our users have good response times, so we concentrate on allocating the server resources during load and calculation. We have learned how to make best use of the available CPUs and memory by load balancing across all the jobs that need to run.

   We have scripted all of our load and calculations, and as part of the scripting we distribute the available memory according to the requirements of each cube. This is done dynamically before each set of loads and calculations start. The scripts set the caches for optimum calculation performance then we reset the caches to the level required to support queries, when the set of calculation is complete. We make sure that we don’t over commit memory by running the right cubes together in a set. A set may run up to 4 concurrent calculations if there’s enough memory to go round.

   a. So at some point you’ll be running 4 concurrent calculations, and at other points one or two, depending on how much memory they require?

      JS&JC: “Yes.

   b. So the maximum concurrence is dictated by the number of CPUs available, and your scripts are dynamically allocating memory to the OLAP Server caches based on the operation that’s in progress?

      JS&JC: “That’s correct.

6. How many cubes are you supporting?

   JS&JC: “We have 27 cubes on the production server, 21 cubes on our test server. They currently range anywhere from just a few MB up to 4 GB.

7. What applications are the cubes supporting?

   JS&JC: “All our cubes are used for financial analysis.”

8. How many users are you supporting?

   JS&JC: “We have about 85 users in total, and maybe 10 to 15 logon at any one time. And we see that growing very much.”

   a. What’s your strategy for supporting more customers, and more users? Will you be adding more NT servers?

      JS&JC: “We are in the process of evaluating our OLAP architecture to get the best performance. Whether or not we continue to use NT servers or larger UNIX server will be determined in the analysis.”

b. What’s going to be the key thing that drives you to add another server, is it going to be calculation time, the number of users, or the physical partitioning that you get because you can serve one customer from one server and another customer from another server. What’s going into that decision process?

   JS&JC: “A combination of all of those.

5.2.4.2 Design

1. Where do your OLAP requirements come from? What’s the planning process that you go through when you are introducing a new OLAP cube?

   JS&JC: “Basically it’s dictated by our clients. By the requirements that they have, their Business Intelligence needs.”

   a. Do the business users understand OLAP?

      JS&JC: “Yes, they do.”

   b. So you’ve got a trained business community and they bring you the requirements?

      JS&JC: “Yes.”

   c. Do you go out looking for new opportunities or do find the business community keeps you busy enough?

      JS&JC: “A little bit of both. We talk to our business users and we tell them the availability of the information we have and we listen to their needs and from those needs we decide what the new opportunities are.”

   d. Business users who aren’t aware of the technology sometimes think they need every available dimension, all at the lowest level of detail, all in one cube. Do you get those sorts of requests and if you do, how do you deal with that?

      JS&JC: “We know what the capability of the warehouse is. We fully explain the functionality and the response time they will receive the larger their cubes get. We have guidelines that we use to decide how large the cubes can be and how much granularity we can provide.”

   e. So there’s an education process, and you mentioned guidelines. Can you tell me something about those?

      JS&JC: “Well, most of our clients take the cube that we have developed, to meet the marketing and financial needs that they might have. They make modifications to meet their specific needs and we limit the dimensions between 7 to 10 depending on the model. We don’t differ very often from that, so the guidelines are the basic cubes that we've built."

f. So in effect you've built templates that the clients can then modify within certain bounds.

   JS&JC: "Yes."

2. When developing a new OLAP model, one that you don't have a template for, what techniques do you use?

   JS&JC: "We have things like sizing spreadsheets that we've created where we can input the number of stored members, and based on the formulae we get an estimate of DASD space."

   a. What about finding the right dimensions?

      JS&JC: "We are very familiar with our data and the business areas, so we will create an initial outline and then work from there to try and achieve a good block size and good density.

   b. Do you use design on paper or do you use Application Manager?

      JS&JC: "We quickly get into Application Manager and use it as an iterative design tool."

   c. How long does it usually take to develop the initial outline?

      JS&JC: "The initial outline development is probably the shortest part of the process. It's satisfying the client's real requirement that takes time. We go through a period of quickly adjusting the outline. At the same time the client is becoming familiar with the OLAP environment and what it's going to actually do. Once the clients see data in a cube and see how they can manipulate the data that's when they really start thinking. They may decide that the current design is not really what they had in mind, scratch the whole thing, start over, and that's OK. They might want to add a couple of existing dimensions, something new, a new measure, and so on. Depending on the application, it can take two to four months to develop it and put it into production. By the end, the clients are usually pretty satisfied with the end result. We haven't had to pull a cube out of production to modify it."

   d. So you are using the technology as quickly as possible rather than doing abstract design, you're using the technology to work through the design.

      JS&JC: "Yes that seems to work best. Give them something that they can see and use, rather than working on an abstract design."

   e. When you're building dimensions, particularly for a new model, do you build directly from your warehouse environment?

   JS&JC: “Yes, we build dimensions from our warehouse environment, then make changes and add hierarchies to meet the needs of each client. For example, each client probably has a different way of doing portfolio classifications. These may not be in the warehouse or our internal systems, but as long as we can load the data at a lower level that’s fine.”

3. What tuning steps do you go through as you develop an application?

   JS&JC: “Beyond sparse dense and block size, use of dynamic calculation is very important. Making parents in dense dimensions dynamic, and formulae dynamic, has helped tremendously. We concentrate on calculation times because we want our batch process to be as small as possible. We haven’t found an increase in the query response time with dynamic calculation.”

   a. Do you have a checklist written down anywhere or do you go through a mental list with things like block size, outline order, dynamic calculation?

      JS&JC: “We do have a checklist in that we follow an outline that we’ve already created though we don’t design an exact copy, it’s very similar. All the dimensions are tagged the way you think they would ultimately perform, so we do have a kind of template.”

   b. Do you put the templates through a review process to make sure they are good?

      JS&JC: “Yes, we incorporate all the design techniques that we have found.”

   c. How did you arrive at the set of design techniques?

      JS&JC: “We did a lot of experimentation and testing.

   d. Any particular tips, techniques that you always put into practice around data load?

      JS&JC: “We follow the recommended hourglass outline design and then we use ORDER BY in the SQL to get the data in the right order.

   e. There has been some debate about the best way to handle formulae, using calculation scripts or outline formulae. Some say put everything in the outline, some say put everything in a calculation script, what have you found?

      JS&JC: “We do use some formulae but it’s been limited because we use substitution variables, and since you can’t incorporate those into a formula we can’t use them.”

   f. What are the substitution variables used for?

JS&JC: “Mostly for the months because we’ve got a lot of time series base analysis going on.“

4. How do you go from prototype to production model? On average how long does it take you to develop and deploy a new model and get it into production?

   JS&JC: “We do carry out frequent tests before the cube goes into production. We watch the cache hit ratios and check that we have enough memory allocated to them. We like to keep our index cache around 100%.

   During the last month of the development cycle, the clients have access to the cube to run queries and do testing with us. So, once they feel it’s OK, we push it into production.”

5. How do you validate your cubes?

   JS&JC: “We go through a formal process in putting cubes into production, which includes data validation. We check back against existing reports that run against the source systems on the mainframe. We work with the client validating how the fields are calculated and where the data came from and we develop a validation process that uses a set of reports generated from the mainframe system that we validate against every time we build the cube.”

5.2.4.3 Client tools

1. What client tools do you use?

   JS&JC: “Some use the Excel spreadsheet add-in, others use Cognos Powerplay or Business Objects. So we’ve got a range of client tools.”

   a. How do the users choose one or the other or do they tend to use a mixture?

      JS&JC: “We find that our customers will tend to use the tool that they already had. One of the advantages of DB2 OLAP is the open architecture that has a wide range of client tools, allowing our customers to use what they prefer.”

2. Who designs the reports that your clients run?

   JS&JC: “The clients design their own reports and a lot of the spreadsheet use is ad-hoc.”

3. Do you publish reports from your data?

   JS&JC: “No, our clients want the ad-hoc capability that an OLAP Server provides.”

5.2.4.4 Administration

1. How do you manage the outline updates?

   JS&JC: “Just new members are added to each dimension from time to time and when those new members come in the dimension build take care of the positioning of them automatically.”

2. How do you manage load updates?

   JS&JC: “We are updating our cubes either by completely rebuilding them or by using incremental loads and calculations. We do both depending on the size of the cube. If it’s small we do a quick rebuild, if it’s large we do an incremental build.

   a. Have you found any advantages or disadvantages to either method?

      JS&JC: “The smaller cubes lend themselves to a complete rebuild more so than incremental and that seems really to reduce the fragmentation.”

   b. How often do you refresh the cubes?

      JS&JC: “We do monthly updates.

   c. How much history have you got in the cubes?

      JS&JC: “Everything from one month to two years.

3. How do you manage security?

   JS&JC: “We don’t spend a lot of time making changes and updates to security. When our clients want to add a user, they’ll notify us through email and we set up the new ID. Security is straightforward and my staff is responsible for looking after the security based on requests from the user community.”

   a. Do you integrate the OLAP security system with any other systems?

      JS&JC: “No.”

5.2.4.5 Operations

1. How do you manage backups?

   JS&JC: “We use ADSM to backup our page and index files and the application directories. We also use export. We’ll export level 0 data to text files, we’ll back those up.”

   a. How often do the backups run?

      JS&JC: “Once a week. Every Sunday, unless there have been a lot of changes on a particular database, then we’ll take a backup as often as we need to.

b. Are the weekly backups part of the batch process for building the whole cube?

   JS&JC: “No, it’s an independent process.

c. You mentioned either backing up page and index files or doing a level zero export and then the application directory for load rules and things like that. What about other parts of the system, things like configuration files and security files, are they included?

   JS&JC: “Those are included.”

d. Do you take incremental backups?

   JS&JC: “We usually do full backups.”

e. So with a full backup if you needed to recover you’d just restore that particular application?

   JS&JC: “That’s right.”

2. Do you have any disaster recovery plans?

   JS&JC: “Yes, we have. One of the reasons we do full backups is in case there is a disaster that would prevent us from accessing the server or something along those lines. We could use the full backup to restore to another server.”

   a. Do you have the tapes stored in two places or do you have offsite storage for the tapes?

      JS&JC: “Offsite storage.”

   b. If the absolute worst happened, an earthquake or something and you whole data center went down, do you have any plans in that area or not?

      JS&JC: “Yes, DST has very comprehensive contingency plans for every aspect of our business including this area.”

3. What’s your strategy for applying fixpacks or patches?

   JS&JC: “We load fixpacks onto one of the stand-alone machines that I mentioned earlier. If they test out OK, then we update the test server and go through some more testing to identify any bugs that may have interfered with our applications. Once we feel confident with the fixpacks, we put them on our production server.”

   a. What’s your strategy with regard to frequency of service. Do you look at every fixpack?

      JS&JC: “Stability is what really drives our need. If it’s something in the readme file that addresses an issue that we have then sure we’ll download it and install it. But if it's for some functionality that we just don't utilize, then we don't bother."

   b. What's your ideal update frequency?

      JS&JC: "We like the fact that fixpack are available frequently, so we can decide which ones suit our needs.

   c. How often do you put a fixpack into production?

      JS&JC: "The test server is running the latest release and it has the latest fixpack. We found we had to apply this fixpack quickly because of a problem with the SQL Interface. The production is server is still running the previous release, and we are about 3 fixpacks behind the latest available level. We have never had to apply every fixpack."

4. What batch window are you working with today?

   JS&JC: "Each month, it's completed within 24 hours. We can't start the batch process until the files are available, but we haven't had problem at all, with meeting our deadlines."

   a. So your OLAP batch window is defined on one side by the availability of the data from the data warehouse. What defines the other side?

      JS&JC: "We have a service level agreement with our clients that the warehouse including the cubes will be available three calendar days after month end processing is completed."

5. How do you monitor the batch process?

   JS&JC: "Between our developers someone is always monitoring the process."

   a. Are you using any automation to detect failures?

      JS&JC: "At this point it's down to the team monitoring the process."

   b. How do you know if everything finished OK?

      JS&JC: "The trigger is each calculation ending. When that happens we run report scripts against the cubes and ship the report files to a different platform where they are compared against reports that are generated directly against the source system on the mainframe. If the values don't equal each other we know something isn't right, and we can research the reason."

   c. So you're not just checking the things finished, you're also checking the accuracy of what you built?

      JS&JC: "Yes, our clients require that. We work with the users to identify the important types of reports they will run, and we build validation reports to match."

d. Have you ever missed the batch window?

   JS&JC: “No.”

e. Is their anything in particular reason why you manage to meet the requirements so often?

   JS&JC: “Our business analysts spend a lot of time with the clients to understand their requirements, so the production applications are quite stable. We do a lot of testing and validation, and our developers are constantly trying different techniques to improve things and reduce the processing time.”

5.2.4.6 Conclusion

1. What do you consider to be the world class things you are doing?

   JS&JC: “First of all, our accuracy. We spend a lot of time on the validation process and we have been able to validate our data to the penny against the mainframe source systems. That really gives the client a good feeling about our product and its accuracy. Next would be the skills that we have developed in working with our clients to understand their business requirements and supply information that is valuable. “

   a. “What is it that enables you to do that?

      JS&JC: “We have a lot of experience understanding the business, the data and the technology. DST has a lot of experience in this.

   b. Any other best practices?

      JS&JC: “On the technical side, our template based development process is very strong, and we manage server resources very efficiently.”

2. What is the most important change you will make to improve your OLAP installation?

   JS&JC: “From a technical standpoint we want to incorporate some of the functionality provided in the latest release, for example Attribute dimensions. We also want to use OLAP Integration Server to give our clients some additional functionality that will take this product one step further. We are also looking at delivering the information through the Web — we would really like to see more seamless access to relational and multidimensional data delivery through the Web.”

5.2.5 Anonymous person's interview

Anonymous (**AA**) is working in a large manufacturing company in USA.

5.2.5.1 Environment

1. Describe your environment

   AA: "We use Windows NT server, Windows clients, Excel/VBA interface, FTP for formatted workbooks, InTouch automation server, Essbase 502 patch11. We have chosen Windows NT server and Windows clients for lower cost and available internal support.

   Each OLAP server has 2 GB of memory, 4 processors and over 60 GB of disks. We are using Compaq Proliant 6500 6/200 processors: it was the optimal NT server configuration when purchased."

2. What about development and test

   AA: "We have 1 system for production, 1 for test and 1 for backup. Each system has 2 GB of memory.

   Separate production and test systems allows testing of new versions/patches, database changes, outline changes. Backup server provides redundancy for production server."

3. What storage devices do you use and how do you use them?

   AA: "Mirroring (EMC) provides redundant disk storage for production server. This provides the ability to quickly switch from production to backup server. "

4. How do you optimize your server resources?

   AA: "For memory usage, we determine database block size, cache settings, Essbase configuration thread settings. For disks, allocations are based on estimated database sizes.The index and page file estimates/size determine the amount to allocate.For processors, resource usage is esteemed during loads and calculations).

   Database design determines resources use. Design alternatives are reviewed for impact on resource usage."

5. How are you exploiting the multiple processors?

   AA: "They are used to the extent that Essbase takes advantage of the multiple processors, that is calculations, retrieves, and so forth."

6. How many OLAP cubes are you supporting?

   AA: "We manage 7 production cubes and all cubes are contained on one production server.The page file sizes varies in range from 200 M to 2 G."

7. What applications are the cubes supporting?

AA: “Actual General Ledger results reporting and Rolling Budget forecasts.Each business unit queries only their own data.”

8. How many users do you support?

   AA: “In average we support 70 users and our maximum is 150.

9. What is the makeup of your team on OLAP system administration?

   AA: “4 people are working on OLAP system administration that means on customer security administration, on outline changes on test/testing, on migration from test to production.”

5.2.5.2 Design

1. Where do your OLAP requirements come from?

   AA: “Customer reporting requirements are identified and then appropriate tools/methods (ex. OLAP) are selected”

2. When developing a new OLAP model, what techniques do you use?

   AA: “We use standard customer requirements analysis. Data analysis is very important. We perform extensive prototyping/design alternatives.”

   a. How do you recognize an OLAP opportunity?

      AA: “Customer requirements identify OLAP reporting characteristics (pivots/drilldowns/rollups). We control number of dimensions and granularity using data analysis. We identify required dimensions/members and then we perform feasibility/optimization/cost analysis.

      We also determine if partitioned cubes/SQL drill-through will provide sufficient reporting detail.”

   b. What tools and techniques do you use for high level modeling?

      AA: “We use data/affinity analysis and also cases. We identify data requirements and views. We are also considering existing reports.

   c. How much design work do you do on paper or with the OLAP Server?

      AA: “We use paper for initial requirements/high level modeling and OLAP for more detailed prototyping/feasibility analysis”

   d. How quickly do you aim to develop an initial outline?

      AA: “We are able to get an initial outline very shortly after high level data requirements are identified.”

   e. What steps do you go through to get the source data in place?

      AA: “We manually collect from existing files/spreadsheets, we extract from operational sources.”

f. How do estimate cube size and resource requirements?

   AA: “We did the following:

   - Determine block size
   - Determine/test cache sizes
   - Determine OS thread count
   - Determine index and page sizes/estimate growth
   - Estimate load/calc usage

g. What are the most important configuration steps at this point - block size, dense/sparse...

   AA: “The most important steps are:

   - Review block size to determine if optimal
   - Test data load times and review data input order
   - Review calc time for entire cube and sub cubes. Review use of dynamic calculations and label only members
   - Review retrieval time for stored, dynamic, and parent members.

h. What tips/techniques do you employ at this stage - dynamic calculation, label only, outline order...

   AA: “We follow the guidelines:

   - Store only what is required (dynamic calculations/ label only/ UDAs)
   - Test retrieval times for dynamic calculations
   - Outline order for dynamic calculations.

i. How do you build full dimensions?

   AA: “We build full dimensions manually or by extracting from SQL or sequential files.”

j. How do you source data?

   AA: “We use worksheets or extract programs. It is important to identify data sources early and to verify data source contents/accuracy.”

k. What tips/techniques do you employ for data load? - ordering, transformation using SQL

   AA: “We perform sample export to determine optimal data load order, we reduce size of data file to reduce load time and we use outline order

l. Do you use calculation scripts or outline formulae?

   AA: “Dynamic outline formulae were extensively used to reduce batch calculation times and storage requirements. The formulae were balanced with retrieval times. There are no calculation scripts other than to clear/move data.”

m. How do you test query response time?

AA: “We load representative data files, we test dynamic calculations and we test all dimension level combinations/zooms/pivots with Excel client and report scripts. We use set of Excel worksheets and report scripts to compare between different implementation alternatives.”

3. How do you go from prototype to production model?

   AA: “We test on Essbase server dedicated to test. We do an initial production testing and we deploy to customers before next Accounting monthly closing cycle.”

4. What tuning steps do you go through tips/techniques do you employ for tuning?

   AA: “We do the following:

   - Review cache settings, there is no set method to determine which settings will be optimal, and we test with different settings
   - Review retrieval buffer
   - Review block size
   - Test use of dynamic calculations and review impact on retrieves and data loads/calcs. Review impact of outline on calc order.
   - Test dynamic calculation retrieves
   - Review default calc times
   - Review all retrieves
   - Perform full data load test
   - Reduce unnecessary rollups, use label only
   - Only store what is required. Use dynamic calculations
   - Use partitions to reduce cube calc times

5. How do you go from prototype to production model?

   AA: “We do the following:

   - Load all dimension members to verify outline
   - Create full production data load files to verify data sources and contents
   - Review all security filters
   - Extensive customer testing of prototype
   - Create new app/db and copy outline and data files to production and we create security groups/filters

5.2.5.3 Client tools

1. What client tools do you use?

   AA: “We use Excel addin, Visual Basic utilities for security and LRO administration. We pick these tools because they are part of standard

Essbase package. Visual Basic API is provided. The most important features of the client tools we use are ease of use and no additional cost. Customers use Excel addin. IT uses Visual Basic."

2. Who designs and builds reports?

   AA: "Corporate customers build formatted reports. Worldwide customers create ad-hoc reports."

3. How do you publish reports?

   AA: "Reports are stored on Windows NT server and available for FTP access in application."

5.2.5.4 Administration

1. How do you manage the outline updates?

   AA: "The outlines change at minimum of once per month. We do manual and automated dimension builds. We use automated build with SQL and host extract dimension build files. We test change impact on database performance and we verify reporting accuracy.

2. How do you manage load updates?

   AA: "Cubes are rebuilt if they are fragmented. Restructuring may lose LROs; then after the outline changes, we restructure all data if there are LROs to preserve. If not, we clear upper blocks and we restructure level 0 data. Incremental data is loaded after each facility closes their books.

3. Do you manage a rolling time period? How? What issues do you face?

   AA: "There is a dimension that identifies which budget submission period."

4. Do you use Application Partitioning and how?

   AA: "Previously, we had used partitioning. An error in Essbase caused occasional excessive replication times. We replaced with report scripts to replicate. You must keep all partition definitions in synchronization."

5. How do you manage security?

   AA: "When new business units or new customers are identified, Essbase groups, userids, and filters are added. Filters are changed each month to allow cube updates to current month/submission data. DB2 tables provide backup and logging for security changes.

5.2.5.5 Operations

1. How do you manage backups for OLAP?

   AA: "EMC is used to provide mirror for all production Essbase directories (system and application). EMC mirror is split and cubes are validated/exported on backup server and copied to be backed up. Mirror is re-established after backups. We use EMC for mirroring and InTouch for job scheduling. Backups are run daily. We recover by restoring backup directories and reloading source data for ssaudit log files."

2. Do you have any disaster recovery plans?

   AA: "Backups are stored at secure site and will be restored as part of overall LAN recovery."

3. What's your strategy for applying fixpacks or patches?

   AA: "LAN support group apply OLAP fixpacks 1-2 times per year. Upgrades are scheduled at same times each year. All versions/patches are tested on test server. All Essbase software is tested (utilities, Application Manager, API). The upgrade process follows:

   a. New patch is applied to test server
   b. Test scripts are updated
   c. Any conversions are performed
   d. All Essbase software is tested
   e. Upgrade to production scheduled w/customers
   f. Upgrade installed on prod server
   g. Run Installation verification scripts

4. What batch window are you working with today?

AA: "There is a 6-day window that all facility data must be loaded and calculated. There is a 2-hour window after the last facility data is loaded that consolidated analysis/reports must be completed. Production Essbase must be available 7/24 and to minimize downtime for backups.

   a. What defines the batch window?

      AA: "Corp Accounting reporting requirements."

   b. What processes do you run during a typical batch process, and how long do they take?

      AA: "The processes we launch in a typical batch process are:

      1. Facility data clear, load, and calcs. 10 minutes
      2. Calculate all data for all General Ledger data cubes. 20 minutes

5. How do you manage errors in the batch process?

   AA: “We use InTouch automation software emails notification (also ability to page). Error files are corrected, re-loaded and calculated.

6. How do you automate the batch process?

   AA: “We still use InTouch automation software. Host jobs create InTouch job and data load files. InTouch notifies if there are any errors.”

7. What makes up your batch process?

   AA: “Applications are always available other than the few minutes for the backup process. We miss the batch window less than 1%. We implemented additional monitoring/logging of data loads to notify when InTouch or Essbase.

5.2.6 Rich Semetulskis' and Alan Farkas' interview

Rich Semetulskis and Alan Farkas (**RS&AF**) are working at Thinkfast Consulting, a business partner. Rich Semetulskis is a Senior Project Manager and has over 15 years in implementing reporting and analytical solutions using multi-dimensional and relational technology.ThinkFast Consulting, Inc. is a full-service provider of enterprise-wide business intelligence services designed to provide customers with access to information for strategic and tactical decision making.

5.2.6.1 Environment

1. Describe your OLAP environment.

   RS&AF: "We used OLAP systems running on SUN SOLARIS with four processors, 1 GB of memory and 30 GB of disks and on NT with two 500 MHz processors, 2 GB of memory and 30 GB of disks. We will recommend UNIX platform for production."

2. What about development and test?

   RS&AF: "We recommend to have a dedicated platform for test and production, with the same operating system to facilitate the migration process from development to production."

3. Where does your source data come from?

   RS&AF: "Our source data actually comes from SQL (we used ORACLE V7 and DB2 V5.2 data sources and we commonly use SQL interface with load rules files. For any new OLAP database development, when we have multiple different sources in input, we try to store data in a relational repository and to build a star schema model, cleaning and transforming data using SQL scripts.

   If we compare the loading time between flat files and SQL interface, loading from flat files is quicker, but we prefer sourcing the data in a relational repository to avoid the time needed to create flat files and the file transfer".

4. How many cubes are you supporting?

   RS&AF: "We have 3 applications by production server with a total of 15 cubes between 10 to12 GB."

5. What applications are the cubes supporting?

   RS&AF: "We have different kinds of applications from marketing and analysis of parts inventory to financial, budgeting, manufacturing applications."

We think that the capability to manage attributes will open and will expand sales and customers analysis."

6. How many users do you support?

   RS&AF: "We have 1000 potential and 300 concurrent users defined. The average number of users connected at the same time is 25".

5.2.6.2 Design

1. How do you validate your cubes?

   RS&AF: "We are using logs to check if all data have been loaded and we compared the results with the star schema model input at the detail level. We have also automated some validation reports for the total and detail levels."

2. What tuning steps do you go through as you develop an application?

   RS&AF: "We take care on dense and sparse dimensions, getting a block size less than 64K, ordering dimensions."

3. How do you move on in a new model using the full dimensions?

   RS&AF: "When using multiple data sources, we always try to define a relational staging area and from it to design a star schema model that will fit the cube."

5.2.6.3 Operations

1. How do you manage backups?

   RS&AF: "We shut down the DB2 OLAP for backup. We are used to creating daily backups, 5 nights per week. And we take care to back up the security file to be sure not to get any data corruption.

2. What batch window are you working with today?

   RS&AF: "We have approximately 6 hours each night for the batch window workload."

5.2.6.4 Conclusion

1. What do you consider to be the world class things to do?

   RS&AF: "The key success for OLAP is to understand what are the reconciliation requirements to insure the integrity of the data for end-users and to build a strong project management methodology."

5.2.7 Aster Hupkes' interview

Aster Hupkes (**AH**) is working as an IT specialist in the Business Intelligence team of IBM Global Services in the Netherlands. She has over two years experience in designing and implementing OLAP solutions at several customers in the Netherlands and has primarily worked with DB2 OLAP Server / Hyperion Essbase.

5.2.7.1 Environment

1. Describe your OLAP environment.

   AH: "We do most implementations on AIX platforms because of the flexibility and scalability in larger, professional data warehouse environments. We have also done an implementation on AS/400, so far never on Windows NT except for prototyping. Usually the data warehouse, either DB2 or Oracle, is running on the same AIX server. However, we have also done a project where the data warehouse was located on OS/390 and the OLAP cubes on AIX.

   We always use SQL interface to populate the cubes. We have used either DB2 or Oracle as RDBMS. At different customers we have seen different environments:

   - Both the database and DB2 OLAP are on the same machine UNIX or AS/400
   - The data warehouse was located at a OS/390 MVS system, while DB2 OLAP was on a RS/6000.

2. What about development and test?

   AH: "At different customers we have seen different scenario's to separate development, test and production. Of course the most ideal situation is to have separate servers. However, because of higher license cost this is not always feasible. Scenario's are:

   - Four different servers and DB2 OLAP installations for development, test, acceptance, and production.
   - One server with a logical separation in the name of the application: prefix D-, T-, A- and P- for development, test, acceptance and production applications.
   - A mix between the above: two servers: one for production and another for development, test and if applicable acceptance. The separation on the development/test server is then done by prefixes in the application name. For example this would result in an application Sample on the production server and applications D-Sample, T-Sample and A-Sample on the development/test server.

In an ideal situation, test servers should have the same configuration as the production server, but in practice they are usually smaller, both in processor power and in disk space."

3. Where does your source data come from?

   AH: "Because we always use DB2 OLAP as part of a larger three-tier Business Intelligence reference architecture, our input source for DB2 OLAP cubes is either the central data warehouse or a relational datamart. Our input for dimension and data load is thus always a relational database, either DB2 or Oracle, and is loaded with SQL interface. Flat file input is only used when we are prototyping a cube as part of the design phase. We never build DB2OLAP cubes directly from the source systems.

   The input for the data warehouse comes from the different operational systems within a company but also external data that is purchased by the customer can be input for the data warehouse."

4. What storage devices do you use?

   AH: "We use SSA storage. The amount of storage space varies between 50 Gigabyte to 10 Terabyte for the whole data warehouse including OLAP cubes. The OLAP data and the index files are spread over multiple disks. Mirroring is usually activated."

5. How are you exploiting multiple processors on the server?

   AH: "We always define only one database per application because this allows us to calculate multiple databases in parallel."

6. How many OLAP cubes are you supporting?

   AH: "This differs per customer, but is usually between 5 and 10 at each customer. They vary in size from a few MB to 40 GB."

7. What applications are the cubes supporting?

   AH: "The cubes we have built support a whole range of applications like Sales, Finance, Inventory, Purchasing and Planning. Most cubes are used for analysis and reporting and users have only read access to these cubes. Only a few cubes are used for planning and allow users to write data back into the cube."

8. How many users do you support?

   AH: "The number of users differs per customer and currently varies between 5 and 30 users that are interactively analyzing the cubes. Many more users use the reports created on the OLAP cubes."

9. What is the make-up of the team?

AH: “The DB2 OLAP developer is part of a larger data warehouse team that contains between 3 and 20 persons. Other roles include DBA, ETL developer, project manager and so forth.”

5.2.7.2 Design

1. Where do your OLAP requirements come from?

   AH: “The requirements originate mostly from business analysts and sometimes from management. We experience that business analysts usually want everything in the cube, including all available details. Our role is to translate these requirements to good cube models that are not too large.”

2. When developing a new OLAP model, what techniques do you use?

   AH: “We use a multidimensional modelling technique to gather user requirements because it easily visualizes the OLAP model to the users. We find this technique extremely useful because it is both very simple for business analysts to understand and can also very easily be translated to the technical design by IT people. We actually draw the initial cube models together with the users in one or a series of design workshop.

   During a design workshop you can sometimes determine to make several cubes because some measures are irrelevant across some of the dimensions. We also have many discussions about the level of detail in the cube. For example if people say the need data on a day-level, you can challenge that by asking if they actually need to know what the sales were on a particular day one year ago or if they just want a daily update of the monthly figures in order to track the Month-To-Date figures for the current month.

   After the first design workshop, we make a prototype and load some dummy data in it. This prototype is input for the next design workshop to validate the requirements. We found that prototyping and letting the users navigate through the prototype is a very powerful way to validate and improve your design.

   We have built cubes varying from 6 to 12 dimensions. With the attribute dimension functionality we could remodel some of the old cubes and transform real dimensions to attribute dimensions. Unfortunately we experienced that using attribute dimensions in large cubes (1,5 Gb, 100.000 base members) can severely impact your performance on the top level of the cube. On the detail level, however the performance is fine. Although there is room for improvement, we are very enthusiastic about the extra possibilities that attribute dimensions offer.

We have developed sizing spreadsheets to estimate the cube sizes based on the calculation rules in the administrator guides. However, this sizing is just a rough indication because the largest determining factors for the size, the density are sparsity, are unknown at this point."

3. How do you go from prototype to production model?

   AH: "The prototype is the basis for our build process, but we basically start over again. The prototype is used to validate requirements and is largely manually defined in the outline. The dimensions in the production model will be generated from dimension tables in the data warehouse.

   Below are some build guidelines:

   - If possible we avoid having multiple database within one application. Since you only have one ESSSVR process per application, a database that hangs causes problems for the whole application.
   - There are multiple places in rules files where you can do things like adding prefixes and decodes / substitution of variables: in the SQL window, in the field properties field using the prefixes option, and by creating a new field (field - create using text) and joining that with the other field. To keep the rules files understandable, always do this at one place, preferably the SQL window because this is most flexible, for example:

     ```
     Select 'P_' || PRO_COD, DECODE(PRO_GRP,null,'Other',PRO_GRP)
     ```

   - If calculation times are large, consider using incremental updates. It is not always possible to do an incremental load, for example, if the history changes. Also when the calculation times are small (less than one hour) and/or only run once every month in the weekend, it is not time-efficient to spend a lot of time developing an incremental load scenario.
   - Set time dimension sparse when using incremental update.
   - Set measures dimension dense if you have many formulae. Set the formulae in the measures dimension on dynamic calc.
   - Try to define all formulae (for instance ratios) in one place, either in the outline or if necessary in a calc script. Calc scripts are only used if the calculations require a specific calculation order, which cannot be achieved in the outline. For example for allocating general cost over departments based on the revenue % of that department."

4. How do you validate your cubes?

   AH: "We find that the best way to validate a design is to use prototyping. After the first modelling workshop we usually build a very simple prototype. This prototype is kept very simple because all dimensions and measures are manually defined in the outline and so is the data. It is however key that the prototype contains all elements of the model: all dimensions and at least one member on each aggregation level in each dimension, including alternative hierarchies and attribute dimensions. This usually takes about half a day, and at the beginning of the next workshop we let users navigate through this model to validate the cube design.

After the cubes are built, the developer performs a technical test. This is usually done by comparing the cube results with the output of SQL queries or existing reports. The developer is also responsible for performance testing and tuning. After that testing the cube is the responsibility of end users.

5. What tuning steps do you go through when you develop an application?

   AH: "The amount of tuning we do largely depends on the size of the cube and whether or not there are any performance issues. In the case of a database of a few Mb, which is calculated within minutes, it does not make sense to spend a lot of time tuning. Tuning is always a trade-off between shorter calculation times and performance, between smaller cubes and user functionality. It is thus not solely a technical issue, but it also involves the users.

   Tuning steps we do include:

   - Experimenting with dense and sparse settings when a representative portion of the data is loaded, and choosing the optimal configuration based on block size, density, sparsity and load and calculation times. In version 5 we always had the tendency to keep the block size as small as possible (it should be between 8 and 64 K, we usually chose close to 8 K).
   - Optimizing the order of the outline (first dense dimensions from large to small, then sparse dimensions from small to large) and sorting the dataload according to this outline order
   - Use 'dynamic calc' or 'label only' if possible. We mostly use dynamic calc on dense dimensions, but we have also sometimes used it on sparse dimensions without too much retrieval performance impact.
   - Calculating the required cache sizes for index, data and calculator cache
   - Developing incremental load scenarios. We found that using incremental load scenario's work best the time dimension is sparse.
   - We sometimes use calc scripts to be able to influence the calculation order and use FIX statements instead of IF statements where possible.

We have formalized this knowledge in a DB2OLAP guidelines document available within our team."

5.2.7.3 Client tools

1. What client tools do you use?

   AH: "At most customers we make a distinction between analysis tools and reporting tools. With analysis tools, users can quickly and interactively navigate through a cube. Reporting tools can be used to build standard and more complex reports on OLAP cubes and distribute these to the users. The reason for this distinction is that the possibilities for scheduling, distribution and printing as well as all the layout options are more sophisticated in reporting tools than in analysis tools like Executive Viewer or the spreadsheet add-ins.

   Our preferred analysis tool is Executive Viewer from Temtec because we this is the most powerful, easy to use and intuitive OLAP Viewer we know. Some customers do use the spreadsheet add-in, but we are not very enthusiastic about the user-friendliness of the add-in."

2. Who designs the reports that you deliver to your customers?

   AH: "In most projects, delivering reporting is an end-user responsibility. We deliver one or more OLAP cubes to the end users and the users are responsible to do analysis or reporting on it. We find that most users can easily use the cubes and build simple reports. If they have more complex reporting requirements, like report generation based on scripts, scheduling and so forth, we can assist the users or build the report for them."

5.2.7.4 Administration

1. How do you manage outline updates?

   AH: "Dimensions are maintained in relational tables and are loaded in the outline using dimension build rules files. The only dimensions that are sometimes manually defined and maintained in the outline are the measures dimension (including all formulae) and the scenario dimension.

   The outline changes before each data load because a data load is always preceded by the execution of the dimension build rules files. The dimensions are loaded from dimension tables in the data warehouse. If some hierarchies are not maintained in one of the source systems, users are responsible for maintaining the hierarchies in relational dimension tables in the data warehouse."

2. How do you manage data load updates?

AH: "For small cubes we always do a full reload, for larger cubes it is worth investigating whether an incremental load scenario is possible. This is however not always possible, for instance if historic values are allowed to change. Incremental loads can be very fast if the time dimension is set sparse.

Some cubes have a rolling time frame which is implemented by adding a where clause in both the data load and time dimension rules files specifying to load members and data based on the system date. The old time periods fall off if the old time members are deleted with the dimension build."

3. How do you manage database fragmentation?

   AH: "We try to prevent it when loading the data, but we never do an export and reload."

4. Do you use application partitioning?

   AH: "We're not using partitioning very often yet, but we do see some positive aspects. Replicated partitioning can easily be used between test and production application. We planned to use linked partitioning between small, quick, high-level cubes and huge, slower detail cubes. The reason is that in these detail cubes we use attribute dimensions and they have a very, very low query response time on the highest level of the cube. Unfortunately we discovered that partitioning is not possible over attribute dimensions."

5. How do you manage security?

   AH: "As a rule, we always define security on group level, even if there is only one person in that group. Security is never defined on user level. Naming conventions are also important and we try to comply with the customer's standards as much as possible. Usually this means using the LAN or MAIL user names. Defining security is the responsibility of a single group.

   We sometimes use filter definitions to hide part of the cubes for some users or to allow users to write in specific parts of the database. This is very powerful, but a problem we found is that filter definitions are not easily copied from one server to the other. Some front-end tools can use the DB2 OLAP security, others need separate security."

5.2.7.5 Operations

1. What steps do you use to put a new application into production?

   AH: "If there are different servers, we FTP the outline, rules files and calc scripts to the production server. If there is one server, we copy the database. Also the batch jobs are copied and scheduling of the batch jobs is completed.”

2. How do you manage backups?

   AH: “I have managed backups in two ways:

   - Using the BEGINARCHIVE and ENDARCHIVE commands
   - Shutting down the Essbase server, make a full backup of all files and restart the server.

   Backups are usually done at night, so it is feasible to stop the server, make a back-up and restart the server. This is the least complex scenario.

   In the backup strategy it is possible to distinguish two things:

   - OLAP definitions like outline, rules files and calc scripts. These cube definitions should be backup regularly and are usually part of the daily backup.
   - OLAP data; the index and page files. In many cases it is not required to backup the OLAP data because the relational data warehouse from which the cubes are loaded are backed up. In case of a disaster, all cubes can be loaded from the data warehouse within one day, and this scenario is usually sufficient. However, most administrators find it easier to just backup everything.

   Backups are usually done every day. There have been many discussions about different backup requirements per application, but system administrator usually end up making a backup of all application every day because this strategy is the easiest to implement.”

3. Do you have any disaster recovery plans?

   AH: “At some customers disaster recovery procedures are very strict and are also tested. However, most customers just take backups and don’t test the disaster procedures.”

4. What is your strategy for applying fixpacks or patches?

   AH: “In practice, we apply a fixpack if we need it because it fixes an existing problem or contains new functionality we absolutely need. If there is no immediate reason, we keep the environment stable and do not apply the fixpack.

   If there are separate test and production server, we have to opportunity to install the patch on the test server first and on the production server a few weeks later if it has proven to work.”

5. What is your strategy for applying new releases?

   AH: “We install the new releases if we really need it because of the new functionality. The attribute dimensions of version 7 were a great new functionality, so we installed that with most customers quickly. We did have some problems with the migration of existing cubes; the calculation times of some cubes grew larger in the new version and SQL interface didn’t work properly. This last problem was solved by the first fixpack.”

6. How do you manage upgrades from client tools?

   AH: “This depends on the customer. Sometimes the software is just installed at each user’s PC, when needed. This is only feasible if there are a maximum of 10 to 20 users and makes client tool updates very labour intensive. In other cases, the front-end tool is part of a standard client platform definition that is rolled out to the entire company.

   We are more and more moving to web-based front-end tools and one of the reasons is the manageability. When a new version of the web front-end is released, only the server software needs to be updated. When users connect, the server automatically detects whether the users need a new plug-in or not.”

7. How do you automate the OLAP operations?

   AH: “All OLAP operations are automated using ESSCMD batch scripts. The scheduling of these OLAP processes is usually dependent on the successful completion of other processes: usually the load from the source system in the data warehouse and /or the load of the relational datamart. We find it easiest to use one single scheduling mechanism for scheduling all data warehouse operations. That can be an existing scheduling tool, UNIX scripts, PostSession commands in Powermart or OPC on OS/390”.

8. How do you monitor the batch process?

   AH: “Each morning the administrator checks whether the data warehouse processes completed successfully.”

9. What batch window are you working with?

   AH: “The batch window is usually from 12:00 AM to 7:00 AM and the weekend, but in that time also the central data warehouse has to be populated from the source systems.”

Appendix A. OLAP datamart design approaches

This appendix, written by Paul Turner from Hyperion, presents a number of design approaches that allow rapid interpretation of multiple requirements and data sources to provide a fast track to building a functional datamart. It assumes a basic familiarity with relational and multidimensional databases. Knowledge of data warehousing concepts may also be beneficial.

A.1 What is a datamart?

Datamarts are typically application-specific databases designed to address particular question spaces, not to be data repositories. A datamart is the business rule-specific data provider to the visualization tool, so effective datamart design is heavily dependent on how users need the data to be presented. This differs from most relational schemas, which generally act as source systems and are designed for redundancy and performance.

Even a star/snowflake/constellation schema is not automatically a datamart. These structures are more commonly used to create data warehouses, which are application-neutral (that means not designed to address a defined question space), instead providing a standardized repository for massive amounts of enterprise data.

This is an important distinction. A data warehouse contains highly granular relational data pulled from many systems and centralized as a standard representation of enterprise data, whereas a datamart is targeted at a certain set of users who need to answer or investigate a specific set of questions applicable to their business function.

Datamarts are often built from a data warehouse. This offers a single, application-neutral, granular view of the world, with data presented to users via a series of datamarts that add functionally specific rules, consolidations, time grains and additional functions (such as budgeting) that do not make sense for the warehouse to support.

To further clarify this definition, a datamart can be thought of as a targeted database that allows:

- *Rapid ad hoc analysis of data and its aggregation levels*.

  The data present in a datamart is generally less granular than the warehouse. In this sense, it can be thought of as a complex aggregate summary table. For example, a datamart may consider sales data at its most granular by city or perhaps by store. This contrasts with a data warehouse, which may be concerned not just with the store, but with sales data for each salesperson and /or point-of-sales terminal.

The coarser granularity of the data in a datamart is one of the reasons queries remain so consistent and rapid. For example, if 10,000 transactions occur on February 1 and five occur on February 2, creating a sales summary from the relational transaction system will take significantly longer on February 1, since the query time grows with the data. This contrasts with a datamart where the time grain is by day, which contains a single number for each day and provides consistent query time regardless of the number of individual transactions.

The addition of functionally specific analytics means that multiple datamarts can be spun out of the warehouse. One mart may be targeted at Sales, another at Marketing, and another at Finance. Each will essentially be working with the same data from the warehouse, but looking at it in different ways. This allows users to:

  - See data with different dimensionality. For example, the Sales datamart could contain data broken down by salesperson, while Marketing may instead be interested in seeing the data broken down by campaign.
  - See different metrics derived from the data. Complex analytics can be layered on top of the source data that would be extremely time-consuming if derived in the warehouse. For example, a simple percent of total calculation in a datamart can be extremely challenging in the warehouse, where it would require aggregating an entire dimension (which may be millions of records because of the highly granular nature of the warehouse).
  - Write-back to the data and perform what-if analysis. For example, the Finance department may wish to test various budget scenarios against the data set. This type of analysis generally does not require the fine grain data present in the warehouse. In fact, such large volumes of detailed information would hinder their what-if calculations. Other requirements, such as storing iterative persistent budgeting results and calculating actual variance are also inefficient to perform with a fine-grain data warehouse.

- *Storing historical data beyond the range of that in the warehouse*.

  In some cases, the warehouse may be of such size that it is not feasible to store a large amount of historical data. Web logs, for example, with their fine granularity and high activity, prohibit lengthy historical storage. In such cases, a mart may be more appropriate for off-loading prior years' data in a summarized, less granular, more manageable form.

All of these key datamart requirements — instantaneous ad hoc analysis of aggregated data, function-specific analytics and access to historical data are much easier to deliver with an optimized multidimensional database than with a relational database designed for transaction processing.

A.2 Designing the datamart

Datamart design is contingent upon a number of factors:

- Reporting requirements (how users want to slice and dice the information)
- Calculation/derived data requirements
- Size of the cube
- Meaningfulness of dimensions
- Granularity of the cube (the most coarse grain of each dimension)
- Dimensional cohesiveness of the data (how many measures are relevant across all of the introduced dimensions)

All of these factors are interlinked. For example, even though a certain dimension may be necessary for a given calculation, it may explode the size of the cube and introduce a meaningless dimension in the process. Therefore, the challenge in datamart design is balancing these requirements.

A.2.1 Determining the granularity

The granularity of the datamart is tightly linked to the size of the resultant cube. Generally, most systems have a series of identifiable finest grains in the fact table (other than time). For example, it could be case number for call center systems, transaction number for sales systems, and cookie or user ID for Web logs.

Datamart designers must consider two types of granularity: intra-cube and extra-cube.

A.2.1.1 Intra-cube granularity

Intra-cube granularity determines the level of detail for dimensional drill-down. The finer the grain, the less summarization is applied to leaf-level dimension members.

Table 15 shows a number of examples.

Table 15. Granularity example

	Fine grain	Intermediate grain	Coarse grain
Time	Seconds as distinct time interval	Hours	Days or months
Product	Product SKU	Product name	Product category
Geography	House address: 35 Laurel Drive	Zip code	City

Users often have a tendency to make their datamarts too granular. Bear in mind that the data in the cube is exactly what users will be navigating, often in a highly stylized user interface. In many cases, more granular data simply creates more cumbersome analysis without adding any real value.

For example, consider the granularity of time within a Web traffic analysis application. Given the fine granularity of the source data, it may be tempting to set the periodicity of time within the datamart as high as every five minutes. But consider the value of this data over the life span of the datamart. Will users really care about what happened at 11:05:15 on February 5th, 1998? Generally, the answer is no.

In most cases, the grain of a particular dimension is mandated by the reporting or calculation requirements. Consider the following basic time reporting requirements: day, week, month, quarter, and year. The problem is that weeks don’t naturally roll up into months; “Week 5,”for example, can traverse January and February. A natural way of addressing this is to load the day grain and have it roll up into months (which, in turn, roll up into quarters and years), and also to define an alternate hierarchy with the day grain rolling up into weeks.

Another method is to use the data loading process to summarize the data at different grains. This can be done in two ways:

- If time may be represented in two dimensions, then the datamart can have a Year dimension that contains years, quarters, and months, as well as a weeks-of-year dimension that contains a flat list of weeks 1 through 52. When loading the data, simply create a two-dimensional reference of month by week. This yields a number of interesting query possibilities, such as: “Show me how the first 5 weeks of each year compare”. This method works best when members are only shown where data exists for a query (that is, “Suppress Missing”), so that drilling into weeks when looking at January, for example, will only show weeks 1-4.

- If it is necessary to have months and weeks in the same dimension, then format the time key as months (this is simple to do with SQL) and load the data into the month members. Then use a second data load to format the time key as weeks and load the data into this time slot. The disadvantage is that this doubles the time required to load the datamart from the warehouse, which may not be feasible.

A.2.1.2 Extra-cube granularity

Extra-cube granularity is determined by what is omitted. For example, Finance may be interested in seeing the Table 16 sales data, but not broken out by individual salesperson. Meanwhile, Sales will obviously want to see the figures for each salesperson.

Table 16. Extra-cube granularity example

	Dough Inc	**Oats Corp.**	**Wheat Int'l**
Paul Turner	423	542	245
Carole Turner	123		145
Jim Turner			124
Total sales by manufacturer	**546**	**542**	**514**

Finance cube: Time X Accounts X Manufacturer

Sales cube: Time X Accounts X Manufacturer X Salesperson

As more dimensions are included, the datamart becomes more granular. More importantly, this new grain (in this case, Salesperson) allows a new series of complex calculations that could not be derived without it. Consider the calculation, "average number of manufacturers covered by a salesperson."Since Paul covers three, Carole covers two, and Jim covers one, the answer is ((3+2+1)/3 =2). This information is lost when looking at the Finance cube, which presents the summarized level, "total sales by manufacturer."

But what if Finance wants to see the metric "average number of manufacturers covered by a salesperson" in their datamart? Forgetting for the moment features such as Partitioning and @XREF, how can Finance calculate this metric without including the Salesperson dimension?

Note: If the Salesperson dimension is included, one way of accomplishing this with DB2 OLAP/Essbase is to introduce a stored formula –SALESCOUNT. If the sales for the Salesperson are <>#Missing, then SALESCOUNT is set to 1;otherwise, it is left at #Missing. This gives the 3,2,1 that is required. To calculate the average, divide the total by the number of salespeople. (This can be preloaded or counted using the count function available in DB2 OLAP 7.1 or Hyperion Essbase 6.0 or later.)

This demonstrates a key challenge for datamart designers. A dimension may be necessary for some calculations, even though it is not necessary for reporting, and including it may massively increase the size of the datamart. Consider another example related to Web traffic. One common metric is distinct users, as in: “How many distinct users bought my SuperWidget?” or “How many distinct users saw my home page?”.

In this case, a unique cookie identifier (“web302ca_941227897_157097”, for example) determines a user. An e-commerce site may have millions of distinct users, each with a different cookie. Drilling into a cookie dimension would yield millions of system-generated keys such as the one above — data useless for reporting purposes. But a useful calculation, such as counting the number of distinct cookies for users viewing a certain Web page, may only be possible if the cookie dimension is included.

One solution to this dilemma, although not a particularly elegant one, is to preprocess the data at load time since the relational database knows all about cookies. But this becomes expensive from a data load perspective. For example, consider counting distinct users for various products and categories. If the distinct user count for SuperWidgetA is 1, and it is also 1 for SuperWidgetB, the distinct user count for the parent Widgets could either be 1 or 2, depending on whether the counts for SuperWidgetA and SuperWidgetB represent the same user. In other words, the distinct user count is not additive across any of the dimensions; each level must be precomputed in SQL, making the load extremely expensive.

Another method is to take a hybrid approach, supporting the distinct user analysis only through relational queries and offering graphical linkage between the datamart and the data warehouse.

Tools such as Analyzer allow this kind of link through their relational connectivity.

A.3 Deciding the dimensionality

Faced with a star or constellation schema, how can a datamart designer quickly determine the dimensionality to build a meaningful, manageable OLAP datamart? All dimension tables could be dimensions in the datamart, and the columns within each dimension table could be dimensions, hierarchies, attributes, or data. The first step is, rather typically, to establish user requirements. But these must constantly be related back to the requirements and structure of the warehouse.

There are a number of other guidelines for rapidly defining the base dimensions of the mart. Consider the information in Table 17:

Table 17. Product table

Product ID	Product Code	Product Name	Product Type	Size	Product Price	Product Intro Date
1234	ABX-1294	SuperWidgetX	Widget	8lbs	5	1/1/99
5678	ABY-1349	SuperWidgetY	Widget	10lbs	7	5/1/99
9012	XYZ-4567	DongleA	Dongle	10lbs	5	8/1/99
3456	WXW-6543	DongleB	Dongle	8lbs	7	10/1/99

There are a number of ways to model this data, depending on the users' requirements. All shaded columns could be dimensions, or some of them could be hierarchies, some measures, and others attributes (discussed later).

For example, product name and product type could be modeled as separate dimensions. Looking at Widgets for all products would give the total Widget sales, and drilling down into the product names dimension would give just those products where data exists (if product names is in the rows) — in this case, SuperWidgetX and SuperWidgetY. But this would create an incredibly sparse relationship, since a product can only have one type. Obviously, in this case, it makes more sense to model the two as a single dimension and hierarchy, with product name rolling up into product type.

Now consider product price. It could easily be a measure loaded into the OLAP database for calculations. (Because it is a one dimensional measure relating only to product names, it would most likely be loaded at the upper level for other dimensions.) But product price could also be a dimension, with the column reformatted in SQL when it is loaded so 5 becomes "0–5," 7 becomes "6–10,"and so on. This allows users to plot pricing histograms such as the one shown in Figure 39 and to answer questions such as "tell me the percentage contribution to total product sales of products priced between 1 and 5 dollars."

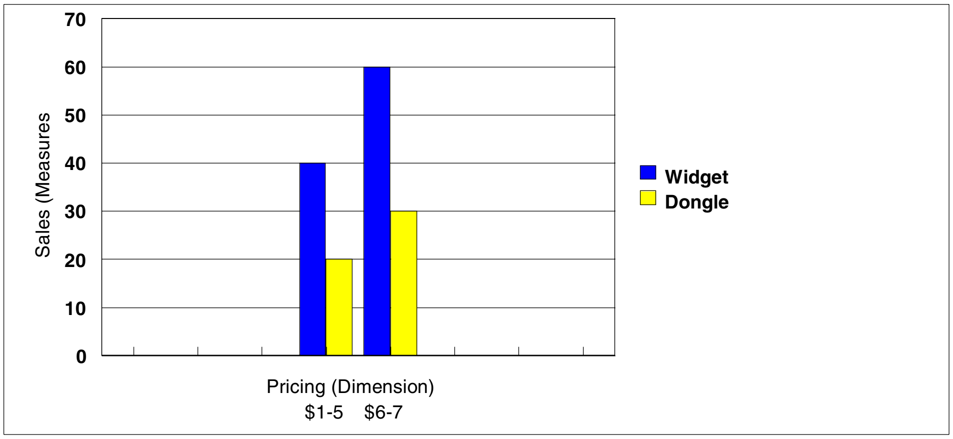


Figure 39. Pricing histogram example

Because data is represented at the lowest grain (product names) in the product table, another option would be to make the price simply a numeric attribute of the product. This would allow queries such as "show me total, average, minimum, and maximum sales of products between 1 and 12 dollars."

The product size column presents another choice. It could be an alternate hierarchy in the product dimension, allowing users to see total sales by product type or by product size. But this would not allow users to create a cross-tab — for instance, reporting total sales of all eight-pound dongles. If this is a problem, then size could be an attribute of product. Alternatively, if the attribute query is too slow because it is a dynamic calculation and many products weigh eight pounds, then it could be a stored case dimension. If users aren't interested in individual products at all, but just in their sizes and types then it would be more efficient to summarize data into the size and type dimensions.

As the possibilities above illustrate, there are many options for building a datamart from data warehouse tables. A dimension table can encapsulate any combination of dimensions, hierarchies, measures, and — by joins with other dimension tables — further hierarchies. It all depends on how the schema has been designed and how the datamart is to be used. It is often easier to determine what not to include, as shown in the following examples with different kinds of dimension tables.

A.3.1 Product and geography type dimensions

A product dimension typically has a number of attributes, which translate easily into hierarchies. These hierarchies are easily defined by questions such as: “Does a given product type always belong in a single product category?”. If the answer is yes, then assume that product type can roll up into product category. If the answer is no, then create an alternate hierarchy or a dimension.

A product dimension could contain enormous amounts of information. Datamart designers must intelligently select the dimensions and hierarchies they need, and should consider drilling through to detail using DB2 OLAP Integration Server/Hyperion Integration Server or Hyperion Analyzer virtual cubes, or enabling a “cubes-on-the-fly” approach where different hierarchies and columns may be selected on demand.

Product dimensions may also contain ragged hierarchies like in Table 18.

Table 18. Product dimensions and ragged hierarchies

Parent Product ID	Product ID	Product Name
null	1	Plastics
1	2	Dongles
1	3	Widgets
3	4	SuperWidgetX
2	5	DongleA

One way to approach this is to simply have the member names as the IDs, and use parent-child build in the dimension prep editor. The product name becomes an alias. This allows a rapid build without a self-join operation.

In the case of uniqueness (for example, plastics appears in a different hierarchy, but is really plastics in a different context), simply use the product ID as a suffix to the product name, creating members like “Widgets (4)” in the outline. There are a number of common workarounds to the uniqueness issue, although some cannot be accomplished without using a different format of the table above (for example, in generational format rather than parent-child format). Here are some of these workarounds:

- When tackling uniqueness across dimensions, use the dimension name itself as the differentiator. Consider an application where there is a salesperson dimension and a user name dimension. Both may include the member Paul Turner. Assuming that Paul Turner will not occur more than once inside either dimension, simply use the dimension name as a suffix — that means Paul Turner (username), Paul Turner (salesperson). This method of differentiation can also aid users in navigation, since it clarifies which dimension they are looking at and reduces confusion.

- When tackling uniqueness within a dimension, the best approach depends on where the uniqueness occurs. If duplication occurs at different hierarchy levels, consider using a suffix containing the name of the parent — that means SuperWidgetX (Widgets), DongleA (Dongles). If the duplication occurs in the same hierarchy — that means siblings use the same name — then use the primary key of the member in its dimension table to guarantee uniqueness, for example, SuperWidgetX(5).

Another potential issue of ragged hierarchies is inconsistent semantic selection of members. If a category contains no products — for example, a brand new category called “Gadgets”— it creates a selection inconsistency in the OLAP outline. Both Gadgets and SuperWidgetX (for example) will be at leaf level, or level 0 in OLAP outline hierarchy, even though one is a category and the other is a product. One way of addressing this issue is with user defined attributes (UDAs), which allow users to assign the attribute “ProductType=Category” and “ProductType=Product” to the members through the addition of a column in the dimension table.

This enables UDA queries on these attributes, rather than level or generational queries, which are inadequate for this type of tree.

A.3.2 Browser type dimensions

Though unique to Web analysis environments, a browser-type dimension illustrates some of the issues faced when modeling data. Web browser type is essentially a string containing information such as browser name, version number and operating system, encoded in an abbreviated, delimited format.

Some datamart designers simply insert this string directly into the dimension table and leave the decoding of it to the user interface. This basically creates a two-row dimension table, with one row containing the primary key (linked to the fact table) and the other containing the browser-type string. Consider how much richer this information could be with a more effective datamart design, as shown in Figure 40.

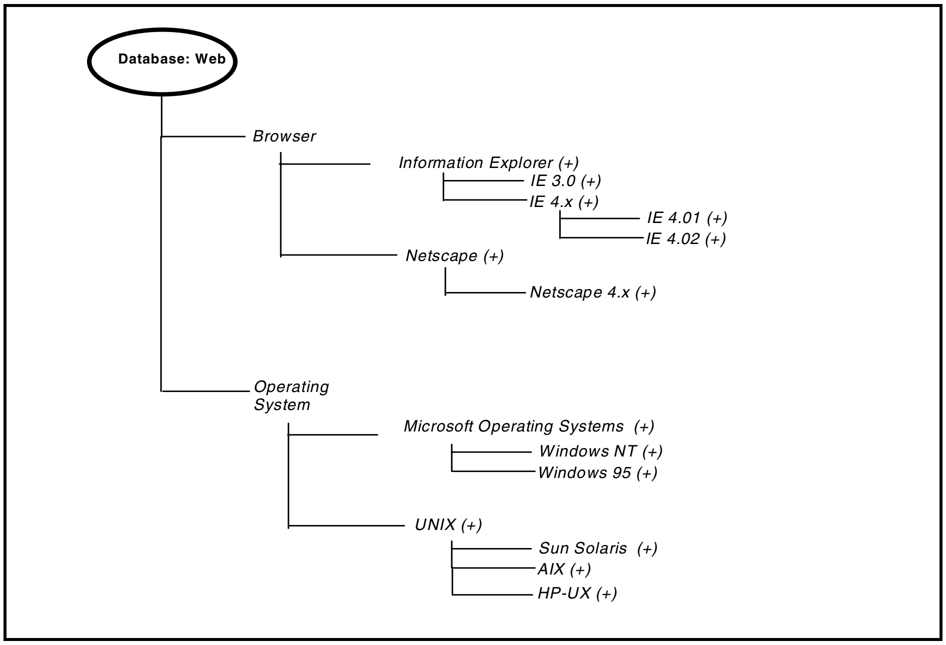


Figure 40. Browser type dimensions

This demonstrates one of the key limitations of many data warehouses: People don't know the kind of analysis that they can do, could do, and may need to do, so they don't design it into their warehouse. They do the data modeling and represent the data in a multidimensional schema, but they don't extract the information from what they are logging and they don't enrich the data with attributes. In general, the richer in attributes a data model is (in relational-speak, this means the more individual columns in a dimension table that are spent describing the member), the more useful and valuable the OLAP cube will be.

In the example in Figure 40, "IE 4.02" may have been found in the log. Preprocessing it in the log could add the attributes "IE 4.x" and "Internet Explorer". If the operating system (that means Sun Solaris 2.7) is also logged, preprocessing can include the attributes "Sun OS" and "UNIX". This enables two additional dimension tables, Table 19 and Table 20, on Browser Version and Browser Platform.

Table 19. Browser version table

Browser Version Key	Browser Version	Browser Release	Browser Class
1	IE 4.02	IE 4.X	Internet Explorer
2	IE 4.01	IE 4.X	Internet Explorer
3	Netscape 4.1	Netscape 4.x	Netscape
4	Netscape 4.5	Netscape 4.x	Netscape

Table 20. Browser platform

Browser Platform Key	Operating System Version	Operating System Class
1	Windows NT	Microsoft
2	Windows 95	Microsoft
3	Solaris	UNIX

This illustrates how some warehouse designs can significantly enrich analysis. In many cases it’s easier to address the underlying inadequacies of the data model than to force DB2 OLAP to work with less than optimal data.

A.3.3 Customer bank account number dimensions

Analyzing data that is inherently highly granular — such as individual customer bank account numbers — is a relational task. For this kind of requirement, consider a relational hybrid solution. Tools such as Analyzer are ideal for this, since they can report against both multidimensional and relational databases, mapping to a logical star schema presented as just another multidimensional data source. Linking the views together allows users to drill to detail while remaining blissfully unaware of underlying data source providers.

A.4 Understanding attributes and base dimensions

In some cases, including the most granular dimension in the datamart allows everything else (apart from measures) to be an attribute of that dimension. Consider a call center application where the finest grain of the database is case number. The database description could be as follows:

CaseNumber (base dimension, consisting of millions of case numbers)
Analyst (attribute of case number)
Location (attribute of case number)
Product (attribute of case number)
Category (attribute of case number)
Time (attribute of case number)

Of course, this would create an enormous and meaningless reporting dimension (case number) and agonizingly slow retrievals, because calculations such as “Show all the cases opened in England” would be dynamic. But it illustrates the point that an attribute can only be introduced when it is an attribute of the grain that the dimension understands. If related, everything can be an attribute of the extra-cube or intra-cube grain. If this grain is not included, the attribute must be a base dimension.

If the dimension “CaseNumber” is not included (because it is a large and meaningless series of system generated keys), then the other items must become base dimensions. Of course, these base dimensions, such as Product, can have their own attributes. But these are attributes of product, while Product is essentially an attribute of the excluded Case Number.

Attributes should be weighed against the retrieval time required to calculate. Just because something can be modeled using an attribute dimension doesn’t mean it should be modeled that way. For example, if there are 10,000 products and 5,000 of them have the attribute “expensive,” it might not be a good idea to create an attribute dimension called “Cost Attribute,” since summarizing that attribute at query time would require DB2 OLAP to run through 5,000 blocks. If each block is 20K, that’s 100,000K of data for just one combination of the other sparse dimensions. In this case, a precalculated base dimension would probably make more sense.

A.5 Tackling the data load

Loading data from a data warehouse can be a time-consuming task. The main reason for this is normalization (the process by which redundant data is removed from a table by breaking it into multiple tables and using keys to represent those records), which is an inherent part of most warehouses. In general, the more normalized the warehouse, the slower the data load because more joins are necessary to extract the mappings between the dimensions and the data.

Since fact tables comprise the vast majority of a data warehouse footprint normalizing dimension tables is unnecessary. It saves a small percentage of total space, but greatly increases data extraction time.

A.5.1 Optimizing regular dimension tables

One way of optimizing loads from relational databases into OLAP cubes often decreasing the dataload time, is simply not making joins unless they are required. Make sure that dimensions in the OLAP outline are represented in the fact table, and that for each foreign key there is a corresponding member in the OLAP outline. In other words, the grain of each OLAP cube dimension should be the same as the grain of the dimension in the fact table.

Consider Table 17 on page 161:

Product code (and its alias, product name) is the finest grain of this dimension table, since each product ID relates to an individual product. Each fact row in the fact table will relate to an individual product defined in the product dimension table.

Normally, loading data would require joining product with the fact table by using the product ID, and then associating the product code or product name (which will be level 0 member names in DB2 OLAP) with the facts. However, this approach is not suitable for very large warehouses with lengthy fact tables. It will result in agonizingly long data load times while the RDBMS spins, trying to join the tables, and perhaps the RDBMS will bomb out after hitting a query limiter or running out of temporary space.

Instead, consider embedding the primary keys from the dimension tables into alias tables within the OLAP database. Scanning the dimension table would build as shown in Figure 41, with no joins required.

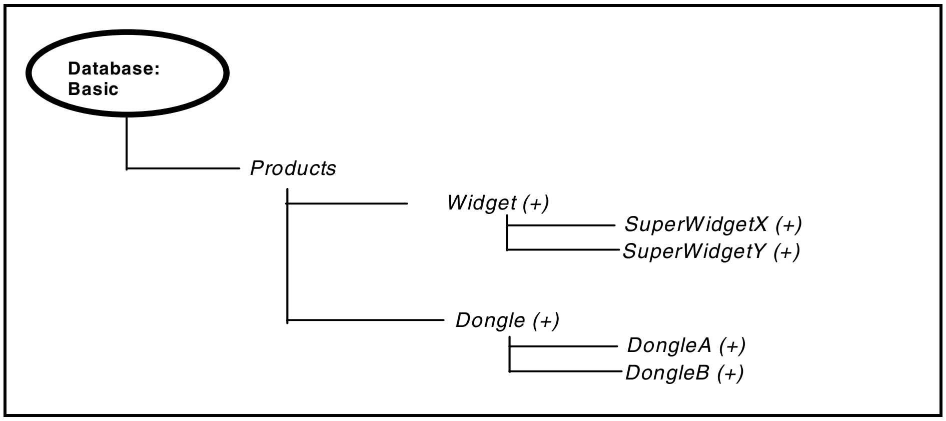


Figure 41. Product Dimension outline

Note the aliases for each product. These can be stored in one of the alias tables and hidden from the user, as they are essentially useless for reporting. But, since DB2 OLAP now knows that 1234 is really SuperWidgetX (along with every other product ID/product name relationship), the dataload process no longer must join the fact table with the product table and will execute much more quickly.

A.5.2 Optimizing time dimensions

If the key in the fact table used to represent time is a date/time format field, there is no need to join it with a time table in the schema, even if there are granularity differences between the fact table and the time dimension in OLAP database. Simply reformat the time key to the matching grain at load time. For example, if the OLAP database dimension goes down to the level of month, just transform the key into a month value during the load, even though the key also contains the day, hour, and minute information.

A.6 Conclusion

One of the keys to successful datamart design and implementation is designing the data warehouse with consideration to the datamarts it will feed. If the warehouse is designed with only the goals of normalization and representation in mind, without rich and easily extractable data and metadata, datamart design will be unnecessarily difficult. It is, therefore, imperative that the design of the warehouse address the reporting and analysis requirements of end users. Otherwise, your data warehouse will be little more than a data jailhouse, locking important information away from the users who need it.

Appendix B. Integration Server implementation guidelines

This appendix presents guidelines and recommendations when implementing Hyperion Integration Server from Hyperion or IBM DB2 OLAP Integration Server from IBM to build DB2 OLAP cubes from relational sources. We will use the term Integration Server (IS) to refer to both products.

This appendix has been prepared by Cheryl A. McCormick who is a Senior Principal Consultant in the Data Integration Services group of Hyperion Solutions. Cheryl specializes in designing and implementing datamarts / data warehouses using a hybrid of both relational and OLAP multidimensional databases.

B.1 Overview

Integration Server provides the critical link between your relational star schema and the powerful DB2 OLAP Server.

The Integration Server product family provides a way to transfer the relevant data in your relational star schema to a multidimensional database quickly (see Figure 42). The Integration Server product family provides graphical tools to help you to:

- Create a logical OLAP model from tables, views, and columns in your relational database. The OLAP model you create is a logical star schema consisting of a fact table surrounded by related dimension tables.
- Use the OLAP model to create a metaoutline, an outline template, containing the structure and rules required to generate a DB2 OLAP outline.
- Use the metaoutline to create and populate a DB2 OLAP multidimensional database.

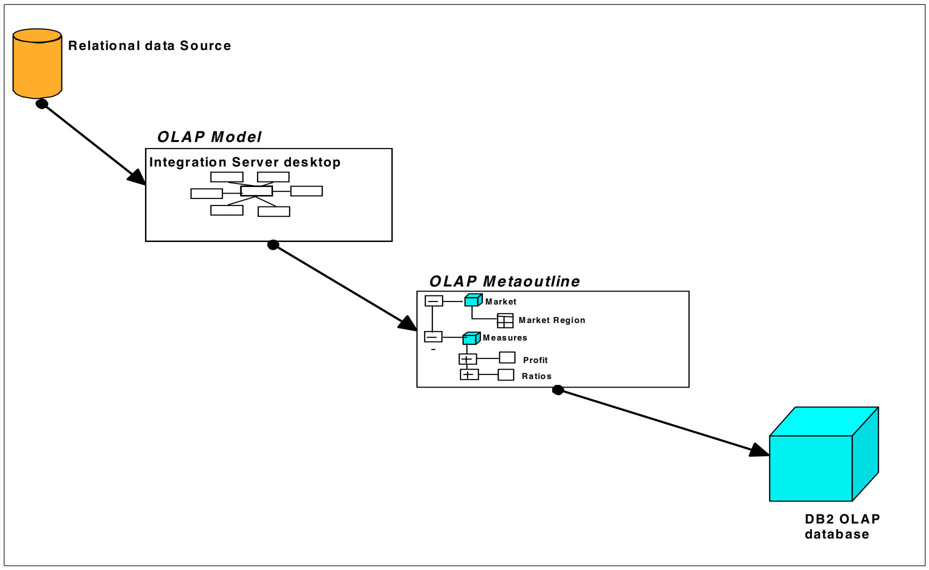


Figure 42. Integration Server to DB2 OLAP flow illustration

Integration Server consists of two major components: the desktop and the server (see Figure 43).

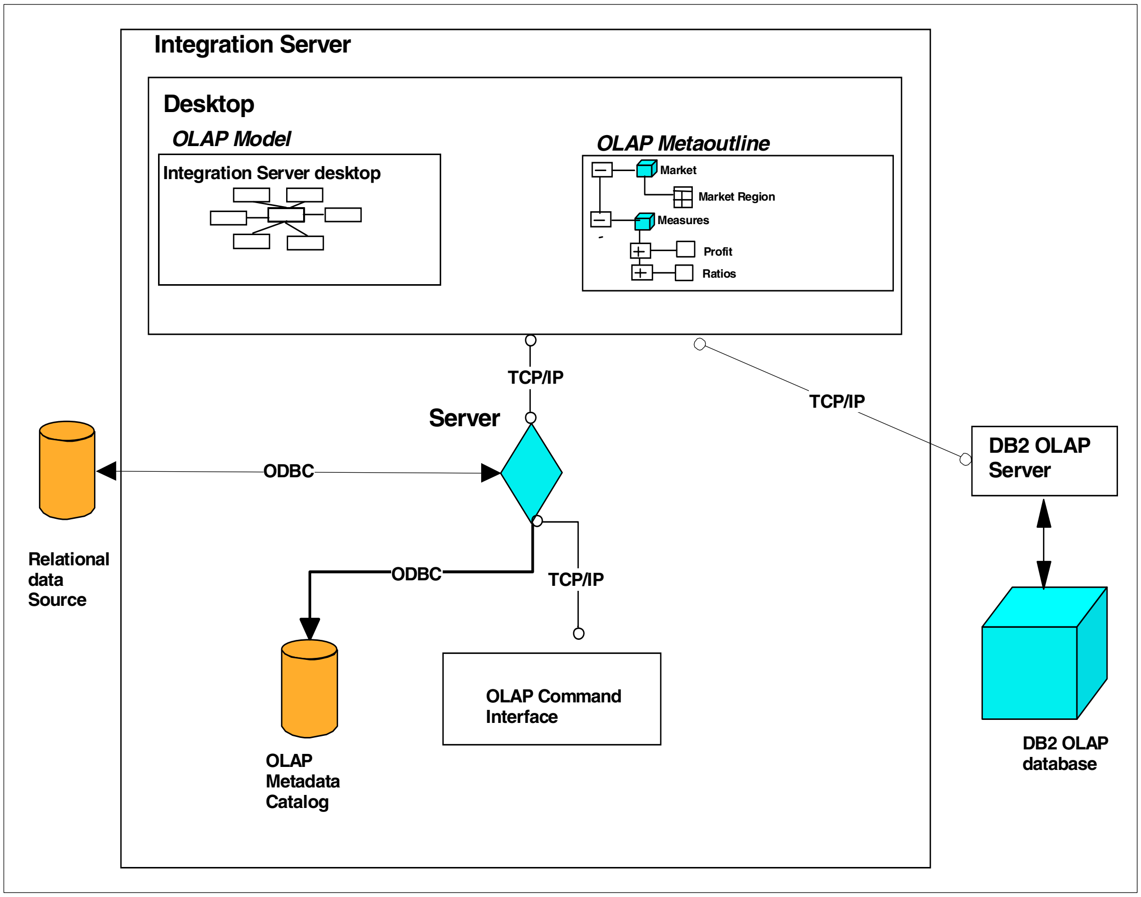


Figure 43. IS components

Workflow for using IS

To create a DB2 OLAP database from a relational data source:

1. Build an OLAP model that is based on the tables in the relational data source. IS stores the OLAP model and the information necessary to retrieve the relevant tables in OLAP Metadata Catalog.
2. Create a metaoutline from the OLAP model. IS stores the metaoutline in the OLAP Metadata Catalog.
3. Load members and data into the DB2 OLAP database.
4. Update the DB2 OLAP database with new members and data.

B.1.1 OLAP model

An OLAP model contains a star schema. OLAP models are based on the idea that values in a relational database can be categorized as either facts or dimensions of facts. Facts are the numeric, variable values in the database, such as sales figures and numbers of units sold. Associated with facts are related data values that provide additional information, such as store locations and product IDs of units sold. An OLAP model contains a fact table, one or more dimension tables, and one or more dimension branches. An OLAP model may contain time and accounts dimensions.

Unlike other integration products, IS creates an OLAP model that is a logical model, not a physical star schema. The OLAP model is a logical representation of the data values that you select from the tables in the relational database and that you want to report in DB2 OLAP.

B.1.2 OLAP models and metaoutlines

Use an OLAP model to create one or more metaoutlines. A metaoutline contains the basic structure required to build a DB2 OLAP outline and to load data into DB2 OLAP. You can use one OLAP model as the basis for another OLAP model by saving the original OLAP model under a different name and editing it as needed to meet reporting requirements. You can create any number of OLAP models for use in building metaoutlines. However, each metaoutline is based on one, specific OLAP model.

OLAP models have the following features:

- They are reusable. You can use the same OLAP model as the basis for multiple metaoutlines.
- They provide a layer of abstraction that insulates the DB2 OLAP database outline from changes in the relational database.
- They enable you to create hierarchies to structure and summarize data from a relational database. You can use the hierarchies in multiple metaoutlines.

B.2 Decision: the initial IS implementation

Working with IS for the first time, either your own implementation or the first implementation for a client, you should follow a very specific path: work backwards from the DB2 OLAP cube! Specifically, the general tasks should include these:

- Define the DB2 OLAP requirements that means to define the dimensions, hierarchies, attribute dimensions, aliases, UDAs and so forth.
- Define and test formulae and calc scripts.
- Define end-user drill-through requirements.
- Review the star schema / DB2 OLAP / IS environment and determine the most effective configuration.
- Install IS Desktop and IS server.
- Using the defined DB2 OLAP outline, develop a star schema design that supports the DB2 OLAP dimensionality and drill-through reporting requirements.
- Develop an indexing strategy for the star schemas.
- If process automation is a requirement, develop / test / deploy the OLAP Command Line Interface commands.

Each of these steps should be reviewed in conjunction with those steps contained in the IS manuals.

B.2.1 No existing data warehouse or datamart or star schema

In most IS implementations, DB2 OLAP consultants handle the DB2 OLAP implementation: consultants define the requirements, write and test the calc scripts and formulae, and define any ESSCMDS that are needed. Once the outline has been defined, or at least 75% of the outline has been defined and tested, the Integration Server consultants can begin the IS implementation. This approach of defining the DB2 OLAP cube prior to the IS implementation is very successful in cases where a data warehouse, datamart, or star schemas are not already in place.

B.2.2 Existing data warehouse or datamart or star schema

When a data warehouse, datamart, or star schemas are already in place, the initial steps should include a review of the existing star schemas and a comparison of the star schemas to the DB2 OLAP outline (dimensionality). If an existing star schema contains the dimensionality required by DB2 OLAP and the granularity of the FACT table is also consistent with the DB2 OLAP requirements, initial development efforts should focus on using the existing star schemas. In addition to the dimensionality and granularity requirements, you must also determine if the indexing strategy of the star schema will support the required performance.

In summary:

1. Determine if the star schema contains the dimensionality required by DB2 OLAP.
2. Determine if the granularity in the FACT table data meets DB2 OLAP requirements.
3. Determine of the indexing strategy of the star schema will meet the IS performance requirements.

B.3 From the very beginning: developing the star schema

In the environment without a data warehouse, datamart, or star schema, a star schema needs to be developed to support IS and DB2 OLAP. The star schema will reside in one of the IS support relational databases and connectivity established either via native ODBC drivers or MERANT ODBC drivers provided with IS.

B.3.1 The dimensional model

Another name for the star schema is a dimensional model. While entity relational models are very symmetrical, the dimensional model is asymmetric. There is one large dominant table in the center of the schema, surrounded by smaller attendant tables. This center table is called the FACT table and contains the numeric data; the smaller tables are dimension tables and contain the metadata.

B.3.1.1 Designing the FACT Table

The fact table contains the values that are later aggregated and reported by DB2 OLAP. All measures values that you want reported in DB2 OLAP are referenced through the fact table and through the accounts dimension, which can be an exact copy of the FACT table, generally in one of two formats:

1. Each dimension is represented by a column and each DB2 OLAP MEASURE is also represented by a column. Each row contains a valid member for each dimension and numeric values under each MEASURE.
2. Each dimension, including the ACCOUNTS dimension is represented by a column. A column labeled ‘DATA’ contains the numeric data values for each DB2 OLAP dimension intersection point.

Method #1 is used when the number of MEASURES is limited. When creating a star schema for a financial application, the ACCOUNTS dimension typically has a large number of members. For this reason, Method #2 is typically used for financial applications.

The Standard FACT Table [Method #1]

As stated above, the standard FACT table will contain a column for each dimension plus columns for each ACCOUNT member or MEASURE. The datatype for each MEASURE must be a numeric datatype. If any other datatype is used, IS will not recognize that column as a MEASURE.

In the TBC sample database, the FACT table is created using Method #1. Figure 44 is sample of the FACT table as shown in a dimension model.

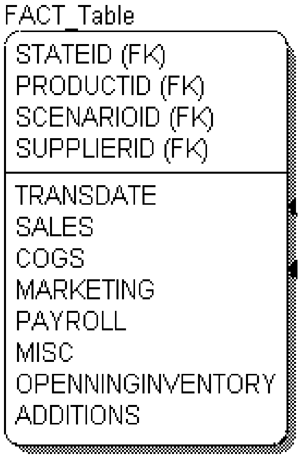


Figure 44. Fact table sample

STATEID, PRODUCTID, SCENARIOID and SUPPLIERID represent a dimension is the DB2 OLAP outline.

SALES, COGS, MARKETING, PAYROLL, MISC, OPENNINGINVENTORY and ADDITIONS represent the MEASURES in the ACCOUNT dimension.

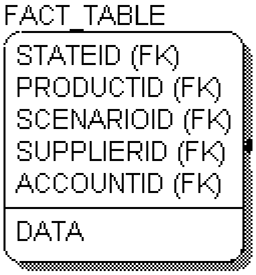
In the example, STATEID, PRODUCTID, SCENARIOID and SUPPLIERID are all foreign keys. The use of foreign keys in the FACT table is standard data warehouse practice.

Each dimension table will contain a ‘Primary Key’. One of the characteristics of a primary key is that it is unique. Each FACT table Dimension code should be a foreign key to the dimension table’s primary key. For example, the PRODUCTID in the dimension table is the PRODUCT dimension primary key [it is a unique value which is also a DB2 OLAP requirement]. The PRODUCTID in the FACT table is the foreign key. This means that the PRODUCTID must exist in the PRODUCT dimension before it can be loaded into the FACT table. This primary / foreign key relationship will also keep the

DB2 OLAP metadata and data in synchronization. For more information regarding the use of foreign keys, please refer to data warehouse documentation.

The Alternate FACT Table [Method #2]

When building a dimensional model for a financial application, Method #2 is typically the preferred method. Financial applications generally use the chart of accounts as the ACCOUNT dimension. Chart of accounts can contain thousands of accounts. Even building DB2 OLAP at a higher level of detail [that is not at the lowest account level of the chart of accounts], may create two to three thousand base accounts. Putting each of these base level accounts in a column would create an extremely large table, making maintenance difficult. Method #2 uses a separate column for the ACCOUNT dimension and one for the actual numeric data. This is illustrated in Figure 45.



FACT_TABLE

STATEID (FK)
PRODUCTID (FK)
SCENARIOID (FK)
SUPPLIERID (FK)
ACCOUNTID (FK)

DATA

Figure 45. Alternate Fact table sample

The ACCOUNT table contains the ACCOUNT metadata, including the ACCOUNTs hierarchy structure. The DATA column contains the actual numeric data related to each of the dimension columns.

Granularity of the FACT Table

IS loads data directly from the FACT table based on the default or the ‘Build to here’ criteria defined in the IS metaoutline. FACT tables are generally very large: millions and millions of records are standard. The main consideration when determining the granularity of your FACT table should be whether you have a limited time window in which DB2 OLAP must be loaded and calculated. If there is a limited time window, an aggregate FACT table strategy can be employed.

Using this strategy, review each DB2 OLAP dimension and determine the most effective aggregation placement. For example, the original FACT table granularity is at the invoice line level of detail and the DB2 OLAP cube is summarized by Product and Month. By creating an aggregate FACT table, summarizing at the TIME = Month and PRODUCT = By Product [not the invoice line item], will dramatically reduce the number of rows in the Aggregate FACT table. Using the original FACT table, IS would actually SUM to the DB2 OLAP data requirement level. Using the aggregate FACT table, IS would not have to sum the data, which would reduce the SQL retrieval time. The end result is that the data load would happen faster.

FACT table datatypes

All MEASURES in the FACT table, whether using Method#1 or Method#2, must be numeric datatypes. For Method #1, each column identified as a MEAURE must be a numeric datatype. The columns identifying the dimensions must be the same datatype as their associated Primary Keys.

For Method #2, the ‘DATA’ column must be a numeric datatype. As stated above, the columns identifying the dimensions must be the same datatype as their associated primary keys.

B.3.1.2 Designing the dimension tables

Dimension tables contain the metadata. Efficiently designed dimension tables make a tremendous difference in the performance of IS metadata loads and data loads. As stated previously, each dimension table must contain a primary key. This unique key also represents the DB2 OLAP leaf-level member. This is not to say that you couldn’t build a DB2 OLAP cube at a higher level using the ‘Build to here’ function in the IS metaoutline. Considering the DB2 OLAP ‘uniqueness’ requirement, using the primary key is a nice fit.

Along with the DB2 OLAP leaf-level member, a column should also be defined for the associated ‘ALIAS’ and ‘User Defined Attribute (UDA)’. The first three columns, therefore, define the DB2 OLAP leaf-level member. When designing the dimension table, keep DB2 OLAP requirements in mind. For example, DB2 OLAP limits the field size of member names and aliases to 79 characters. This limitation should be included in the dimension table design.

Dimension table: hierarchies

IS supports various table layouts to build DB2 OLAP hierarchies. One table layout is preferred. This format is a hybrid generation build. Basically stated, the primary key remains as the first column (+ the associated alias and UDA, if applicable). The next group of columns represent a ‘generation build’, with GEN01 being the highest parent, GEN02 the second highest parent(s) and so forth. We have to continue building the generations in columns to the right, and not to include the leaf-level member [the primary key] in any of the generation columns.

A sample of this dimension table layout is shown in Table 21.

Table 21. Sample dimension table with hierarchies

PRODUCT_ CODE	PRODUCT_ALIAS	GEN_01	GEN_02
100_10	COLA	COLA VS NONCOLA	COLAS
100_20	DIET COLA	COLA VS NONCOLA	COLAS
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA

PRODUCT_CODE is the primary key. GEN_01 is the highest parent of the primary hierarchy. GEN_02 is the second highest parent of the hierarchy. If aliases and UDAs are required for the parent levels then add the necessary columns to the right of the associated generation columns.

Dimension table: alternate hierarchies

Alternate hierarchies are a very powerful DB2 OLAP function. The preferred method for handling alternate hierarchies is to build the alternate hierarchy structure in columns to the right of the primary hierarchy. Not all leaf-level members, or primary keys of the dimension table, are members in alternate hierarchies. For those members, the alternate hierarchy columns should be left ‘NULL’.

A sample of a dimension with a primary hierarchy and an alternate hierarchy is shown in Table 22.

Table 22. Dimension table with alternate hierarchies

PRODUCT_ CODE	PRODUCT_ ALIAS	GEN_01	GEN_02	ALT_GEN _01	ALT_GEN _02
100_10	COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	REGULAR
100_20	DIET COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	DIET
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	REGULAR
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	REGULAR
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	DIET
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	REGULAR
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	DIET
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	REGULAR
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	DIET
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	REGULAR
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	DIET

Dimension Table: ragged hierarchies

One of the most difficult hierarchies to build is the ragged hierarchy. A ragged hierarchy is a hierarchy in which there are an unequal number of levels in the hierarchy. One parent may have two (2) levels prior to the leaf-level member; another parent may have five (5) levels prior to the leaf-level member.

To build the dimension table for a dimension with ragged hierarchies, use the same method outlined above. With GEN_01 as the top level parent, each column to the right contains the next parent. When the leaf-level member is the ‘next’ member in the hierarchy, enter a ‘NULL’ value. Therefore, in the example Table 22, the first parent would have values in Gen_01, and GEN_02 and NULL values in the GEN_03 through GEN_05. The second parent would have values in GEN_01 through GEN_05. The leaf-level values do not appear under any of the GEN_XX columns.

A sample dimension with a ragged primary hierarchy is shown in Table 23.

Table 23. Dimension table with ragged hierarchies

PRODUCT_ CODE	PRODUCT_ ALIAS	GEN_01	GEN_02	GEN_03	GEN_04	GEN_05
100_10	COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
100_20	DIET COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA	GRAPE FAMILY	GRAPE REGULAR	SAMPLE 1
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA	GRAPE FAMILY	GRAPE DIET	SAMPLE 2
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA	ORANGE FAMILY	ORANGE REGULAR	SAMPLE 3
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA	ORANGE FAMILY	ORANGE DIET	SAMPLE 4

The snowflake question

Questions regarding the use of a ‘snowflake’ structure instead of a standard star schema always come up when discussing dimension hierarchies. A snowflake structure contains branches from one or more dimension tables. For example, the PRODUCT dimension could have an alternate hierarchy for ‘FAMILY’. The hierarchical information for the alternate hierarchy would be stored in a separate table. The FAMILY table would be a branch from the PRODUCT dimension table.

A sample of a snowflake dimension is shown in Figure 46.

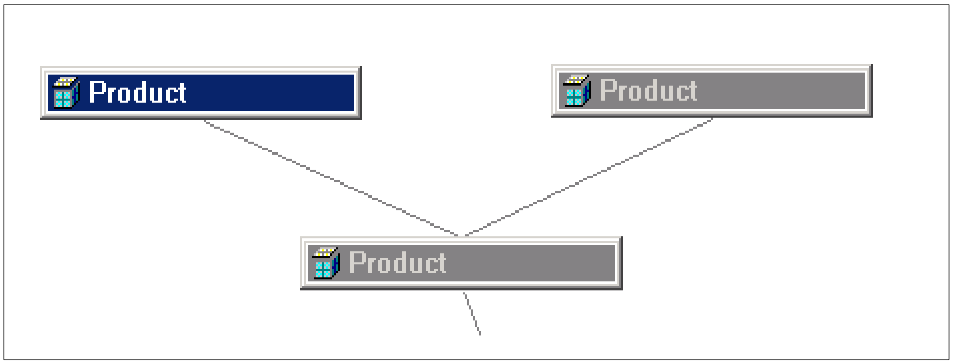


Figure 46. Snowflake example

A non-snowflaked star schema would only contain a single PRODUCT dimension table with primary and alternate hierarchies defined in the single table.

We advise against ‘snowflaking’ because of the additional SQL time required to run the queries. Snowflaking schemas require additional SQL time to consider the multiple joins between the dimension tables and the snowflaked tables. Performance time is one of the key considerations in the IS and DB2 OLAP engagement.

B.3.1.3 Star schema indexing strategy

While each star schema is different, there is a basic indexing strategy that can be employed during the preliminary phases of the IS implementation. Keep in mind that indexes are expensive in relation to the space the indexes require. It is not unusual for the indexes to take up as much space as the data.

The indexing strategy can make or break the IS implementation.

The establishment of the primary and foreign keys identifies these keys as unique indexes.

For a preliminary indexing strategy, you should index each column that is contained in the SELECT statement generated by IS. You should make sure that you review both the metadata SQL statement and the data load SQL statement to identify possible index candidates. In addition, if drill-through reports are also in use, review the SQL statements generated by the drill-through reports and index appropriately.

B.3.1.4 Surrogate keys

Surrogate keys are not required by IS but they are just plain ‘Good data warehousing’ and should be implemented in all star schemas. What is the definition of a surrogate key? Per Ralph Kimball:

“In a data warehouse, a surrogate key is a necessary generalization of the natural production key and is one of the basic elements of data warehouse design. Every join between dimension tables and fact tables in a data warehouse environment should be based on surrogate keys, not natural keys.”

Basically it means replacing your natural keys with a unique sequentially assigned integer. There are many reasons for using surrogate keys but the most important to IS is the improved query performance time.

For additional information regarding the use of surrogate keys please see any of the data warehousing books by Ralph Kimball or search the DBMSMag.com Web site for articles regarding the use and deployment of surrogate keys.

B.4 IS installation and environment configuration

The DB2 OLAP / IS / Star schema environment can vary greatly depending on the client environment. Answering the following questions can help you determine the most advantageous environment for DB2 OLAP and IS.

1. Obtain the following server specifications:

   a. Server RAM
   b. Number of processors on the server
   c. Amount of hard drive space
   d. Is this a dedicated server or are other applications also using this server?

   - Identify the applications that share the server and usage percentage.

2. How many DB2 OLAP applications are running on this server?

3. For the DB2 OLAP cubes on this server, how often are they updated? How often are calc scripts being run? How long do the calculations take to complete?

4. Is IS being used to generate and load IS cubes?

5. How often is the star schema being updated? How long does it take to populate or update the star schema?

6. Has the client Network performance been tested? Are there known problems with the network?

Answering these questions for each of the servers available for this project will help determine the optimal DB2 OLAP / IS / Star schema configuration. Also keep in mind the following information:

1. When DB2 OLAP runs a calculation, one processor is in use. A second calculation will take a second processor. On a dual processor server, two DB2 OLAP calculation can already have created a queue (the Operating System requires some processor time as well).
2. When IS is running either a meta data load or a data load, a processor is in use.
3. When the RDBMS is running the IS-generated SQL, a processor is in use.
4. When IS generates a DB2 OLAP cube, a metadata build, the SQL is run and all of the result set is stored prior to being loaded into DB2 OLAP. The IS server handles the DB2 OLAP validation rules prior to loading the metadata into DB2 OLAP.
5. When IS loads data into DB2 OLAP, the SQL is generated and data is loaded into DB2 OLAP in 100 record increments. The 100 record increment can be modified.

B.4.1 IS environment configuration recommendations

Based on the above information, the following recommendations should be considered:

1. DB2 OLAP / IS / Star schema can all reside on the same server if the following conditions apply:
   a. The server specifications include a quad processor, 2 GB of RAM and a maximum amount of hard-drive space.
   b. There are a maximum of two DB2 OLAP cubes on the server. The DB2 OLAP cubes are updated and calculated monthly.
   c. The Star Schema is updated incrementally in a reasonable time frame.

   In this scenario, there should be limited queuing because of the number of available processors and RAM. The process may look like this:

   - The Star schema is updated (it uses one processor + required RAM).
   - IS loads the metadata and data into DB2 OLAP (after the star schema is updated, it uses one processor + required RAM).

    - DB2 OLAP calculates the cube (it uses one processor + required RAM).

    Loading two DB2 OLAP cubes would obviously take twice the number of resources. In this case, we still would not anticipate queuing or performance issues.

2. IS / Star Schema on one server and DB2 OLAP on a separate server if the following conditions apply:

    a. The server specifications include a dual processor, 2 GB of RAM and a maximum amount of hard-drive space.

    b. There are more than two DB2 OLAP cubes in production or two DB2 OLAP cubes that are updated more than on a monthly basis. The DB2 OLAP cubes should reside on a separate server.

    c. The star schema is updated more than once a month for example weekly updates.

    In this scenario, there should be limited queuing because we split the processes that require processor time and RAM. The process may look like this:

    - The star schema is updated (it uses one processor + required RAM) each week
    - IS loads the metadata and data into DB2 OLAP (after the star schema is updated, it uses one processor + required RAM), again weekly.
    - DB2 OLAP calculates the cube (it uses (one processor + required RAM) * number of DB2 OLAP cubes). Since DB2 OLAP is running on another server, the number of processors and RAM required to run multiple calculations does not effect IS or to update the star schema.

3. Consider having IS, star schema and DB2 OLAP all on separate servers if the following conditions apply:

    a. The servers are not dedicated servers. If this is the case, it is essential that you ‘load balance’ your applications. You must understand the other applications running on each server and balance their RAM and processor requirements with the IS / DB2 OLAP and star schema processor and RAM requirements.

    b. IS loads the metadata and data to multiple DB2 OLAP cubes on a daily basis.

    c. End-user drill-through activity is high.

    d. Multiple DB2 OLAP cubes are calculating daily.

B.5 The IS model

From this point forward, this document will only point out good practices and provide supplemental information to the IS documentation. The IS documentation should be reviewed for the step-by-step methods of building the IS model and IS metaoutline.

B.5.1 Building the IS model

When building the IS model, incorporate all dimensions defined in the star schema into the IS model. Many times this could be fifteen or twenty (or more) dimensions! By including all possible dimensions in the IS model, you are guaranteeing the referential integrity of the DB2 OLAP outlines (this assumes, of course, that the star schema has the required primary and foreign key relationships defined!).

B.5.1.1 The TIME dimension

The first step in building the IS Model is to drag the FACT table to the center of the right panel. When asked if you want to add a TIME dimension, the answer should be ‘NO’.

Why? The FACT tables generally contain millions and millions of records. By using the FACT table to build the TIME dimension, you would basically be doing a “Select distinct” on the TIME column of the FACT table in order to populate the TIME dimension. Instead, create a separate TIME dimension table and join to the FACT table using the surrogate key. This step will greatly influence performance.

Dynamic TIME Series

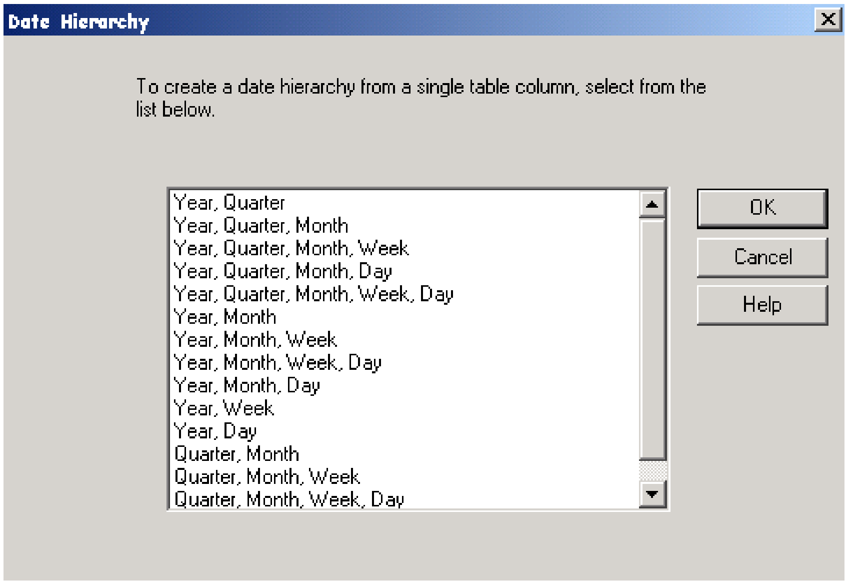
Creating hierarchies and employing Dynamic TIME Series functionality is accomplished in the following steps:

1. Create a TIME Dimension table that contains a column for the surrogate key to the FACT table and, using a DATETIME datatype, a column that contains the month end (or any particular) date (see Table 24).

Table 24. Create a time dimension table

TIME_KEY	TIME
1	1/31/2001
2	2/28/2001
3	3/31/2001
4	4/30/2001
5	5/31/2001
6	6/30/2001
7	7/31/2001
8	8/31/2001
9	9/30/2001
10	10/31/2001
11	11/30/2001
12	12/31/2001

2. Drag the TIME Dimension table onto the IS desktop and add the TIME dimension. Join the TIME dimension to the FACT table. When “Creating a New Dimension”, make sure you identify the TIME dimension as a ‘TIME’ Dimension Type.
3. Highlighting the TIME Dimension, select View /Properties / Tables.
4. In the Table Properties dialog box, select the ‘COLUMNS’ tab. Select the DATETIME column and then select the Date-Hierarchy box on the right side.
5. The Date-Hierarchy dialog box is retrieved as shown in Figure 47.



To create a date hierarchy from a single table column, select from the list below.

Figure 47. Creating date hierarchies

6. Select the TIME hierarchy that is appropriate.
7. After the Date-Hierarchy selection is made, the TIME Dimension is altered to appear as in Figure 48, adding three extra columns to the TIME dimension table: Month, Quarter, and Year.

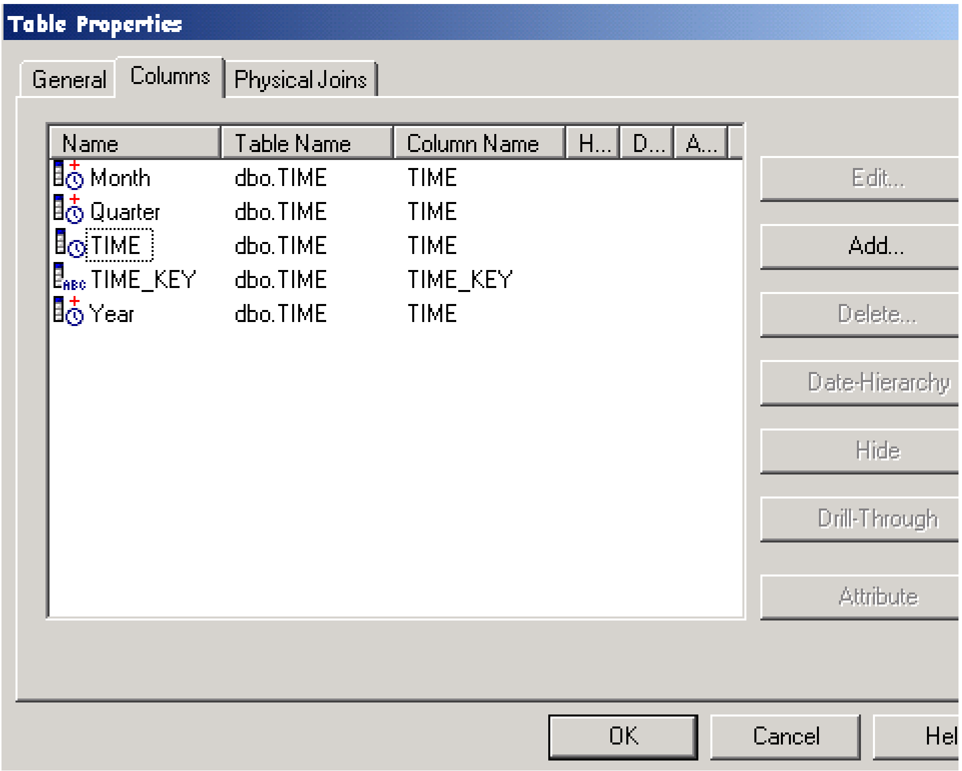


Table Properties

General | Columns | Physical Joins

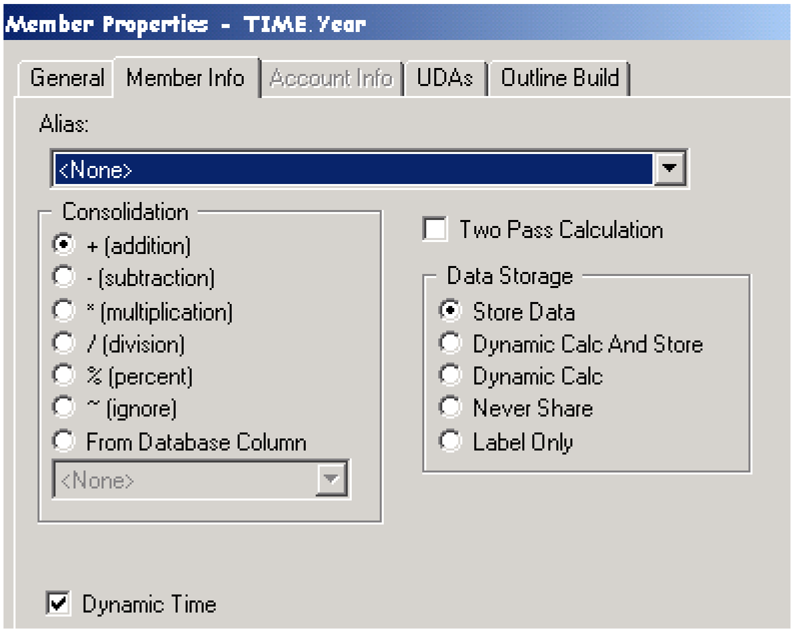
Name	Table Name	Column Name	H...	D...	A...
Month	dbo.TIME	TIME			
Quarter	dbo.TIME	TIME			
TIME	dbo.TIME	TIME			
TIME_KEY	dbo.TIME	TIME_KEY			
Year	dbo.TIME	TIME			

Figure 48. Time dimension altered

8. The following steps are made in the IS Metaoutline:

   a. Highlighting the YEAR level in the TIME Hierarchy, go to the Member Properties and select the DYNAMIC TIME box (see Figure 49).

   b. Complete the previous step for the QTR level in the TIME hierarchy.



Alias: <None>

Consolidation

● + (addition)
○ - (subtraction)
○ * (multiplication)
○ / (division)
○ % (percent)
○ ~ (ignore)
○ From Database Column

<None>

□ Two Pass Calculation

Data Storage

● Store Data
○ Dynamic Calc And Store
○ Dynamic Calc
○ Never Share
○ Label Only

☑ Dynamic Time

Figure 49. Selecting dynamic time

B.5.1.2 The ACCOUNT dimension

If you are using Method #1to build the FACT table (MEASURES are reflected as columns in the FACT Table), then you have to answer 'YES' when asked if you want to build the ACCOUNT Dimension after adding the FACT table.

If you are using Method #2 to build the FACT table (MEASURES are reflected in a single column in the FACT Table), then you have to answer 'NO' when asked if you want to build the ACCOUNT dimension after adding the FACT table. We need to identify the specific ACCOUNT column in the FACT table for IS. Then, we drag it over the ACCOUNT dimension table and we identify the dimension as a generic dimension:

You will have to identify the ACCOUNT dimension as an 'Accounts' type dimension in the metabolite.

B.5.1.3 Generic dimension tables

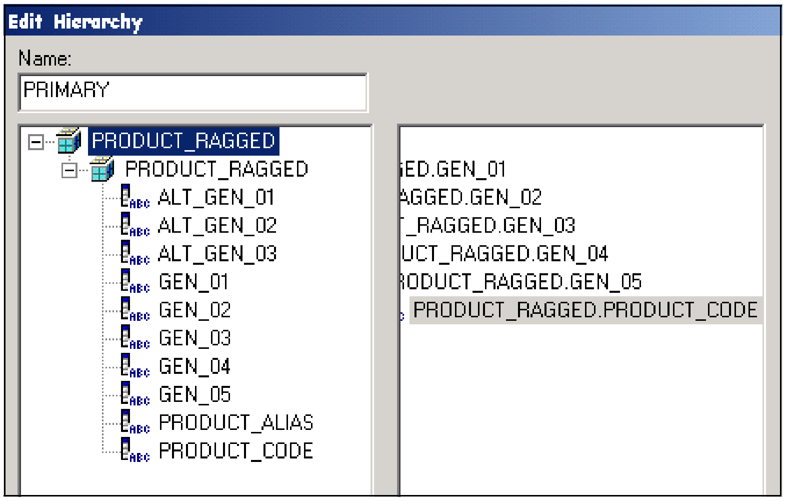
When building generic dimension tables, you should have to consider transformations and to build ragged hierarchies.

Transformations

Consider the SQL performance cost of using the FIND and REPLACE functionality available under the TRANSFORMATIONS tab. If the FIND and REPLACE function is required for all possible DB2 OLAP cubes (for instance, replacing an imbedded “with a single ‘), you should consider handling this in the extract, transform and load stage or prior to loading the star schema. Cleansing the data once on the way into the star schema is much more efficient than transforming the data each and every time it is loading into DB2 OLAP using IS.

Building ragged hierarchies

With the recommended star schema format, building ragged hierarchies with IS is simple! From the Edit hierarchy dialog box, you drag over each GEN_XX column, top to bottom order (see Figure 50).



Name:
PRIMARY

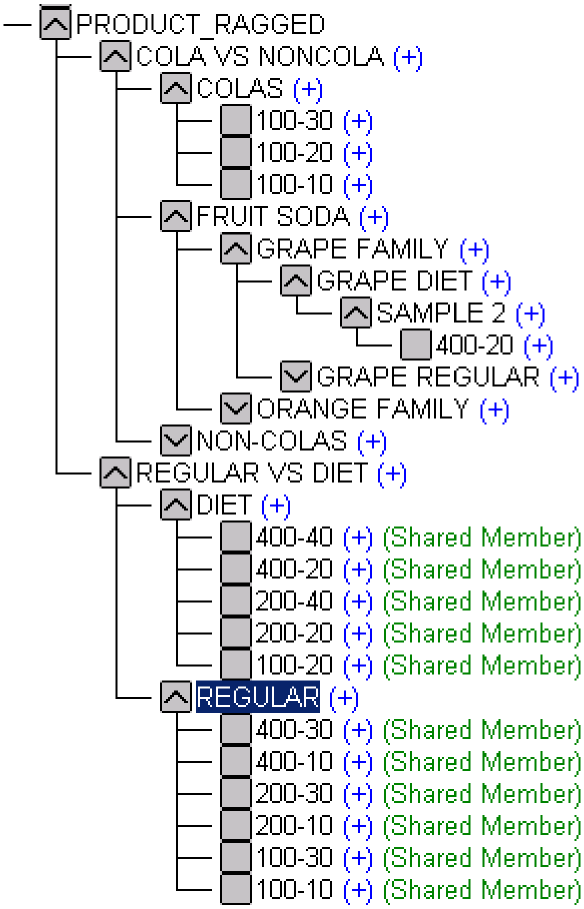
ED.GEN_01
AGGED.GEN_02
_RAGGED.GEN_03
UCT_RAGGED.GEN_04
ODUCT_RAGGED.GEN_05
PRODUCT_RAGGED.PRODUCT_CODE

Figure 50. Building ragged hierarchies

The last generation should be the leaf-level member.

The <NULL> values contained in any of the GEN_xx columns will be ignored. The resulting RAGGED_PRODUCT dimension generated in DB2 OLAP looks like Figure 51.

Again, an efficiently build star schema dimension table makes the IS implementation of difficult ragged hierarchies a simple process.



```
PRODUCT_RAGGED
  COLA VS NONCOLA (+)
    COLAS (+)
      100-30 (+)
      100-20 (+)
      100-10 (+)
    FRUIT SODA (+)
      GRAPE FAMILY (+)
        GRAPE DIET (+)
          SAMPLE 2 (+)
            400-20 (+)
        GRAPE REGULAR (+)
      ORANGE FAMILY (+)
    NON-COLAS (+)
  REGULAR VS DIET (+)
    DIET (+)
      400-40 (+) (Shared Member)
      400-20 (+) (Shared Member)
      200-40 (+) (Shared Member)
      200-20 (+) (Shared Member)
      100-20 (+) (Shared Member)
    REGULAR (+)
      400-30 (+) (Shared Member)
      400-10 (+) (Shared Member)
      200-30 (+) (Shared Member)
      200-10 (+) (Shared Member)
      100-30 (+) (Shared Member)
      100-10 (+) (Shared Member)
```

Figure 51. Resulting ragged dimension

B.6 IS metaoutline

All of the dimensions from the star schema, and therefore the IS model are available now to be included in the DB2 OLAP outline.

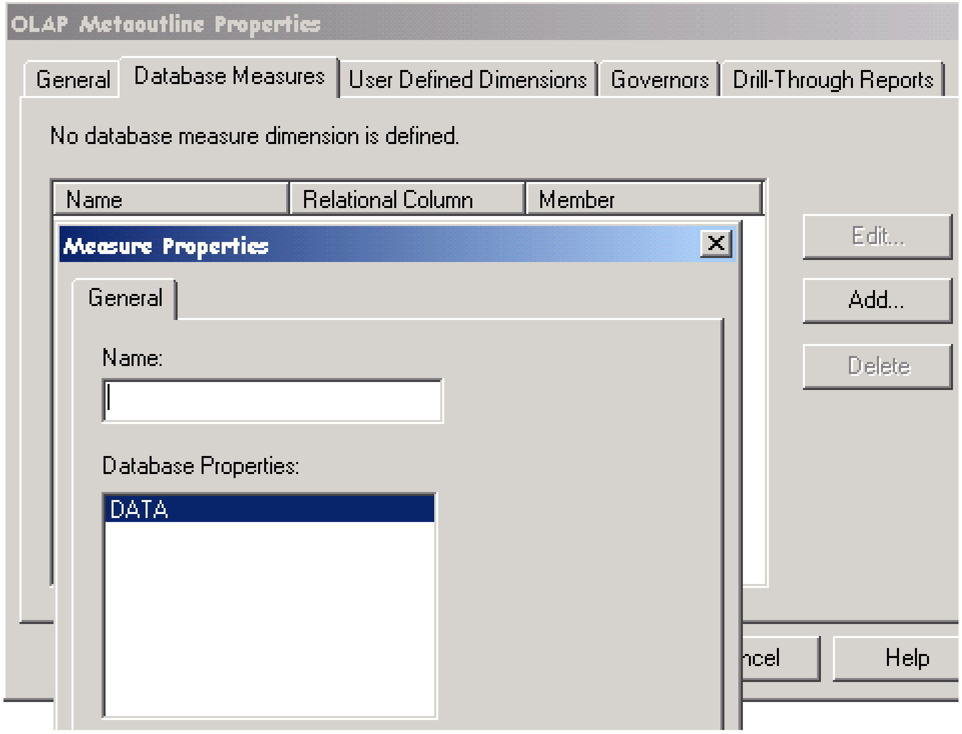
B.6.1 Building the IS metaoutline

When you create a new metaoutline, all the available dimensions appear on the left side panel. In addition to the hierarchies established in the IS model, the IS developer can also build hierarchies from scratch in IS metaoutline. The hierarchies built in IS metaoutline are only available for that particular metaoutline.

B.6.1.1 The MEASURES dimension

If you’ve used Method #2 to define the MEASURES dimension, you’ll need to make one modification to the metaoutline in order for IS to recognize the MEASURES column.

To define the MEASURES column, highlight the metaoutline name and right mouse click. Select ***Properties*** from the drop-down menu. Select the ***Database Measures*** tab and then the ***ADD*** button (see Figure 52).



No database measure dimension is defined.

Database Properties: DATA

Figure 52. Defining the MEASURES dimension

Any numeric datatype will appear under the Database Properties list. Select the column that contains the data for the MEASURES dimension. When dragging over the ACCOUNT dimension, make sure you identify this dimension as the ‘ACCOUNTS” dimension type.

B.6.1.2 Generic dimensions

Rules regarding transformations, specifically the FIND and REPLACE function, mentioned in B.5.1, “Building the IS model” on page 187 above apply to the IS metaoutline also. Each and every time you use the TRANSFORMATION functions, you are adding time to your metadata and data load into DB2 OLAP.

Concatenation and substring

In both the IS model and IS metaoutline, concatenation and substring functionality is available when viewing a table. Prefixing and suffixing is available under the Transformation button.

If you find that a column by itself is meaningless and therefore requires a concatenations or substring function to make sense, consider making this a global change and handling this transformation during the extract, transform and load (ETL) process. It’s much more efficient to handle this process once during the star schema load then multiple times every time an DB2 OLAP cube is built and loaded.

Filtering and building hierarchies

Filtering on specific column members make IS extremely flexible when building hierarchies. Filters create a ‘where’ clause in the generated SQL. You can filter on any column associated with the particular dimension. If using a filter parameter allows you to build your hierarchy simply but the filter-able data is not in your dimension table, consider adding a column to the dimension in the star schema.

B.6.2 Loading data

Building the star schema and the hierarchies as detailed in the document will result in improved data load performance. Here’s why: defining the primary and foreign relationships (the referential integrity of the star schema) as the IS model joins creates a very simple ‘WHERE’ clause in then SQL generated for the data load. Other methods, particularly when there is no star schema referential integrity, create very inefficient SQL scripts for the data loads resulting in poor performance.

As a hint for the indexing strategy of the star schema, you should:

- Review the SQL generated during the data load. It can be found in the olapisvr.log, contained in the IS/ BIN subdirectory.
- Verify that each of the ‘SELECT’ columns have been indexed, at a minimum.

- Conduct benchmark testing to determine the cost (additional tablespace required) of additional indexes is worth the benefit (how much faster is the dataload with the new index?).

B.6.2.1 Granularity of the FACT table versus data load performance

IS will sum the MEASURES data during the data load process. For every unique occurrence of the dimension string, the MEASURES are summed together. For instance, if the FACT table is populated at the invoice line-level detail and DB2 OLAP is loaded at a monthly level, IS will sum the FACT table up to the month. If the data load performance time is unacceptable, consider creating an ‘Aggregate FACT table‘. In DB2 this is called an Automatic Summary Table (AST). By aggregating the FACT table during the star schema build, you effectively reduce the total number of columns in the FACT table and the SQL performance time required for the SUM function. You have to keep the detail FACT table joined to the star schema for drill-through purposes.

B.6.3 IS drill-through functionality

IS drill-through functionality is a major function of IS. One simple fact to remember:

IS drill-through reports allow the end-user to drill-through to the detail contained in the star schema.

Drill-through reports do not drill back to the source relational databases: drill-through reports drill back to the star schema. Any and all tables that are part of the star schema and the IS model are available for IS drill-through.

B.6.3.1 Drill-through report details

When creating a drill-through report in IS metaoutline, you define two things:

1. The DB2 OLAP intersection points which are the elements of the ‘WHERE’ clause
2. The columns or data to be returned which are the columns identified in the SELECT statement.

The most important things to remember concerning the creation of drill-through reports are:

1. When defining the DB2 OLAP intersection points, select the leaf-level members from the FACT table rather than the leaf-level member from each of the dimension tables. This should reduce the complexity of the SQL script and therefore improve performance during retrieval.

2. When defining the data columns to retrieve, select the columns from the least number of tables. Again, by restricting the number of tables and reducing the number of joins, you will improve performance time during retrieval.
3. Any changes made to a drill-through report require you to rebuild the DB2 OLAP outline. The DB2 OLAP outline file (.otl) contains the information necessary for drill-through. Changes to drill-through reports without rebuilding the DB2 OLAP outline will result in the drill-through reports not changing.
4. Consider modifying the star schema indexing strategy to incorporate drill-through requirements.

Appendix C. Case study: example of end-to-end approach

To illustrate tools integration in an OLAP end-to-end approach, we implemented a case study based on a system defects application.

C.1 Context and business scenario

A worldwide software company SoftCompany has numerous Customer Interaction Centers (CICs) which receive defect cases on products from their customers. The CICs are located throughout the world and share a common transaction system to record details on customer requests. Customer requests may be registered in the transaction system by the call receiver, via the Web, or via an automated telephony application.

The company typically operates on a ‘call-back’ model where the first contact with the customer is usually to gather information and to create a case record in the transaction system. The call is then assigned to technical people with the appropriate skills to look at the information provided and to begin a dialog with the customer to answer his request. Once the customer request is satisfied, the case record is marked closed.

The company wants to categorize and analyze:

- The performance of its technical support people who answer a call:
  - Who solves the highest number of cases?
  - Who needs minimum days to resolve the case, on average?
  - How was the case solved?
- The customer calls:
  - Who is calling the most often?
  - When are customers calling the most often?
  - With what kinds of requests?
- The products impacted:
  - What products require the most technical support?
  - What products require the least technical support?

The OLAP solution must enable the company to find answers to all of these questions, and to anticipate others using a more complex analysis.

C.2 Proposed approach

We loaded input data provided by SoftCompany in a relational staging area on IBM DB2 Universal Database (DB2 UDB). We used DB2 UDB to easily manipulate, clean, and transform data in order to load a relational datamart (refer to “*The Data Warehouse Toolkit*” book from Ralph Kimball). The table structure in a star schema is especially well suited for multidimensional databases, and is the best way to prepare and structure relational data to be loaded in a multidimensional database.

A star schema consists of a FACT table which holds key figures of the business subject area and numerical measures. The FACT table relates these measures to different dimension tables which contain the context information for the recorded facts and holds the aggregation hierarchies to enable drill-down operations.

To be very flexible regarding the number of OLAP databases to implement and to be able to deal with different dimensionalities, we used IBM DB2 OLAP Integration Server (OIS) to define a logical OLAP model on the star schema datamart, prerequisite for OIS.

The OLAP model includes all the potential dimensions that we combined using the OIS metaoutline to create different OLAP datamarts for different testing purposes.

We used Intelligent Miner for Data for further knowledge about our input data and to find the useful dimensions for our OLAP design.

We used Analyzer as a reporting and write-back tool. Analyzer virtual cube functionality was selected over OIS drill-through reporting because it enables multi-pass drilling within the relational database.

Figure 53 details the whole architecture, and Figure 54 details the hardware and software configuration.

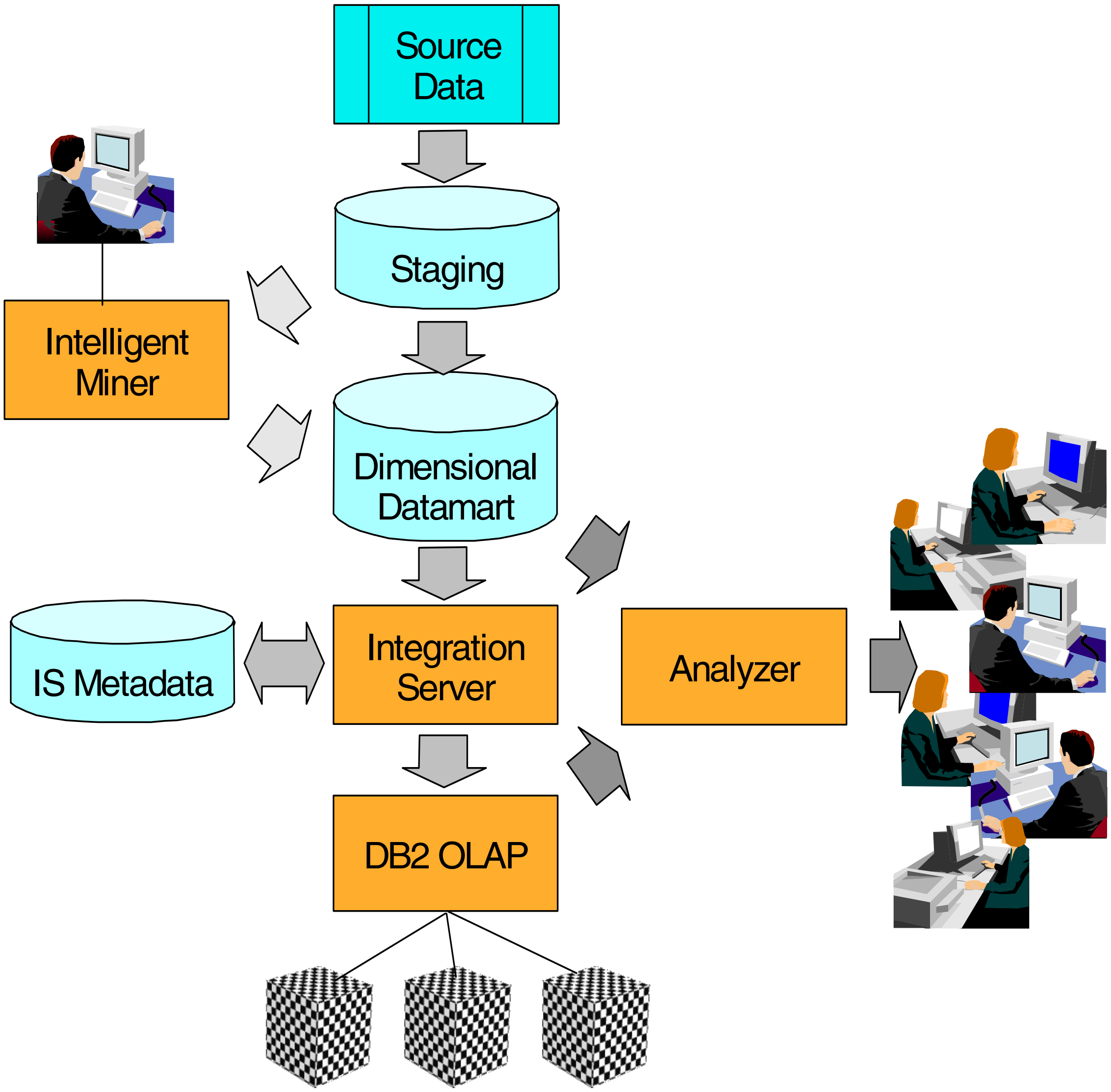


Figure 53. Products involved in case study

Concerning the hardware and software architecture, we used a J50 server from RS/6000 family with 1 GB memory and 4 PowerPC_604 200MHz processors. We installed DB2 V7.1, DB2 OLAP Server V7.1, DB2 OLAP Integration Server V7.1 and Intelligent Miner for Data V6.1. Source data is on RS/6000 server as well. As an end-user tool, we used Hyperion Analyzer Server V5.0.1 running on IBM PC 365 with Pentium Pro 200 MHz processor and 256 MB memory. The hardware configuration is shown in Figure 54.

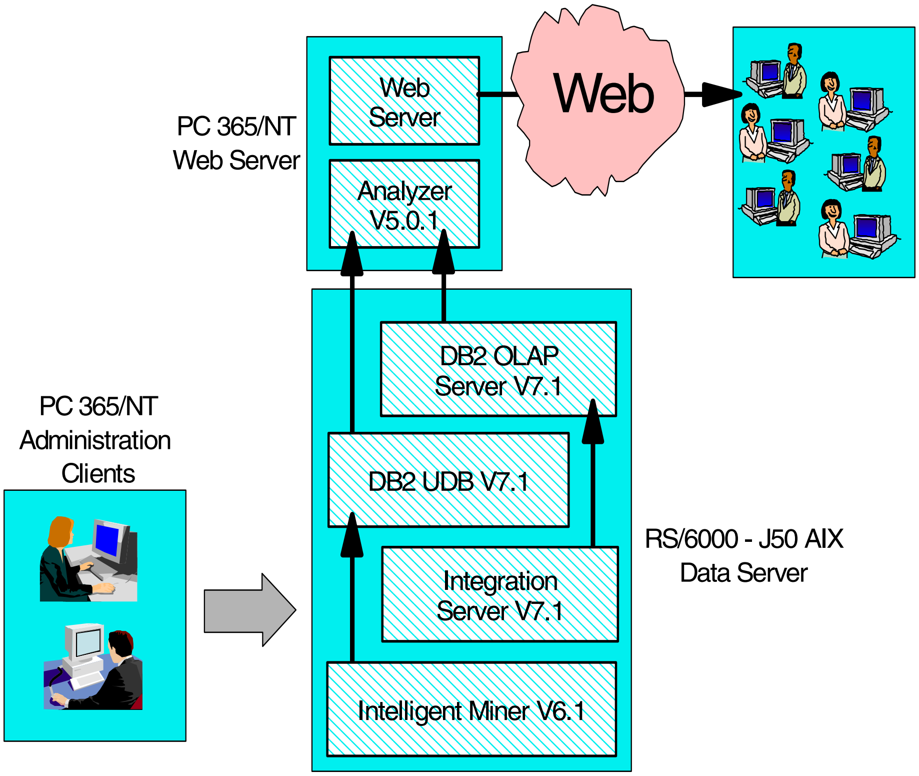


Figure 54. Case study hardware and software configuration

C.2.1 Building a relational datamart

To be able to clean and transform easily the input data, we imported source data into DB2UDB on AIX in a relational staging area.

To prepare the input data for OLAP databases using RDBMS capabilities and the strength of SQL, to best fit in the OLAP application, we built a star schema model.

The relational datamart created on this star schema model contains 1 FACT table and 14 dimension tables that you can see in Figure 55. The FACT table contains 48420 rows. Only 0.07% of the cases have status New, and 4.3% of them have status Open. All the other cases are closed.

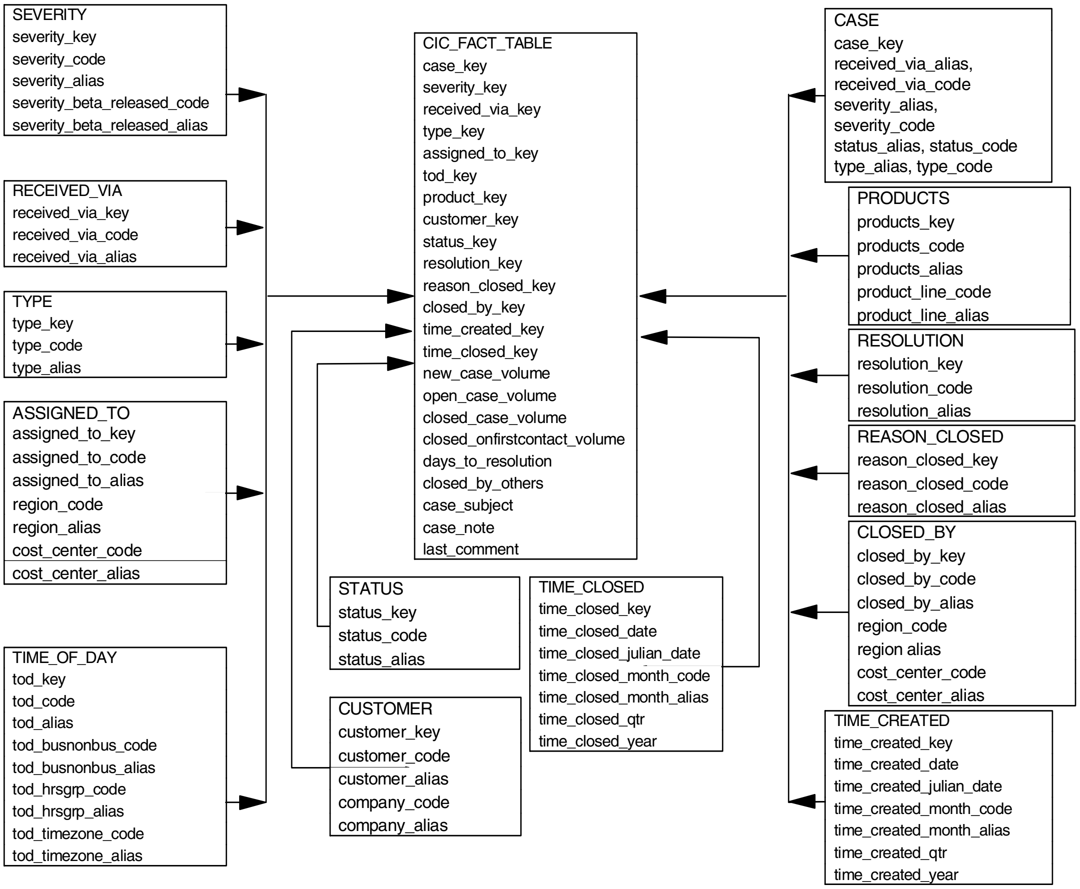


Figure 55. Star schema in datamart

We built this datamart with the knowledge of what should be dimensions and measures in our OLAP application.

C.2.2 Using Intelligent Miner for Data for designing OLAP dimensions

Data mining is the process of discovering previously unknown and ultimately comprehensible information from large stores of data.

Data mining is a complementary tool to DB2 OLAP when building and developing new OLAP models:

1. To uncover the most meaningful dimensions
2. To incorporate new dimensions
3. To define the most useful members, hierarchies, and attributes

C.2.2.1 Find out the most meaningful dimensions

We used Intelligent Miner for data to uncover the most meaningful dimensions. One of the most critical aspects in OLAP model design is the definition of the limited set of dimensions to be included in a particular cube. Including too many dimensions makes analysis difficult and affects performance and storage. The final analysis would be more efficient if the chosen dimensions are the most relevant, with high and predictive value.

In our call center case study we used bivariate statistics, neural networks, and factors analysis, in order to determine the most meaningful attributes for OLAP reports and to validate results from one technique with complementary approaches.

By combining these approaches we found that the eight most relevant attributes that should be included as dimensions for 'days to resolution' were:

- Type
- ClosedBy
- Time_Created
- AssignedTo
- Severity
- Closed_by
- ReceivedVia
- Days_to_resolution

In any case we should make sure there are no redundant attributes included in these analysis. Sometimes, if we suspect the existence of possible redundancies, it is worth while to perform specific correlation tests among candidate fields, in order to eliminate superfluous information.

C.2.2.2 Incorporate new dimensions

To incorporate new dimensions in our OLAP models, we can consider the following data mining tasks:

- Clustering
- Classification
- Prediction

In our case study we used clustering in order to find meaningful segments of cases within our population. We uncovered eight different segments for the set of cases, and observed their different profiles through Intelligent Miner visualization. For instance, specific segments uncovered correspond to cases where critical severity is much less than the average.

The results were incorporated as a new dimension in our OLAP outline in order to consider these new segments in our analysis.

For our case study, cases belonging to the same segments, represents cases that are similar in some way. Taking this into account, OLAP users now can filter information by these segments, observe what are their characteristics, and find out if one or more of them are interesting for further analysis. In Figure 56 we observe the list of the eight segments, global result of neural clustering.

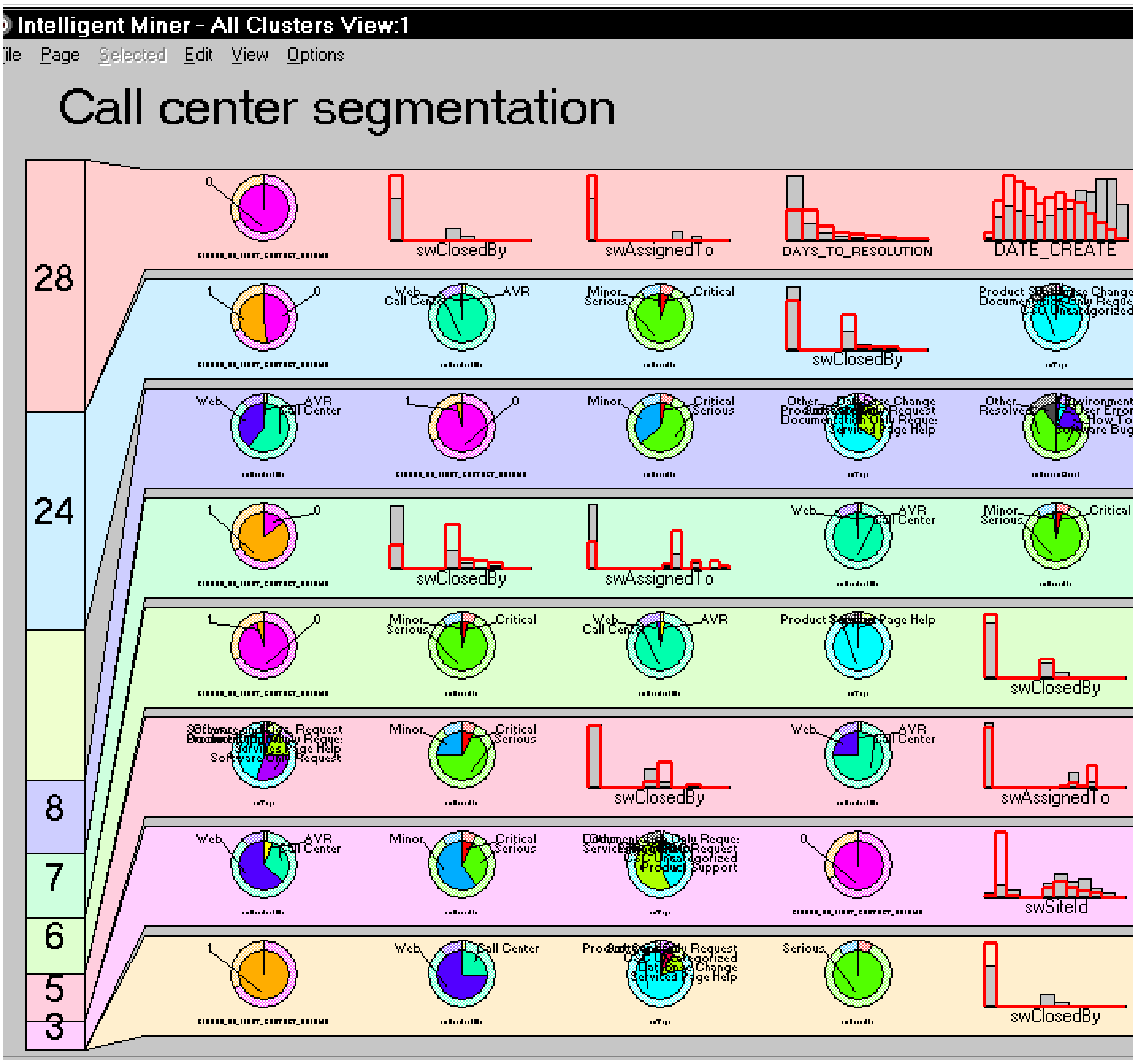


Figure 56. Clustering visualization for the case study

In Figure 57 we can see statistics about one of the clusters discovered. Cluster number 5 corresponds to 7.3% of our population of cases. We observe that in this cluster, cases are solved on average in less time that in the whole population.

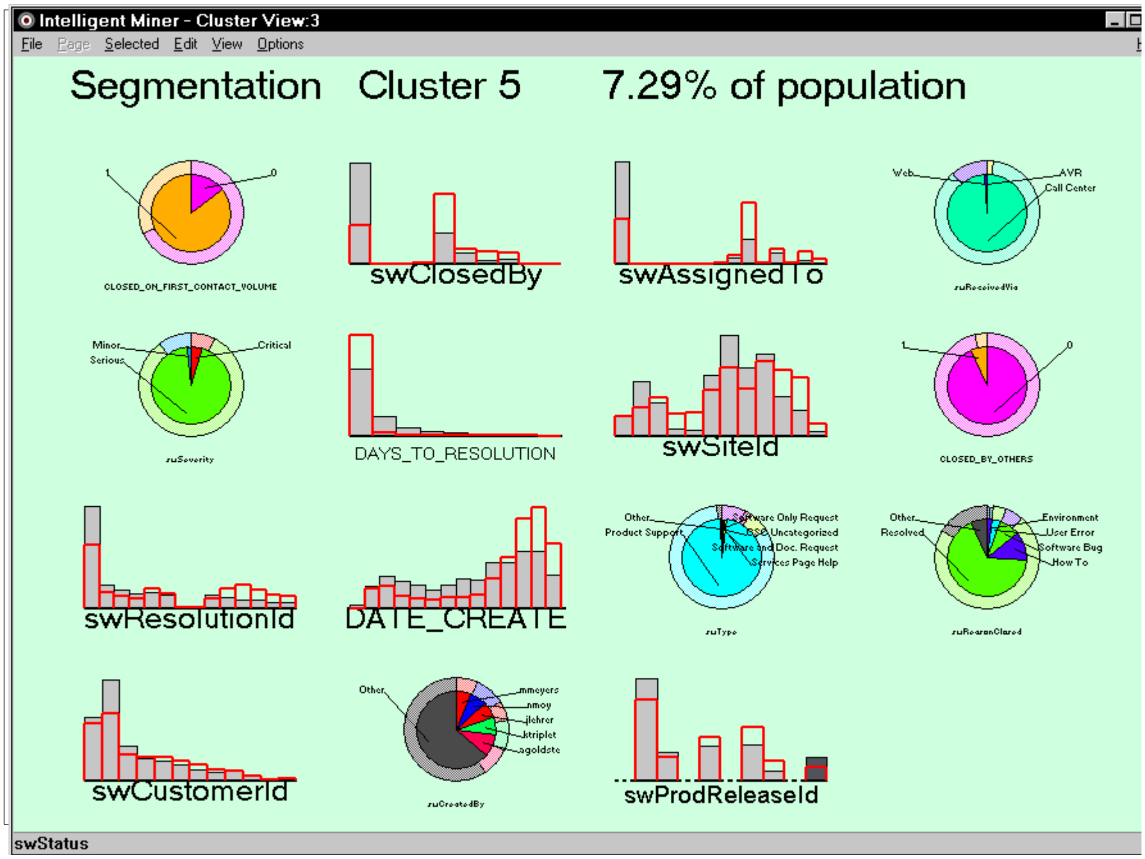


Figure 57. Segment example

C.2.3 Using OIS to build OLAP databases

In this section we discuss the use of OIS in building OLAP databases.

C.2.3.1 Defining a model

Of all available techniques for dimension building, using OIS is the most sophisticated and flexible. A prerequisite is to be connected to a star schema model datamart.

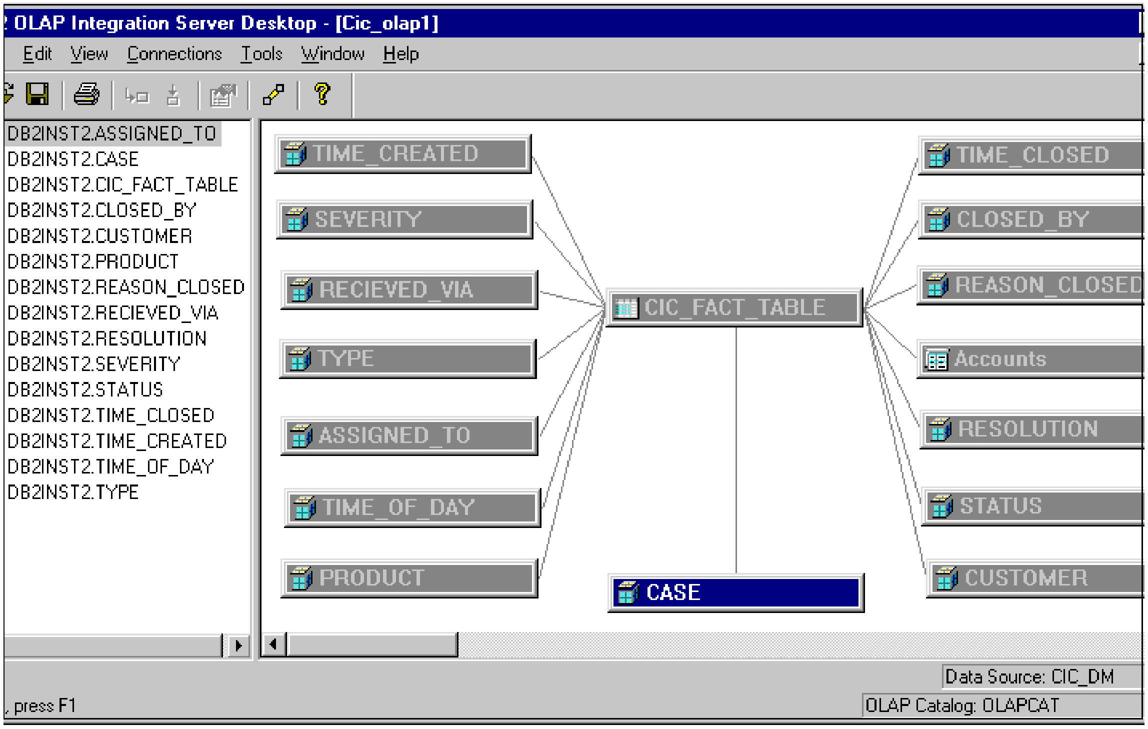
Figure 58 shows the CIC OLAP model we built from the case study.

We tried to do things as simply as possible. We used single tables, and we avoided joining tables, as well as manipulating and transforming data (as with SQL Interface), because this slows down the dimension building process.

We made sure the relational tables we were using as a source for dimension build were as ready to use as possible. Note that if there is no suitable table available, you can ask a database administrator to create one for you. If this is not possible, your second best option is to use the OIS transformation option.

As with using the SQL Interface to access source data stored in relational tables, the connection with the RDBMS is not native, but through ODBC.

The advantage of using OIS for dimension build over SQL Interface is that it enables you to reuse existing OLAP models and metaoutlines and therefore avoiding the need to write and maintain many SQL statements.



DB2INST2.ASSIGNED_TO
DB2INST2.CASE
DB2INST2.CIC_FACT_TABLE
DB2INST2.CLOSED_BY
DB2INST2.CUSTOMER
DB2INST2.PRODUCT
DB2INST2.REASON_CLOSED
DB2INST2.RECIEVED_VIA
DB2INST2.RESOLUTION
DB2INST2.SEVERITY
DB2INST2.STATUS
DB2INST2.TIME_CLOSED
DB2INST2.TIME_CREATED
DB2INST2.TIME_OF_DAY
DB2INST2.TYPE

Data Source: CIC_DM
OLAP Catalog: OLAPCAT

Figure 58. OLAP model from case study

C.2.3.2 Defining multiple metaoutlines

In our case study, we have created multiple metaoutlines to generate OLAP cubes based on the same OLAP model.

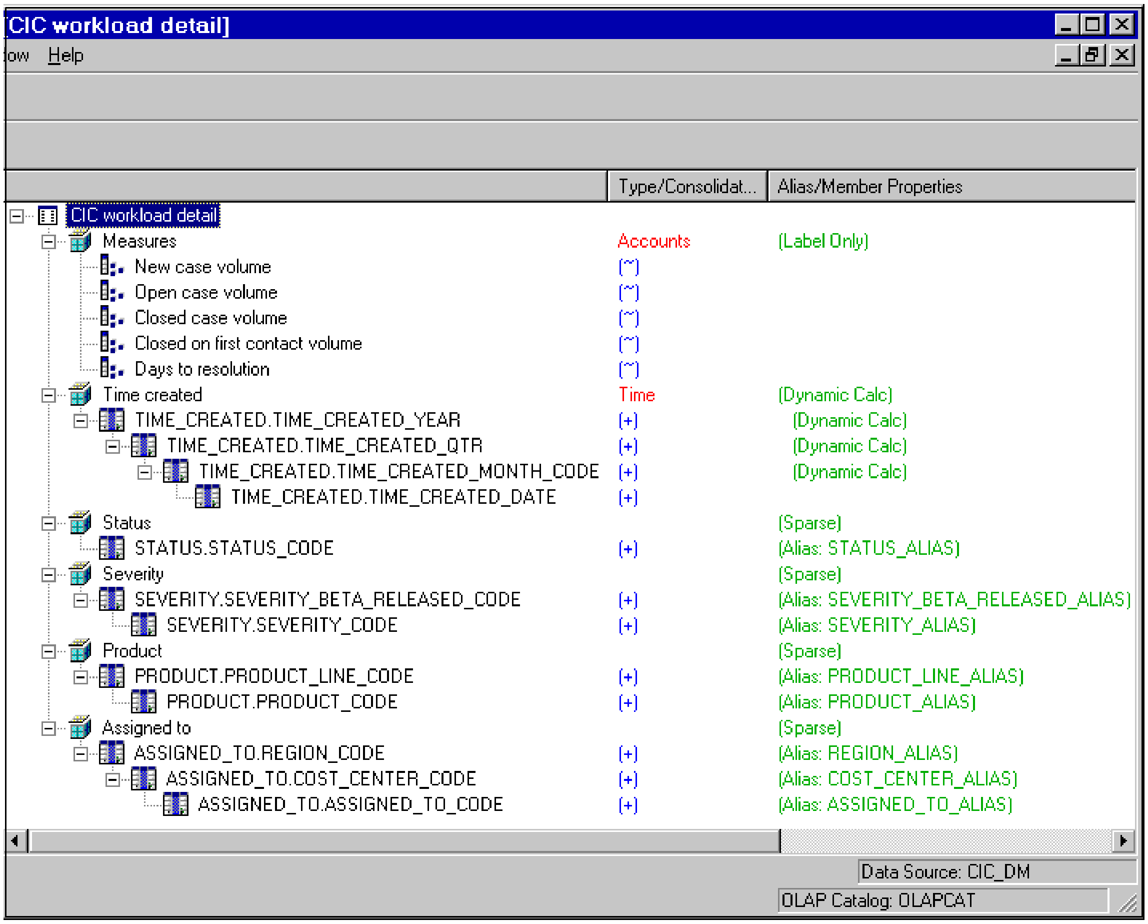
Here is what we did:

- For Technical Support management, to measure the performance of their technical people, we used the outline provided by Intelligent Miner, focusing on days_to_resolution, or the *CIC Workload Detail metaoutline* shown in Figure 59.
- For Sales people, before a customer visit, to check the statistics of all open and closed cases for the customer, customer calls, and their requests, we used the *CIC Customer* metaoutline shown in Figure 60.

  We used the dimensions RECEIVED_VIA, SEVERITY, and STATUS as attribute dimensions. Attribute dimensions need to have their base dimension and for that purpose we included CASE dimension. Status, severity, and type are attributes of every particular case, and therefore we had to include CASE as a dimension.

  We ordered the attribute dimensions Received via, Severity, Status in the Metaoutline after the sparse dimensions and according to their association with the base dimension Case.

  To be able to get detailed information on the customer and find a way to solve the case, we created 2 virtual cubes through Analyzer that we linked (see C.2.4, “Using Analyzer as a reporting tool” on page 211).

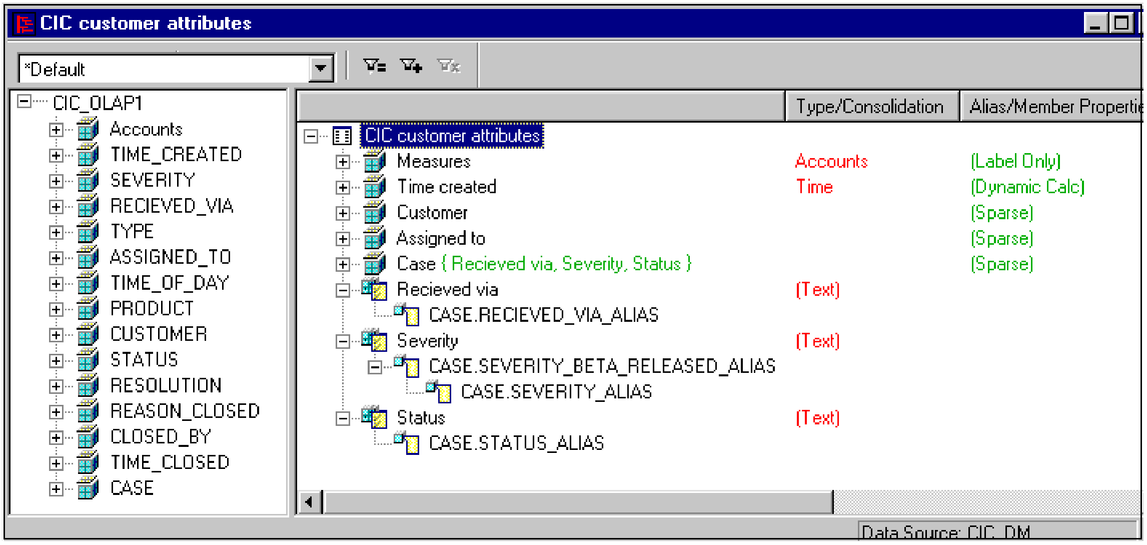


	Type/Consolidat...	Alias/Member Properties
CIC workload detail		
Measures	Accounts	(Label Only)
New case volume	(~)	
Open case volume	(~)	
Closed case volume	(~)	
Closed on first contact volume	(~)	
Days to resolution	(~)	
Time created	Time	(Dynamic Calc)
TIME_CREATED.TIME_CREATED_YEAR	(+)	(Dynamic Calc)
TIME_CREATED.TIME_CREATED_QTR	(+)	(Dynamic Calc)
TIME_CREATED.TIME_CREATED_MONTH_CODE	(+)	(Dynamic Calc)
TIME_CREATED.TIME_CREATED_DATE	(+)	
Status		(Sparse)
STATUS.STATUS_CODE	(+)	(Alias: STATUS_ALIAS)
Severity		(Sparse)
SEVERITY.SEVERITY_BETA_RELEASED_CODE	(+)	(Alias: SEVERITY_BETA_RELEASED_ALIAS)
SEVERITY.SEVERITY_CODE	(+)	(Alias: SEVERITY_ALIAS)
Product		(Sparse)
PRODUCT.PRODUCT_LINE_CODE	(+)	(Alias: PRODUCT_LINE_ALIAS)
PRODUCT.PRODUCT_CODE	(+)	(Alias: PRODUCT_ALIAS)
Assigned to		(Sparse)
ASSIGNED_TO.REGION_CODE	(+)	(Alias: REGION_ALIAS)
ASSIGNED_TO.COST_CENTER_CODE	(+)	(Alias: COST_CENTER_ALIAS)
ASSIGNED_TO.ASSIGNED_TO_CODE	(+)	(Alias: ASSIGNED_TO_ALIAS)

Data Source: CIC_DM

OLAP Catalog: OLAPCAT

Figure 59. OLAP metaoutline — workload detail


CIC customer attributes

*Default

```
CIC_OLAP1
  Accounts
  TIME_CREATED
  SEVERITY
  RECIEVED_VIA
  TYPE
  ASSIGNED_TO
  TIME_OF_DAY
  PRODUCT
  CUSTOMER
  STATUS
  RESOLUTION
  REASON_CLOSED
  CLOSED_BY
  TIME_CLOSED
  CASE
```

```
                                                      Type/Consolidation   Alias/Member Properti
CIC customer attributes
  Measures                                            Accounts             (Label Only)
  Time created                                        Time                 (Dynamic Calc)
  Customer                                                                 (Sparse)
  Assigned to                                                              (Sparse)
  Case { Recieved via, Severity, Status }                                  (Sparse)
  Recieved via                                        (Text)
    CASE.RECIEVED_VIA_ALIAS
  Severity                                            (Text)
    CASE.SEVERITY_BETA_RELEASED_ALIAS
      CASE.SEVERITY_ALIAS
  Status                                              (Text)
    CASE.STATUS_ALIAS
```

Data Source: CIC_DM

Figure 60. OLAP metaoutline — CIC Customer with attribute dimensions

C.2.4 Using Analyzer as a reporting tool

To create ad-hoc reports and charts from DB2 OLAP Server, we used Analyzer and we tested drill-through capabilities to access both relational and multidimensional data.

With Analyzer we can use two drill-through capabilities to navigate in relational data:

- We can build a view to access data in CIC Customer OLAP database to know the number of days of resolution and the number of closed cases by product (see Figure 61).

  From the results, we can access relational columns CASE_SUBJECT, CASE_NOTE, LAST_COMMENT to get the details on the case subject using OIS drill-through capability. To do so, when building a model in OIS, we need to define a drill-through report.
  OIS automatically generates the SQL, and reports are used via LRO (Linked Reporting Objects) in Analyzer and spreadsheets. We are not able to link multiple drill-through. For that reason, we included in the CIC_fact_table 3 more columns to detail the reasons to close and solve the case (CASE_SUBJECT, CASE_NOTE, LAST_COMMENT).

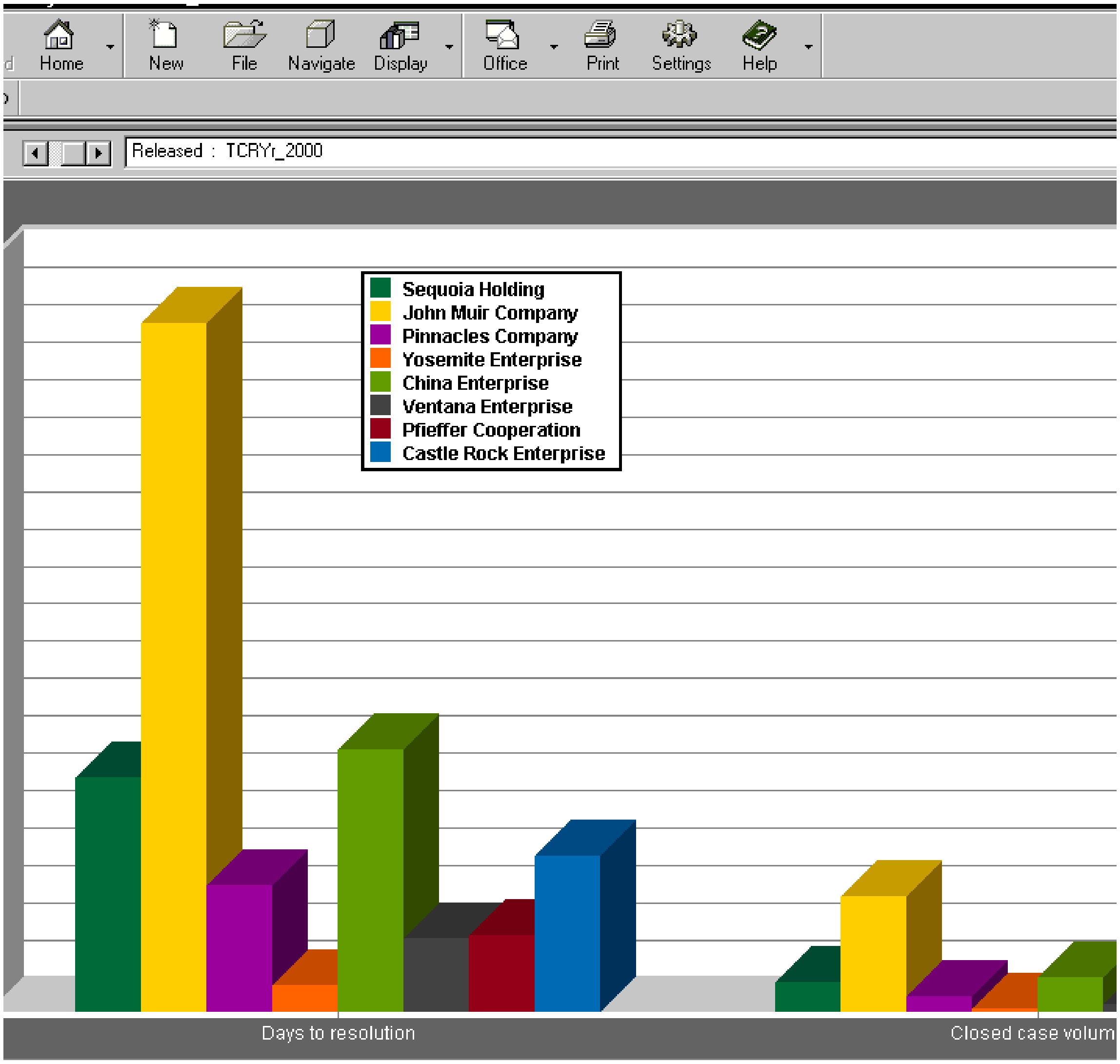


Figure 61. Performances per customer and per severity for 2000 year

We can use the virtual cubes capability directly provided by Analyzer. A virtual cube is an SQL select statement that returns data to Analyzer, enabling users to present relational data in multidimensional format. Before defining a virtual cube, a view needs to be pre-defined to transform, join, select, and aggregate the required data, making the SQL that defines the virtual cube a simple select.

Each virtual cube should operate directly against a single view. Performance can be improved by instantiating the view as an Automatic Summary Table. By linking views, users can jump between various relational and multidimensional data sources transparently. When drilling to a virtual cube View from another view (drill-linking), filters and page member selects are passed over as constraints within dynamically generated SQL “where” clauses. Only relevant records are retrieved.

Analyzer allows each individual view (report) to contain multiple links to other views. A link can be defined per dimension and/or at the data cell intersection level. For example, if we want to get the detailed information not only on the case resolution, but also on the customer, we can link together the virtual cubes and can drill-through to detail on multi pass (see Figure 62).

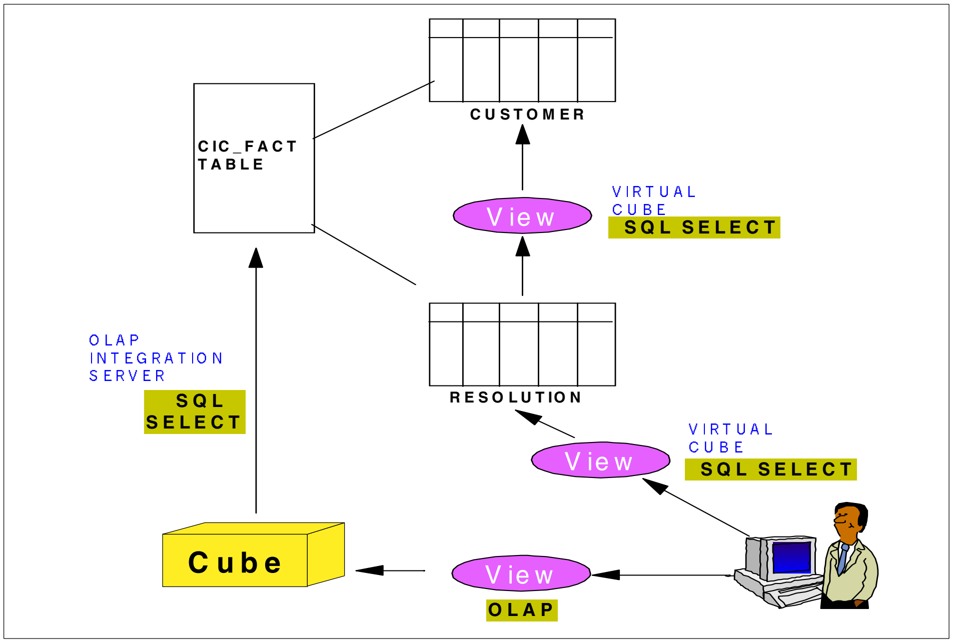


Figure 62. Drill-through capabilities

To build the virtual cubes :

1. Achieve the connectivity:
   - register the source relational datamart for ODBC as a system data source
2. Create a model using Analyzer Administrator for the first virtual cube (RESOLUTION)

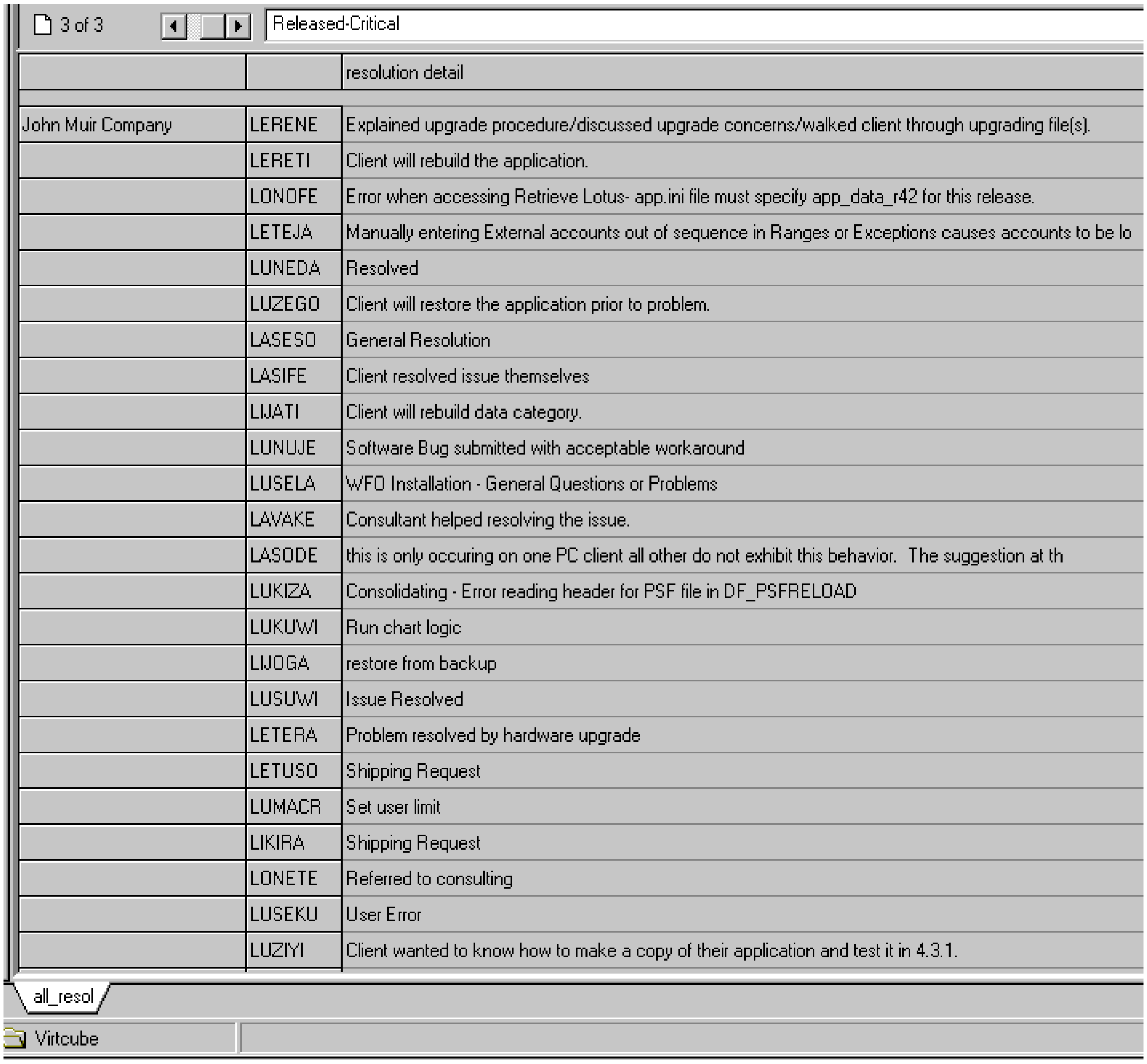
- Click Relational from the toolbar and insert a new model. The Model Name identified the virtual cube used in Analyser Client tool to create the views. The BDE Alias Name identified the relational database
- Type the SQL query.
- If the SQL query needs to find out the description of keys in lookup tables, first define the lookup tables and its SQL query and test them

```
(ex: select type_key,type_alias from type)
```

- Define for each column selected in the SQL request if it will be a dimension or a measure in the cube.
    - if we want to see items from the column as row or column headers in the Analyzer view, then we define it as a dimension description. Moreover if the column is a key column and we want its description, we define a lookup table (and the SQL select to select both code and description from the lookup table) and we select dimension code instead of dimension description.
    - if we want to see items from the column in the data cells of the Analyzer view, then we define it as a measure (For example, RESOLUTION_ALIAS is defined as a measure.)

  We need to define at least one column as a dimension and one column as a measure.

3. Test the model.
    - Click on Test.
4. Synchronize the model.
    - This step adds the virtual cube in the database list in Analyzer Administrator.
5. Manage access to virtual cube to users.
    - From Analyzer Administrator click on Manage from the toolbar and set up user access in the same way as you would a normal OLAP database.
6. Create a view for the virtual cube (see Figure 63).



		resolution detail
John Muir Company	LERENE	Explained upgrade procedure/discussed upgrade concerns/walked client through upgrading file(s).
	LERETI	Client will rebuild the application.
	LONOFE	Error when accessing Retrieve Lotus- app.ini file must specify app_data_r42 for this release.
	LETEJA	Manually entering External accounts out of sequence in Ranges or Exceptions causes accounts to be lo
	LUNEDA	Resolved
	LUZEGO	Client will restore the application prior to problem.
	LASESO	General Resolution
	LASIFE	Client resolved issue themselves
	LIJATI	Client will rebuild data category.
	LUNUJE	Software Bug submitted with acceptable workaround
	LUSELA	WFO Installation - General Questions or Problems
	LAVAKE	Consultant helped resolving the issue.
	LASODE	this is only occuring on one PC client all other do not exhibit this behavior. The suggestion at th
	LUKIZA	Consolidating - Error reading header for PSF file in DF_PSFRELOAD
	LUKUWI	Run chart logic
	LIJOGA	restore from backup
	LUSUWI	Issue Resolved
	LETERA	Problem resolved by hardware upgrade
	LETUSO	Shipping Request
	LUMACR	Set user limit
	LIKIRA	Shipping Request
	LONETE	Referred to consulting
	LUSEKU	User Error
	LUZIYI	Client wanted to know how to make a copy of their application and test it in 4.3.1.

Figure 63. View on RESOLUTION virtual cube

7. We can also create another model using Analyzer Administrator for the second virtual cube (CUSTOMER) and its associated view to get detailed information on the customer LERENE, LERETI, LONOFE and so forth.

8. Create links from the OLAP cube to the virtual cube (See Figure 64) and between virtual cubes.

9. When we click on the cell on the OLAP cube, automatically Analyzer jump to the linked view that can be itself linked to another view and so forth.

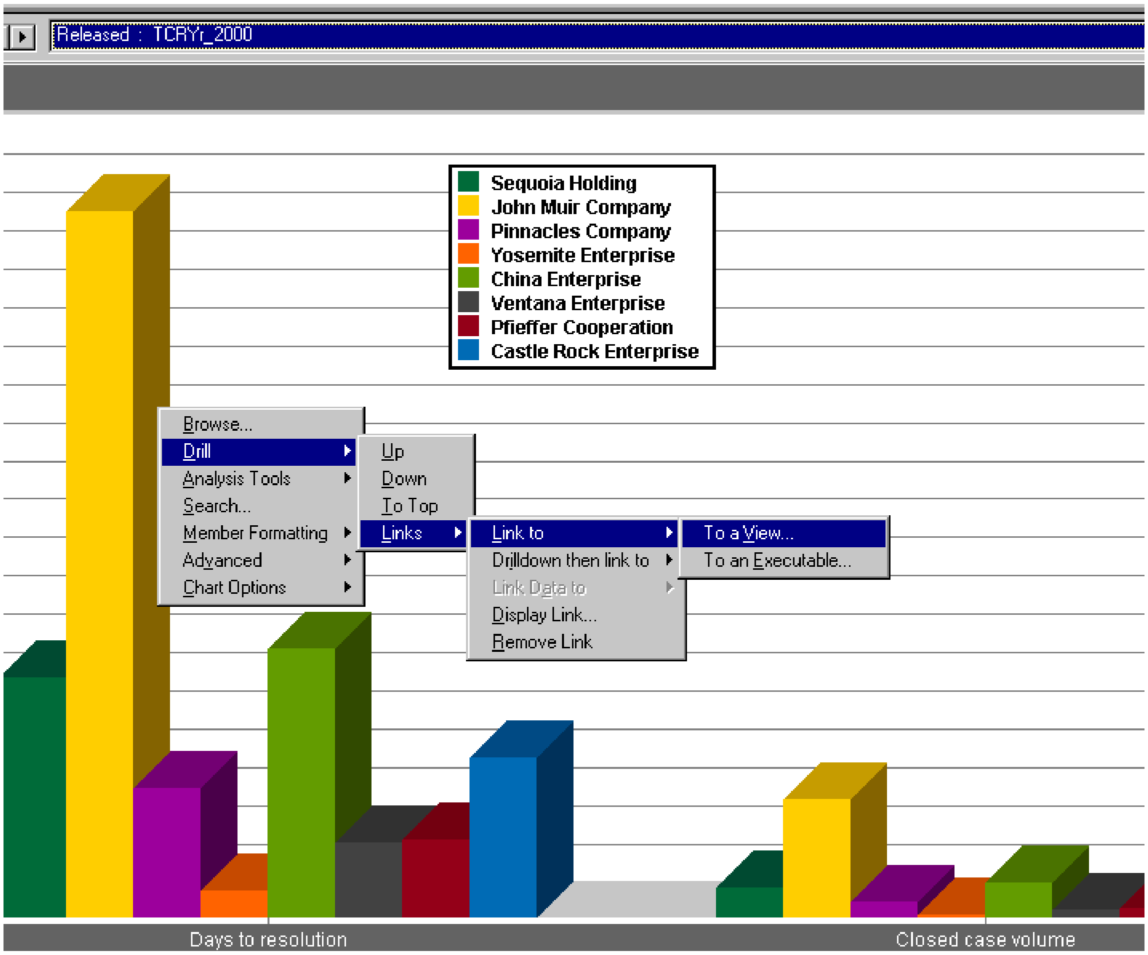


Figure 64. Create links between views

Note: To limit the number of SQL requests and to be more performant, we have to use Automated Summary Table.

C.3 Summary

With this case study, we have attempted to demonstrate a practical approach to build OLAP databases, integrating multiple tools such as DB2 OLAP, Intelligent Miner for Data, OIS, and Analyzer.

Appendix D. Considerations for getting the best OLAP delivery

Reporting requirements have greatly evolved from the static predefined reports of the past. Today, users are asking for:

- More flexibility to be able to interact, navigate, and analyze data for their own business needs, not just for reporting purposes
- Easier access to information, to be able to focus on business issues
- Faster decision-making capabilities
- Wider deployment, catering to an extensive audience of users inside and outside the enterprise

D.1 What properties are needed in an OLAP reporting tool?

When defining a checklist of properties, we need to be thinking about planning for the best in OLAP delivery. The reporting tool should address the following requirements:

1. Advanced reporting capabilities:

   Interactive/analytical More than having a limited set of views available to them, users should be able to manipulate views and data to perform more analysis. They should be able to define formulas based on data retrieved and to format reports based on data criteria.

   Usability An extremely important question to most organizations is: How much work does it take to create a view? Frequently, the users creating the views are not programmers, but are analysts with strong technical aptitude. If they must learn to write Java or some other scripting language, then the product may not be the best fit.

   Write-back ability A tool that can modify the data on the server will have write-back capabilities. Simulations require write-back capability.

   Buy versus build Do I want an “out-of-the-box” solution, easy to implement, mostly point-and-click, or do I want to devote one of my programmers to the tool, able to use script and Java languages for more flexibility.

Multi-source Can the tool bind to several sources to produce a view? What is the level of multi-source capacity needed? Can the tool drill through from the OLAP system to the transactional system? First, the OLAP system must support this, then the front-end must pass this capability through to the user.

2. Deployment capabilities:

Zero-install A zero-install client is one that resides completely on the server.This type of tool is frequently a Web enabled tool that requires no setup on the client computer. It is less development intensive and is therefore a good solution for many organizations. Can the user perform all of the functionality needed through the desktop browser? If so, then this is a good tool for them.

Client types Windows/Java/HTML/Dynamic HTML. These are rendering modes and client types. A rendering mode is a means by which the data is presented. Some tools allow you to create views that can be rendered by several of these means. For example, one tool allows you to create Java, HTML and Dynamic HTML views depending on the level of interactivity needed. Another tool has a Windows client that you use to build all of your views and can be used as the client, but also can publish the views to the Web.

3. Cost issues:

No cost The last important factor to be aware of is the cost, and what can be sacrificed if the tool meets all of the above requirements, but does not fit budget.

D.2 What about DB2 OLAP Server Analyzer?

DB2 OLAP Server Analyzer (Analyzer in the following) is a new add-on feature to DB2 OLAP Server V7.1 based on Hyperion Analyzer V5.0.

Analyzer was built from the ground up with multidimensional concepts in mind, which is very different from traditional reporting tools designed from earlier relational architectures. As a result, concepts such as hierarchies, nesting dimensions on any axis of a report, and so on, are natural and intuitive to the product interface.

Analyzer's main assets in the reporting tool arena are:

- Its interactive and analytical abilities
- The way it can be deployed to a wide range of users
- Its extensibility using open, mainstream development tools like API toolkit
- How it exploits and leverages the power of DB2 OLAP

D.2.1 Interactive and analytical capabilities

Analyzer provides end-users (according to their security profile) a dynamically generated report containing one or multiple groups of views. Each view is a report that is interactive, easy to create, and supports a strong analytical feature set which allows the user to perform data analysis and exploit the strengths of the DB2 OLAP Server database.

Analyzer supports several view types:

- **Spreadsheets:** Tables or grid representations of the data.
- **Charts:** Bar, Line, Cube, Area, and other charts that provide graphic representations of the data.
- **Forms:** A Form is a composite view which allows users to see multiple Analyzer displays and other 3rd party content (Web pages, OLE objects and so forth) on the screen at the same time. Additionally, graphical components like list boxes, radio controls, and buttons (to internal and external applications) allow the user to create easy to use applications, which guide the user and motivate them to explore their data or launch alternative reports. Forms, though sophisticated, require no coding to create them. Forms can be created to provide an intuitive EIS (Executive Information System) interface, which invites users to use the application.
- **Pinboards:** A Pinboard can be any graphical object that helps you to visualize your data. For example, a computer chip manufacturer uses Analyzer to track Inventory. A picture of the chip is used as the Pinboard, and specific areas of the chip appear in red when inventory is running low.

Users can interact directly on the report layout or use the Cube Navigator interface to build a report.

User interactions include all kinds of activities required by analytics: ability to combine and add dimensions; to drill (up, down, to bottom, to next) on charts, spreadsheets, and pinboards; and to pivot data on spreadsheets. In addition, they can add value to data by including calculations (a wide range of calculation functions are available, including basic mathematics, min/max, percentages, ranking, ad-hoc calculation) along with Analyzer's traffic lighting on displays (to compare members between them).

The interactivity and live data retrieval allows users to spot problems or drill to identify the details of a problem.

Analyzer retrieves on-line data from the DB2 OLAP Server in sub-second response times. By drilling to lower levels of data, swapping to display data in a variety of ways, users can spots trends and gain deeper business intelligence. On-line data ensures that all users are using the same “one-truth” of data.

Building views or forms can be done by the end-user. No programming or IT department intervention is required. By empowering the end-user, decision-making can be faster, as transferring the “report requirement” to another department is time-consuming and often an interactive time-delayed process. Users remain in an easy-to use-graphical environment.

Likewise, reports they create can be optionally shared to groups of users, privately for themselves, or shared to the entire enterprise.

D.2.2 Deployment capability

Designed for widespread enterprise deployment, Analyzer adapts to meet the needs of different types of users. It reaches all levels of the enterprise — from the executive who desires an Executive Information System (EIS) and Key Performance Indicator reports, to the power user who wishes to continue his analysis in a spreadsheet.

The out-of-box experience of the packaged Windows, Java, and HTML client makes this possible to easily install Analyzer.

D.2.2.1 Web strategy

Analyzer is enterprise deployable in part due to its Web strategy. In addition to the Windows client, a robust Java client and thin HTML client are also available.

Additionally, the ability to send report data to HTML for static distribution and format customization is also appealing.

When exporting to HTML, several sample templates are included with the product, so that it’s easy to deploy to HTML without a deep underlying knowledge of its syntax. However, if Web masters wish to customize the output, Analyzer templates use an open HTML strategy. Therefore, full customization over the format, look-and-feel, and third party extensions (Java applets, ActiveX controls, JDBC calls to IBM DB2 UDB or other relational data stores) are all at the Web master’s disposal.

A wide range of flexibility exist for the Analyzer content on your Web pages including “cell harvesting”, allowing users to create rich formatted output (sentences, script to automate notification of detected traffic light problems, and so forth).

The Java client

In addition to the Windows client, a Java client is available for enterprise deployment. This client doesn’t require installation of software and uses your Web browser for its interface. As with the Windows client, the common repository for easy and efficient report sharing and personalization is utilized. The Java client is interactive and provides not only report interactivity, but also standard report creation and data editing. Advanced report creation (complex selections, for example) and Form creation is only available from the Windows client.

The Java client supports view reading and standard report creation capabilities. Spreadsheets, Charts, Pinboards, Forms are supported. Additionally, Excel spreadsheet objects and Web pages can be easily launched from the interface. Likewise, any links to applications from a Form is supported if the operating system can locate the program.

As with the Windows client, displays are interactive and easy to use. Views are also grouped and automatically updated to the Home Page.

Since views are stored in a common repository (like DB2 UDB), any new reports you create or modify from the Java client are instantly available to other Web or Window users. Saving a report makes it published to other Window and Web users.

Dynamic HTML client

In addition to the Windows client and Java client, a thin dynamic HTML client is available for enterprise deployment. This client doesn’t require installation of software and uses your Web browser for its interface. As with the Windows and Java clients, the common repository for easy and efficient report sharing and personalization is utilized. The HTML client is interactive and provides not only report interactivity but also data editing capabilities.

Only spreadsheet and charts are supported in this release. The formatting is slightly different because of the need for speed. With the HTML client we are trying remain ultra-thin and reduce network/Web traffic. If the user does have a higher speed modem, they may elect to draw the additional formatting by selecting the Toggle Style button bar.

Displays are interactive and support drilling and display changes, pagination and through the Information Panel additional dimension pivoting and moving.

By saving the view, it becomes available to the Windows and Java clients.

The HTML client also allows for data entry. Because the HTML client is based on HTML templates, administrators can alter the look-and-feel of the application to brand it to corporate standards, remove functionality (edit, for example), as they require.

D.2.2.2 Publication and distribution of reports

When a view has been built, it becomes available to the Windows client, Java client, and HTML client interfaces immediately and without administration, being registered in the centralized Analyzer repository.

The ability to send the current view or information from the view to Excel, 123, clipboard, text file or attach a view reference in email or Lotus Notes allows users to distribute information to other applications.

When using Excel, Analyzer has a feature which allows users to save the Excel workbook into the Analyzer Repository, for centralized secure sharing and distribution.

Often, users want to read static information, or desire a rich formatting display. By exporting an Analyzer view to HTML, both can be achieved. Additionally, users can round-trip (hyperlink) back to the Java client or the HTML client for live data and continued analysis whenever desired. By round-tripping, users can gain advantages of static distribution (that means no time to process when the client asks for the report) and yet still have facilities to jump to a live interactive mode when additional analysis is required.

D.2.2.3 Maintaining reports on the Web

Member selections in a dimension can be static (explicitly selecting East or West), or dynamic (Children of Market dimension). By using dynamic selections, as you add members to your DB2 OLAP Server (example, introduction of new product lines or markets), the cost of maintaining your Analyzer views is low, as no maintenance is required — the reports automatically update themselves! This adds to a powerful enterprise deployable strategy.

D.2.3 API toolkit

In addition to the Web strategy, developers can extend Analyzer using the Analyzer API toolkit, which enables rapid assembly of custom Web-based business analysis applications using open, mainstream development tools.

This allows developers to customize the front-end interface, and to integrate Analyzer views and technology in existing corporate Web strategies and portals.

Some users may only want to customize one or two displays to meet their individual requirements. For example, perhaps they need a specific data entry screen instead of the out-of-box deployed Analyzer Edit Data dialog. Or perhaps an individual chart type requires some specific customization (such as a hyperbolic tree).

Other users may want to seamlessly integrate the Analyzer views, infrastructure, and dialogs into their existing Web pages or portal applications.

Let’s walk through a few customizations that are dependent upon your knowledge of the Web and its accompanying technologies (such as Java Script, plug-ins, and JDBC). The many possibilities are too numerous to count or illustrate.

D.2.3.1 Basic grid component

Analyzer ships a series of components (grid, charts, common dialogs, and function calls) that allow developers to assemble custom Web-based analysis applications.

The developer may include the Analyzer Grid and reference an existing view in the Analyzer Repository.

This page could have easily have been integrated into a Web page containing other portal or Web page content.

Since the developer included the Analyzer Grid component, many grid controls are inherited (Right-Mouse Menu, swapping, drill-down detection, and so on), thus saving the developer many costly development hours. If the developer wanted to customize the grid, a third-party grid control could have been used instead.

D.2.3.2 Chart component

Regarding charts, for example, the developer may have assembled the Analyzer chart component and tied it through several function calls to a series of HTML controls.

This illustrates the ability to add your own menu items to the Analyzer Right-Mouse Menu system.

Developers can plug in their own third party controls for custom chart requirements.

D.2.3.3 Multiple applets

Multiple applets can be embedded at the same time, on the same page, each pointing at a different view, and may combined a third party content.

Analyzer provides an out-of-box Data Entry dialog. For instance, suppose that the user wants a variation of this dialog and wants to include additional business logic or a specific look-and-feel interface. Through the Analyzer API Toolkit, this is possible.

Once a Web application is build (or any other Web page), the contents can be placed on an Analyzer Form and redisplayed within the Windows client interface seamlessly.

D.2.3.4 Relational drill-through via the API

Analyzer may be extended to reference relational data.

Analyzer Windows, Java, and HTML clients also include a robust relational integration strategy. Analyzer Virtual Cubes provide a mechanism to display relational data directly into the Analyzer displays. Analyzer also exploits the DB2 OLAP Integration Server and its relational drill-through capabilities.

A Right-Mouse Menu may be added to invoke Java script, which runs an SQL query to obtain the details of a customer (phone number, address) out of a relational database.

D.2.3.5 Summary of examples

These samples demonstrate how developers can extend Analyzer using the Analyzer API toolkit.

We have seen simple examples of customization and an additional customization to access relational data or to merge data with third-party content in a Portal type application.

By using the Analyzer API Toolkit, components can be assembled or custom created, all the while using the out-of-box infrastructure for view storage, security, firewalls, and enterprise deployment to other developed front-ends or the shipping of out-of-box Windows, Java, and HTML clients.

D.2.4 Leveraging DB2 OLAP

Analyzer was built from the ground up with multidimensional concepts in mind, including the basic parent/child hierarchy relationships as well as the advanced DB2 OLAP Server knowledge of Levels, Generations, Substitution variables, Dynamic Time, and Attributes.

As a result concepts like hierarchies, nesting dimensions on any axis of a report are natural to the product interface, making it easy and intuitive and of high performance.

Using the server provides better performance and reduces network traffic for any organization. For example, when Analyzer needs to build a High and Low Products view, the server is performing the analysis to read through large amounts of data and return only the top and bottom three records per quarter. Analyzer retrieves data from the DB2 OLAP Server in sub-second response times.

Analyzer leverages the power of Hyperion Essbase OLAP Server and DB2 OLAP Server, while also providing drill-through to relational data.

Analyzer supports two out-of-box methods for drilling through to relational detail data:

- Analyzer Virtual Cubes (in the Windows and JAVA clients).
- DB2 OLAP Integration Server Drill-Through Reports. (Additionally, with the Hyperion Analyzer API Toolkit, additional relational technology integration and customization can be achieved. See D.2.3.4, “Relational drill-through via the API” on page 224.)

Additionally, the DB2 OLAP Server security is applied to ensure that users can only change data, which they are secured to edit. From our interface, users can elect to run a particular calculation script to update the entire model.

Appendix E. Web log incorporation

In the traditional business world, merchants can only track information that they can capture. For information on potential customers this generally assumes the form of customer surveys, market research, focus panels, and product testing. To gather information on the customer, merchants have to track and analyze data generated from sales records, customer comments, help desk calls, customer feedback and product returns, and so on.

Today e-business presents us with a different situation. The Internet presents for the first time in the history of commerce an environment where merchants can track and analyze the movements and decisions of existing and potential customers.

This appendix explains how to incorporate Web logs as sources for DB2 OLAP analysis to provide more valuable information.

E.1 Web site analysis suite

Web site analysis should be represented by a suite of business analysis applications designed to help organizations understand, manage and optimize the experience of visitors to their Web sites. By analyzing and understanding visitor behavior, organizations can not only respond more effectively to customer requests, they can become pro-active in soliciting more business from existing customers as well as new business from new customers. Web site analysis is thus an integral part of the organizations customer relationship management (CRM) and data warehouse strategies.

Web logs provide information that can be used to identify a Web site visitor (or customer), the activities of that visitor during a site visit (a session), and as well provide information about how that visitor was referred to the site. Such information is critical when attempting to analyze customer behavior and the effectiveness of a corporate Web site.

E.2 Overview of Web log files

Before information about the Web site visitor can be used for reporting and analysis, the raw data in the log files must be transformed into usable, relevant information. This can be a daunting task because log files can be very large and difficult to read.

- Web log files capture every single visitor interaction and server response. A busy site Web log can grow to an extremely large size very quickly. A process must be defined in such a way as to identify and track visitor activity. Identifying a unique visitor sounds easier than it is. Web log files by themselves do not identify unique visitors. They must be interpreted in order to retrieve this information. Implementing a method to identify unique visitors that summarizes visitor activity into individual sessions is key in Web site analysis.
- Web logs do not maintain visitor sessions. The exchange of information between a unique IP address (visitor actions) and the Web site is entered into the Web log as it occurs within the clickstream. Moreover, a Web server must be able to serve up multiple Web pages simultaneously because of the requirement to respond to potentially thousands of concurrent visitors.Because the log files can only record the clickstream as it happens, each unique visitor's actions are recorded scattered throughout the Web log. A strategy must be devised to associate actions and behaviors with each visitor.

E.3 Web logs as source data

In this section we discuss the use of Web logs as source data for DB2 OLAP.

E.3.1 Web log fields

To repeat, Web logs provide raw data items that have to be interpreted to create an individual visit session. In order to provide effective data for analysis the information that is captured in Web logs needs to be planned and overtly defined by the organization. This functionality is of course provided by the Web Server software. Once this clickstream data has been captured, it can be converted from these raw data elements into information that can be summarized to populate an OLAP model to provide a full suite of reports and analytics.

Most Web server logs can be configured to contain, but are not limited to, data items such as DATE, TIME, CLIENT IP ADDRESS, BYTES RECEIVED, BYTES SENT, COOKIE, HTTP STATUS, METHOD, REFERRER, SERVER NAME, TIME TAKEN, URI QUERY, URI STEM, VISITOR AGENT, and VISITOR NAME. All of this data is potentially valuable in determining the effectiveness of a Web site. Once Web log data is converted into actionable data it can be used to determine, for example, successful user visits (those that result in a sale), or to improve Web site efficiency.

E.3.2 Visitor and session Identification

Since visitor actions contained in a Web log are not in a cohesive format, a process called sessionization is used to derive meaningful information from the log. The sessionization algorithm essentially filters the data to collect accurate information about an individual visitor's session. The goal of the sessionization algorithm is to bring together the unique attributes on which to sessionize and summarize the data.

E.3.3 Sessionization methods

There are a variety of ways to identify the visitor. Some of the methods are more accurate than others, and all methods are dependent on the type of information captured in the Web log.

E.3.3.1 IP only sessionization

IP address only sessionization is the method of counting the number of unique IP addresses. The method is unfortunately not very accurate because many users can enter the same site through a single IP address. For example, most corporate users come through either a single proxy server or a set of proxy servers, which will all be recorded as the same IP address. AOL users, for example, all exhibit a single IP address.

Dynamic IP address assignment (DHCP) can also assign a different IP address to the same workstation over time. Using the IP address only method, this new IP address would be interpreted as a new visitor. Currently, IP address only based sessionization is not considered to be a reliable way of calculating page views or visits, because it does not comply with any audit rules.

E.3.3.2 IP with external referrals

An external referral is a Web site where a user clicked on a link or an advertisement that referred them to another site. The IP Address with external referrals method uses external referrals as the key indicator of unique visits: if an IP address has an external referral then it is recorded as a new visit. This method too is unreliable because it requires that all aliases of a Web site address be recorded. For example, the site *www.ibm.com* has the aliases of ibm.com, IBM.com, IBM.COM, 9.19.138.191, and so on. Moreover, using the IP address with external referrals method, every *page visit* will be interpreted as a *unique visitor*. This makes the method unacceptable for audit rules.

E.3.3.3 IP with user agents and external referrals

User agents identify the operating system and Web Browser employed by the user. In combination with the IP address and external referral, the user agent can be used to better distinguish between visitors coming through the same IP address. By including the agent in the identification process, visitors coming from the same IP address can be identified as unique based on the different agent.

This method is more accurate than the previous two, and the assumption being made is that different visitors will be using different browsers or different versions of browsers, and different computers. For example, if two visitors come from 9.19.138.242, but one has a user agent of MSIE 3.0 and the other has a user agent of MSIE 4.0, then it can be safely interpreted as two separate visitors.

E.3.3.4 IP with user agents and cookies

Visitor cookies are another way to determine the identity of a visitor on a Web site. A cookie is a persistent client-state HTTP identifier associated with a user. The combination of IP address, user agent and cookie can be used to identify a unique visitor with more accuracy than the previous methods. However, there are several issues with cookies that are described below.

E.4 Cookies

Servers store cookies on the client side of the exchange between the Web server and visitor. The servers can then use that information to record and determine user preferences, remember passwords and login IDs, and gather additional statistics about the user. There are two types of cookies that can be issued, *Session* and *Visitor* cookies.

E.4.1 Session cookies

Session cookies are used for visitor identification. They are issued by the site for each visit, and expire after 30 minutes of inactivity. They are arguably the most accurate method for calculating the number of visitors.

Differences in the distribution of session cookies that Web sites employ affect the outcome of the results. For example, a server can assign a session cookie only after the first several pages are viewed. Using this method of cookie distribution, both the pre-cookie page views and the post-cookie page views will be counted as separate session. This will inflate the visit and visitor count. Moreover the fact that visitors are now increasingly disabling (not accepting) cookies from within their browser invalidates the use of the session cookie for visitor identification.

A strategy that increases the complexity of using Session cookies is the use of a fallback method for when cookies are not available. Selecting one or more of the different methods as the fallback method means that the statistics and summarized information about sessions and visitors for the two methods will (most likely) not be comparable. For example, reverting to IP-base sessionization when cookies aren't available produces different Web log information than would be gathered if session cookies were present.

E.4.1.1 Visitor cookies

Visitor cookies are persistent cookies that are distributed to visitors and are not deleted from the client when the session ends. Use of the visitor cookie constitutes another method that can be used to calculate visits.

Many sites assign a visitor cookie only at a certain page. All Web log data previous to that page must either be discarded or included in the analysis separately in order for the statistics to be correct. For example, consider a site that has registration but does not require visitors to register until a certain point has been reached. This type of site has a portion of its pages that are open to the public access. The rest of the site is private. In this scenario, detailed analyses about registered user behavior based on site referrals could not be performed.

Some algorithmic intelligence can be used to help solve this problem but most of these types of sites are not designed so that the information about newly registered users can be tied back to their unregistered information. Additionally, visitor cookies can also be disabled and this will have the same impact on accurate sessionization as the disabling of session cookies has.

E.5 Additional issues

There is more work that must be done once the method of how to identify and bring together unique visitor information has been selected. Decisions regarding which pages should be defined as pay-off pages (to determine whether a visitor's time on the Web site can be considered a successful visit), which pages are viewable or non-viewable, what types of files are downloadable, all need to be made in accordance with the type analysis to be performed. The Web logs must be parsed, filtered and summarized to generate both the metadata and numeric data appropriate to the define and load the OLAP model for analysis.

E.6 Practical example

To perform analysis, such as which page is the most frequently requested, where was the visitor referred from, or to see what the most requested downloadable files were, you may need to construct several different OLAP models. Of course, the number of models that need to be built will depend on the types of analysis to be performed as well as design and scale issues that pertain to DB2 OLAP implementations.

The first step is to identify the data elements necessary for analysis in the Web log. Then parse, filter, summarize and store the data in a format applicable to your analytic requirements.

E.6.1 Sample Web log entry

Consider the following sample log entry example:

```
206.24.101.81 - John 30/Sep/1999:17:53:41 -0400] "GET
/javaclasses/headline.txt HTTP/1.0" 200 1175
“http://www.hyperion.com/index/" "Mozilla/4.6 [en] (Win95; U)" GET
/javaclasses/headline.txt - "HTTP/1.0"
```

Some of the different parts (fields) of the log entry are described in Table 25.

Table 25. Log entry

Log field	Example
Hostname or IP address of client	206.24.101.81. (In this case, the IP address is shown because the Web server’s setting for DNS lookups is not enabled. If DNS lookups were enabled, the client’s hostname would appear.)
RFC 931 information	RFC 931 identity not implemented, authentication server protocol
Username	John (username entered by the client for authentication)
Date/Time of request	30/Sep/1999:17:53:41 –0400 (-0400 is GMT offset)
Full request	GET /javaclasses/headline.txt HTTP/1.0
Status code	200
Bytes transferred	1175
HTTP referrer	Http://www.hyperion.com/index/

Log field	Example
User agent	Mozilla/4.6 [en] (Win95; U)
Method	GET
URI	/javaclasses/headline.txt
Query string of URI	- (in this case, nothing)
Protocol	HTTP/1.0

We can interpret these log fields to reflect the different dimensionality of the Web log data. For example, the HTTP Referrer field would be used to construct the Referrer dimension and the User Agent field would be used to populate our Agent and Platform dimensions and so forth. Moreover, the sessionization process must enable the calculation of useful metrics pertaining to visitor behavior like number of pages visited, duration spent on each page, total length of the session, and so on.

Combining and relating the dimensional information with the numeric statistical we can create a logical and physical star schema representation of Web log data (from differing perspectives.) The star schema can then be used to populate multidimensional OLAP cubes as illustrated in Figure 65.

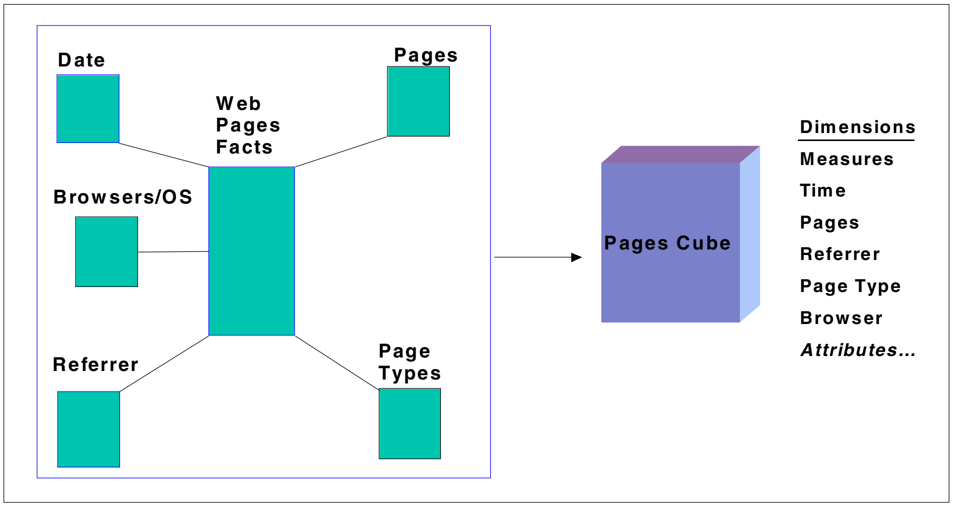


Figure 65. Populating Web cubes

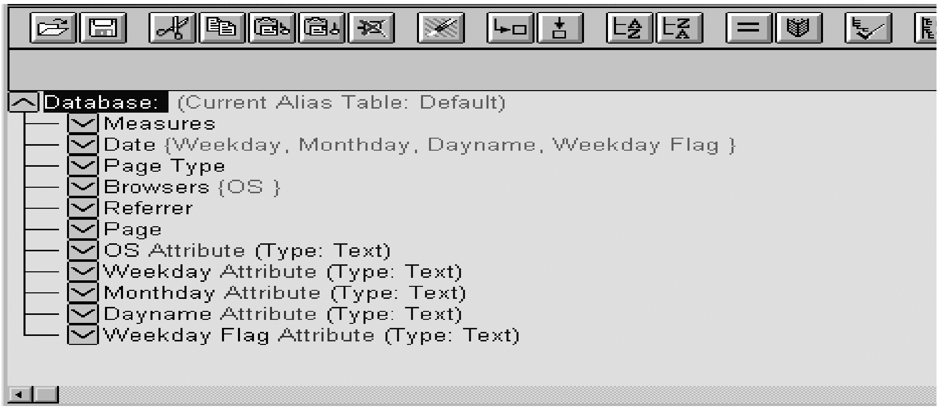
E.6.2 Two sample OLAP models

An OLAP model constructed with the following characteristics would provide answers to questions such as which pages are the most frequently visited, where visitors are being referred from (and to which pages), what the most requested files for download are, and so on (see Table 26).

Table 26. Sample OLAP model

Dimension	Constructed by	Description
Date	Manual build	Daily time dimension
Measures	Manual build	Metrics to capture the number of views
Referrer	Dynamic build from Web logs	Referring page
Pages	Dynamic build from Web logs	Capture all site pages. Need to consider a hierarchical structure for reporting and analysis needs.
Page type	Manual build	Used to tag pages with attributes such as Download, Search Type, Non-Content, Payoff.

The DB2 OLAP outline for this model could look like Figure 66.



```
Database: (Current Alias Table: Default)
  Measures
  Date {Weekday, Monthday, Dayname, Weekday Flag }
  Page Type
  Browsers {OS }
  Referrer
  Page
  OS Attribute (Type: Text)
  Weekday Attribute (Type: Text)
  Monthday Attribute (Type: Text)
  Dayname Attribute (Type: Text)
  Weekday Flag Attribute (Type: Text)
```

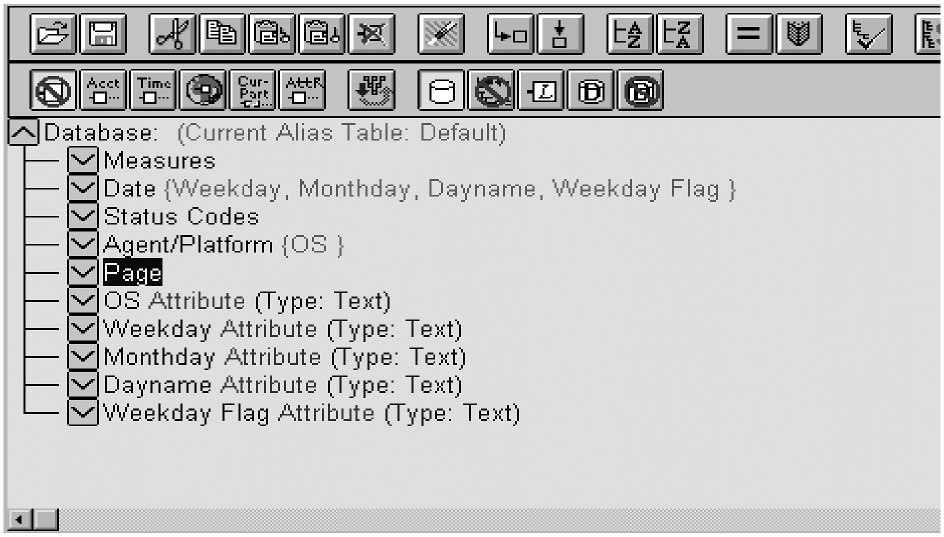
Figure 66. Outline for the sample model

In addition to this first model, a second OLAP model (see Table 27) could easily be generated by leveraging some of the dimensions from the original and adding new dimensionality as required. This would also require appropriately altering the star schema structure. The addition of a few new dimensions prepares a model to analyze information such as Pages Not Found errors or Internal Server errors (see Table 27).

Table 27. Sample additional OLAP model

Dimension	Constructed by	Description
Date	Reuse from pages	Daily time dimension and attributes
Measures	Manual build	Metrics to capture the number of requests
Status code	Manual build	Standard status code
Agent/ Platform	Dynamic build from Web logs	Capture all user agent and platforms used by visitors. Need to consider a hierarchical structure for reporting and analysis needs.
Pages	Reuse from pages	Capture all site pages. Need to consider a hierarchical structure for reporting and analysis needs

The DB2 OLAP Outline for this second model could look like Figure 67.



```
Database:  (Current Alias Table: Default)
  Measures
  Date {Weekday, Monthday, Dayname, Weekday Flag }
  Status Codes
  Agent/Platform {OS }
  Page
  OS Attribute (Type: Text)
  Weekday Attribute (Type: Text)
  Monthday Attribute (Type: Text)
  Dayname Attribute (Type: Text)
  Weekday Flag Attribute (Type: Text)
```

Figure 67. Outline for the additional model

E.7 Conclusion: Integration

Analyzing Web log information as outlined above has obvious benefits, but the focus remains mostly within this electronic domain. The most promising and rewarding return would accrue to the ability to combine Web visitor information with brick and mortar customer information.

To maximize the value of Web data to the corporation, the data we have discussed so far should be considered a data mart which has analytical value for the team running the Web site, and value to the corporation if it is integrated with corporate business intelligence systems and data warehouses.

This data should be considered as data from an additional channel, which stands alongside existing brick and mortar channels. Integration can be accomplished by extracting relevant data from the Web Analytics cubes or data marts and integrating it with similar data from more traditional channels. In this way, we move toward a single view of the customer that encompasses all channels and is richer than the view from any single channel.

Although a corporate data warehouse is the ultimate integration point, it is not necessarily the required starting point for integration efforts. What we must avoid is isolating Web data or attempting to enrich our Web data to the point where we are duplicating corporate data from other sources.

Appendix F. OLAP model development short checklist

This appendix presents in Table 28 an abstract from the OLAP model development checklist described in Chapter 2, “OLAP model development checklist” on page 21.

Table 28. Project shortlist

Activity	Comment?	Check?
1. Plan for OLAP		
a. Recognize an OLAP opportunity.		
b. Evaluate the OLAP opportunity.		
2. Begin high level modeling of the OLAP database structure		
a. Look at current and desired reporting requirements		
• Evaluate the dimensions required		
• Do it for each report		
• From this, begin sketching dimensions		
b. Do modeling development in a group		
• Use Application Manager as a Rapid Application Design tool		
• Project the outline on screen for very fast progress		
c. Evaluate the data using SQL to know data volumes involved and to check the relevance of data		
• If data is relational, use SQL		
• If data is not relational, make it so		
• Check for nulls in data used for building dimensions		
d. Estimate development software hardware required		

Activity	Comment?	Check?
3. Build a prototype for size testing		
a. Build the dimensions		
• Type in the dimension names manually, using directly Application Manager		
• Check if partitioning is required		
• Set preliminary dense/sparse		
• Always use esscmd.exe scripts to build large dimensions		
b. Let the outline do the work		
• Use Member Tags - Use Label only. - Use Dynamic Calc. - Put Formulae in data blocks		
c. Load test data		
• Sort the data by sparse dimensions		
• If SQL sources, make transformations in SQL		
d. Calculating and tuning		
e. Test sizing, calculation performance and query as you go		
• Evaluate the model - Check the model for size - Evaluate the model for calculation time		
• Evaluate the query response time - Check the application log for the spreadsheet retrieval factor - Verification by users and acceptance by users - Adjust sparse/dense dimensions to accommodate user data requests		

Activity	Comment?	Check?
f. Refine dense/sparse to optimize the outline		
• Use the sparse/dense methodology • Use the configuration Wizard		
g. Adjust dimensions and members based on prototype size testing		
4. Build and load the final model		
a. User acceptance testing: the data		
• Use a spreadsheet tool		
b. User acceptance testing: the access user tool		
c. Building and setting up a security model		
• Check the security model with users		
d. Train users		
5. Migrate to production		
• Develop and write the production procedures: - To maintain the outline - To refresh/update the data - To validate the data - To backup and restore the OLAP database - To maintain the software level and set up the patches - To trap errors		

Appendix G. Special notices

This publication is intended to help designers and technical people with an understanding of DB2 OLAP fundamentals to design and implement future DB2 OLAP solutions. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM DB2 OLAP Server. See the PUBLICATIONS section of the IBM Programming Announcement for IBM DB2 OLAP Server for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®
IBM ®
AT
DB2 OLAP Server
Intelligent Miner
DB2 OLAP Integration Server
DB2 Warehouse Manager
Redbooks
Redbooks Logo
AIX
DB2
DataJoiner
QMF

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix H. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 247.

- *Data Modeling Techniques for Data Warehousing*, SG24-2238
- *Business Intelligence Certification Guide*, SG24-5747
- *Business Intelligence Architecture on S/390 Presentation Guide*, SG24-5641
- *Capacity Planning for Business Intelligence Applications: Approaches and Methodologies,* SG24-5689
- *Getting Started with DB2OLAP Server on OS/390* , SG24-5665
- *Intelligent Miner for Data: Enhance your Business Intelligence*, SG24-5522
- *The Role of S/390 in Business Intelligence,* SG24-5625
- *Building VLDB for BI Applications on OS/390: Case Study Experiences*, SG24-5609
- *Managing Multidimensional Data Marts with Visual Warehouse and DB2 OLAP Server*, SG24-5270

H.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **ibm.com**/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037

CD-ROM Title	Collection Kit Number
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

H.3 Other resources

These publications for DB2 OLAP Server are also relevant as further information sources:

- *OLAP Database Administrator's Guide, Volume I* , SC27-0788
- *OLAP Database Administrator's Guide, Volume II* , SC27-0789
- *OLAP Quick Technical Reference* , SC27-0790
- *OLAP SQL Interface Guide* , SC27-0791
- *OLAP Integration Server Model User's Guide* , SC27-0783
- *OLAP Integration Server Metaoutline User's Guide* , SC27-0784
- *OLAP Integration Server Administration Guide* , SC27-0787
- *OLAP Spreadsheet Add-In User's Guide for Excel*, SC27-0786
- *OLAP Spreadsheet Add-In User's Guide for 1-2-3*, SC27-0785

H.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www-4.ibm.com/software/data/` IBM Database and Data Management Home Page
- `http://www.ibm.com/software/download/` IBM free software download Web site
- `http://www.ibm.com/software/` IBM software site
- `http://www.essbase.com/` Hyperion Essbase Web site
- `http://www.olapunderground.com/` Site dedicated to the distribution of freeware Essbase information, applications, and code

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

□ Invoice to customer number

□ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

A

API. Application Programming Interface. For example, the Essbase API is a library of functions that you can use in a custom C or Visual Basic program to access the DB2 OLAP Server.

Application Manager. Administration tool provided in standard with IBM DB2 OLAP Server and Essbase to build OLAP databases.

argument. A name/value pair passed to a web server in association with the URL of a dynamic resource. Different content may be served based on different values within an individual argument. Usually argument strings are separated from the page resource by a question mark and separated from each other with an ampersand (&).A

array. A matrix structure related to a multidimensional database.

B

browser. A software program running on a computer that can request, load and display documents available on the World Wide Web. In our definition, it is assumed that a “human being” is operating the browser manually and viewing the page

C

cache. A component of memory. Each OLAP database contains a data cache, an index and you can configure a calculator cache.

calc or calculation. An equation within a database outline, a calculation script, or a report script that calculates a value for a particular member or point in a report.

calc script or calculation script. A text file that contains instructions to perform calculation within an Essbase database.

cell. A single datapoint that occurs at the intersection defined by selecting one member from each dense dimension in a multidimensional array.

click-thru. This term is typically used for advertising purposes. It refers to the ratio of impressions to clicks on a particular advertisement or link. For example, a toothpaste ad that is shown 100 times but only clicked once has a 1% click-through.

cookie. Persistent client-state HTTP identifier associated with a user.

cube. See OLAP database. In relational, the cube is represented by a fact table.

D

data load. The process of populating an Essbase database with data. Loading data establishes actual values for the values of the cells defined in the database outline for the database.

data load rules. A set of operations registered in a rules file that the DB2 OLAP Server performs on data as it is loaded from an external source file.

datamart. A subject-oriented, integrated, time-variant collection of data to enable decision making for a specific group of users.

data mining. The use of a computer application to search for and discover new insights about the business and significant relationships among the data.

data warehouse. A subject-oriented, integrated, time-variant collection of data to enable decision making across a disparate group of users.

DBA or database administrator. . A person responsible for administering a relational database.

dense. A multidimensional database is dense if a relatively high percentage of the possible combinations of its dimension members contain data values.

dimension. A dimension is a structural data category such as time, accounts, products. In an outline, the dimensions represent the highest consolidation level.

domain. The affiliation of a user's network location at the highest level including geographic location - .edu, .com, .org, .de, .jpe.

Domain Name System (DNS). A protocol and system used on the Internet to map IP addresses to user-friendly names.

drill through. Or drill thru is a specific analytical technique whereby the user can navigate from an OLAP cube among data ranging in a relational database and retrieve detailed data stored in a data warehouse.

E

Entry page. The first page within a visit.

Esscmd. A command-line interface used to perform server operations interactively or through a batch file.

Exit page. The last page within a visit.

F

fact table. A collection of facts consisting of measures and contest data.

FTP File Transfer Program.

H

hit. A hit is a browser request to the web server for any item including graphics, pages, blank lines or other resources. It may take many hits to bring up a single web page as displayed in a web browser. A hit is also any line in a log file. Hits are used to calculate summary hit counts and total bytes transferred.

hostname. The name and affiliation that a particular IP address resolves to, such as Hyperion.com or ibm.com.

HTTP status code. The status code associated with a request indicating whether it was successful, unsuccessful, or otherwise. For example if a resource is missing, the browser will return a 404 error, which is one example of an HTTP status code.

Hyperion sessionizer. An extensible, configurable process that manages the collection, processing and aggregation of log files and other web data for the Hyperion Web Site Analysis application.

I

IP address. A unique number called an IP address identifies every computer on the Internet. Each time you connect to the Internet your machine is assigned an IP address. If you have a static IP address you get the same number each time you connect. If you have a dynamic IP address a number is assigned from a pool of addresses maintained by your ISP (Internet Service Provider) on a first-come basis. You can get a different IP address each time you connect.

IP resolution. The process of transforming IP addresses into user recognizable domain names. This process is also known as reverse DNS. For example 198.105.232.4 becomes Hyperion.com. This method of identifying users can be time consuming and may result in incorrect assumptions about user location.

IT. Information Technology represents the technical staff working on Information System in an enterprise.

M

measure. A numeric attribute of a fact representing the performance or behavior of its dimensions.

member. A discrete component within a dimension. When embedded, members compose the hierarchy in a dimension.

metaoutline. A metaoutline built using Hyperion Integration Server or OLAP Integraton Server contains the basic structure to build a DB2 OLAP outline and to load data into a DB2 OLAP database.

model A model built using Hyperion Integration Server or OLAP Integraton Server contains a star schema and is based on relational datamarts.

O

OLAP. On-Line Analytical Processing is a software technology that enables analysts and executives to gain insight into data through fast consistent, interactive access to a wide variety of possible views of information that reflect the real dimensionality of the enterprise as understood by the user.

OLAP database. A multidimensional database or cube. An OLAP database includes a database outline, data, associated optional calculation scripts, optional report scripts, and data load rules.

OLAP Integration Server. Feature to model and develop multiple outlines from a relational source.

outline. The multidimensional structure that defines all elements of an OLAP database. It contains definitions of dimensions and members, dense or sparse dimension tags and attributes, calculations, shared members, and alternations to the basic roll-up structure of the database.

P

page resolution. A measure in seconds of the amount of time a user spends on a single page.

page hits. A measure of all resources not pre-filtered by the Hyperion Sessionizer. This number includes both viewable and non-viewable page counts.

page view. Page view is a single web page as viewed through an Internet browser. Pages are pre-defined in the Hyperion application as either viewable or non-viewable. Only viewable pages are counted as page views. Pre-defined page views could include all resources ending in HTML, ASP, JSP, to name a few. Page views are also known as page impressions.

payoff. Payoff pages are pages tagged by a site as important because they have revenue implications if a user hits them. For example: Edmunds.com can tag the Jeep Grand Cherokee page as a payout page and attach an attribute of dollars (.60) per page view. Hyperion Web Site Analysis can then produce a report indicating dollars for that section or that page. Similarly, suppose the AHN Web site must pay Dr. Danoff anytime that somebody hits the page containing Dr. Danoff's article. Hyperion Web Site Analysis can tag the payoff page as a payout, and indicate the dollars of cost for this page. Then when we start looking at new versus repeat traffic for this set of articles we could start figuring out whether the articles are a good investment in terms of attracting and/or retaining customers.-

Q

query string. String attached to the end of a page request. For example: recipe.jsp?cuisine=Mexican&timeofday=morning.

R

RDBMS. A Relational DataBase Management System to manage and control relational objects as databases, tables, indexes.

referring page. References the previous page to a particular page that a user is looking at.

referring site. Refers to any site where a user clicked on a link or an advertisement to get to your site. Also known as visitor origin. Example: A user clicks on a link from yahoo.com that sends them to Hyperion.com.

relational database. A database that can be perceived as a set of tables and indexes and manipulated in accordance with the relational model of data, through the SQL language.

relational table. A 2 dimensions structure in a relational database composed of columns and rows and accessed through SQL.

report script. An ASCII file that contains Report Writer commands that generate one or more production reports. Report scripts can be run in batch mode, using the ESSCMD command-line interface, or through the Application Manager. The script is a text file that contains data retrieval, formatting, and output instructions.

RLE or Run Length Encoding. A compression method used by Essbase engine.

S

search. An attribute of a page indicating its status as something that is used in the search process.

search. query Search terms that the user has submitted as a query.

sparse. A multidimensional dataset is sparse if a relatively high percentage of the possible combinations(intersections) of the members from the dataset's dimensions contains missing data.

spreadsheet_add-in. Software that merges seamlessly with Microsoft Excel and Lotus 1-2-3. The software library appears as a menu add-In to the spreadsheet and provides such features as connect, retrieve, zoom-in, and calculate.

star schema. The type of relational database schema composed of a main fact table and a set of dimension tables. The fact table holds the actual data numeric values and the dimension tables hold data about members and their relationships.

status code. Search terms that the user has submitted as a query.The code returned by the web server to the web log that defines whether the request was successful, redirected, or resulted in an error.

Structured Query Language. (SQL) An established set of statements used to manage information stored in a relational database and add, delete, update information in a table.

sub-domain. The affiliation of a user's network location at the provider level such as aol.com, ibm.com, Microsoft.com. Using the process of reverse IP lookup, a user's IP address is resolved into a subdomain.

U

Uniform Resource Identifier (URI). The generic term for all types of names and addresses that refer to objects on the web. A URL is one kind of URI.

Uniform Resource Locator (URL). An address to an object or other destination on the Internet. Example: www.hyperion.com/solutions.

unique visitor. A measure that represents the count of unique identifiers within a time period. Usually measured per day, week and month. A visitor that visits a site twice in one day will only be counted once.The

user agent. The part of a web log, which identifies the Operating System and browser employed by the user.

user name. Registered user name of a user at a particular site.

V

VBA or Visual Basic. Development program.

viewable. An attribute of a page indicating if it is a page view or a page hit. Usually a viewable page is defined as any resource that can contain an ad banner. A frame or other navigation element would not be identified as viewable.

visit. A visit is a series of requests (pages viewed) by a user while at a single web site. A visit ends when a specified period of time (default is 30 minutes) has passed without any additional requests to the site. A visit is usually identified by some unique combination of IP address, user agent and/or cookies. A visit is also known as a session.

visit duration. A measure in seconds of the amount of time a user spends within a single visit.

visit length. A measure in page views of the amount of pages a user views within a single visit.

visit timeout. The length of time of inactivity during which a visit is determined to have ended. A default value of 30 minutes is normally used.

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6138-00 DB2 OLAP Server Theory and Practices
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM’s privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

DB2 OLAP Server Theory and Practices

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

DB2 OLAP Server Theory and Practices

Matrix management and storage structures in depth

Advanced OLAP design practices

Practical implementation experiences

OLAP has evolved into a mainstream information technology and it is a major component of Business Intelligence solutions across all industries today.

The increasing demands for analyzing large amounts of data and empowering a growing number of employees throughout the enterprise to make informed business decisions are pushing the limits of OLAP solutions. Although leading OLAP products, such as DB2 OLAP Server, provide for high levels of platform scalability, it is the design of the models that ultimately determine how well the system is performing and how easy it is to adjust them to fit new business requirements.

This IBM Redbook will provide valuable insight into successful design approaches for building OLAP solutions based on DB2 OLAP Server. It will help architects, data modelers, and implementers of Business Intelligence solutions to understand and avoid design and implementation issues.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6138-00 ISBN 0738419680