ClearLine Group

Pre-Tuning And Sizing Your Hyperion Planning And Essbase Applications Before Building Anything

Agenda

- Introductions
  - Who is this guy and what is ClearLine?
- Major Areas to be covered
  - Pre-Tuning
    - Outline Design
    - Database Settings & Caches
    - Data Load Tuning
    - Calculation Optimization
    - Compression
  - Application Sizing
    - Cube Size
    - Calculation Cache Size
    - Sizing Spreadsheet
  - Q & A

ClearLine Group

ORACLE GOLD PARTNER

INTRODUCTIONS

Steve Light

- Director in ClearLine's Hyperion Practice
- 17 years experience with Hyperion Products
  - Started on Essbase 3 and certified on 4, 5, 6, 7, 9.0, 9.3, & 11
  - Beta tester for ASO on 7.1 and worked with 7x & 9x
  - Hyperion Planning 2, 4, 9.3, and 11 (Classic & EPMA)
  - Analyzer / Web Analysis, Financial Reports
- Worked on 58 different projects at 42 companies
- Industries
  - Aviation, Banking, Consumer Products, Financial, Health Care, Insurance, Manufacturing, Retail, Pharmaceutics, Publishing, Telecom, and Transportation

ClearLine Group

ORACLE GOLD PARTNER

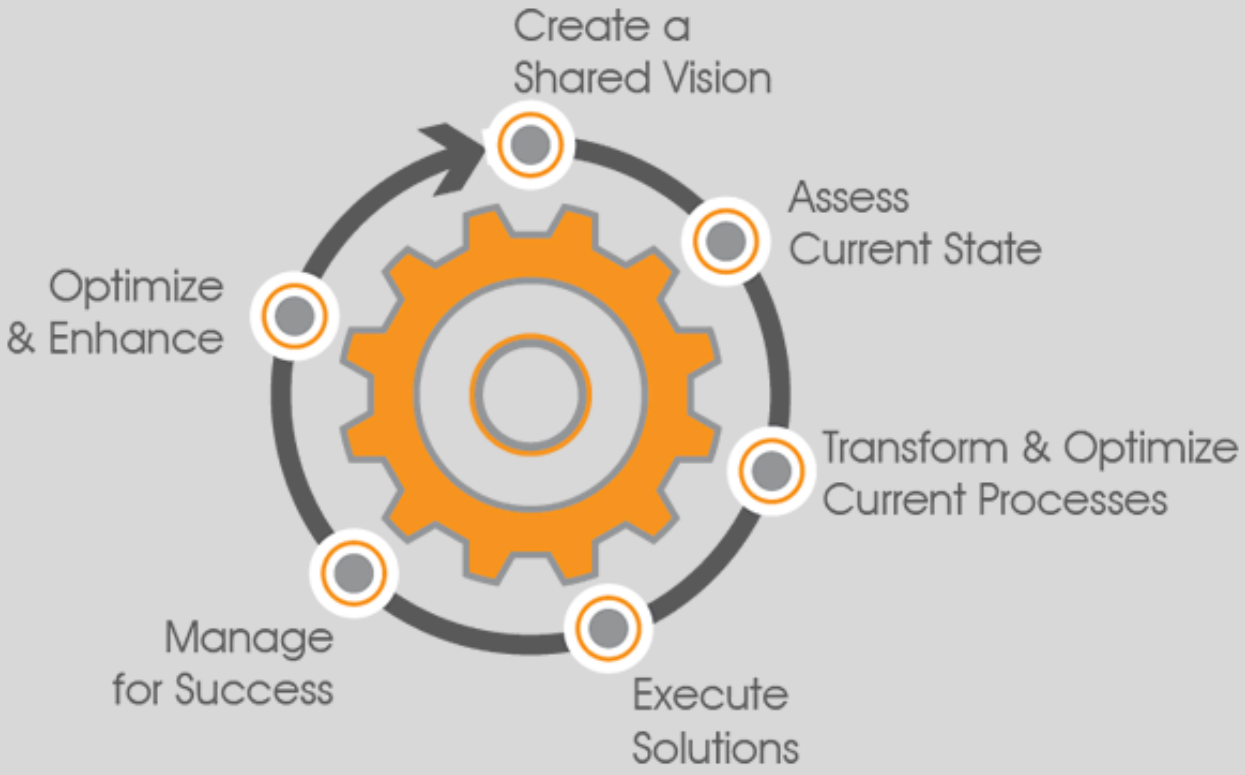
Overview of ClearLine Group

Boutique consulting firm of highly experienced Hyperion EPM experts

- Core team has a deep history of success working together for over 12 years. 'Pioneers' consulting in the Hyperion technologies
  - Backgrounds blend industry and consulting experience
- Breadth of skills from expert level advisory, executive visioning and project management to hands-on architecture and implementation
- All full-time employees, focused on the long-term success of the firm
- ClearLine provides a consistent source of high caliber experts who leverage the collective experience of our group to deliver superior results

ClearLine Group

ORACLE GOLD PARTNER



Provide customers with ‘end-to-end ‘ services

Services range from initial visioning / roadmap process to project oversight and advisory to full implementation and production support services.

Introduction To

PRE-TUNING & SIZING DURING REQUIREMENTS & DESIGN

ClearLine Group

ORACLE GOLD PARTNER

Why pre-tune & size during requirements & design?

- Allows for better up front “go / no go” decisions on the project
- Sets the correct expectations with the end users and project sponsors
- Will aid in project management and CoE standards
- Saves time across the entire project
- Better defines “wants” verse “needs” during requirements by allowing more accurate cost benefit analysis
- Avoids “trial and error” prototyping to see what is possible verses practical
- Helps to stop the development team from painting themselves into a corner during the build
  - It is usually more difficult to reduce scope in later phases of a project
- Stops the common mistakes like:
  - Let’s put it all in and take out what doesn’t work later
  - We’ll worry about tuning after we get the numbers right

What is the bottom line for Planning & Essbase tuning?

- Simplify it where possible
- Reduce the size of the database
  - Avoid reserving space
  - Avoid inter-dimensional irrelevance
- Improve the use of memory
  - Set the appropriate caches
- Improve the Hardware
  - Memory / Processors
  - 64-bit
- Reduce the amount of I/O
  - Reduce passes through the database
  - Restrict the number of data blocks
  - Streamline processes
- Remember that there may be other applications on the server

Introduction To

PRE-TUNING OUTLINES

ClearLine Group

ORACLE GOLD PARTNER

Designing an Outline to Optimize Performance

- Optimizing Calculation Performance
  - Hourglass
  - Modified Hourglass
- Application requirements will drive the decisions
  - Test assumptions in a sizing spreadsheet
  - Lay of the outline correctly up on the first build
    - Changes later can cause extra debugging on calculation due the impact on calculation order

Traditional Hourglass Outline

You may improve calculation performance by changing the order of standard dimensions in the database outline. This will let you have the best calculator cache while using the lest amount of memory. This works well on large cubes with simple aggregations that can leverage the calc cache for the creation of new data blocks. Standard hourglass order is:

Dimension with Accounts Tag
Dimension with Time Tag
Largest Dense Dim.
Smallest Dense
Smallest Sparse
2nd Largest Sparse
Largest Sparse Dimension

Attribute Dimensions

ClearLine Group

ORACLE GOLD PARTNER

Modified Hourglass Outline

You can improve calculation performance by changing the order of standard dimensions to a modified hourglass design. This can help to improve the parallel calculations. It works well on more complex cubes with multiple pass calculations or recalculating cubes that are not leveraging the calc cache. Modified hourglass order is:

Dimension with Accounts Tag
Dimension with Time Tag
Largest Dense Dim.
Smallest Dense
Smallest Sparse Agg
Largest Sparse Dimension Agg

Non-Agg Sparse Dimensions

Attribute Dimensions

ClearLine Group

ORACLE GOLD PARTNER

Modified Hourglass Outline 2

Consider placing Time at the top of the outline. The Time and Accounts tags may force the calculation order, but the outline order will change the cell arrangement in the data block. This can aid RLE compression across the Time dimension in planning applications where values repeat across time.

Dimension with Time Tag
Dimension with Acct Tag
Largest Dense Dim.
Smallest Dense
Smallest Sparse Agg
Largest Sparse Dimension Agg

Non-Agg Sparse Dimensions

Attribute Dimensions

ClearLine Group

ORACLE GOLD PARTNER

Incremental Restructuring

- Incremental Restructuring
  - This will help shorten the restructure time by deferring the restructure until either the index or the affected blocks are access.
  - Incremental restructuring is not available with Linked Reporting Objects (LROs)
  - This will increase the retrieval time
  - This feature is normally not used.

Other Outline Tuning Items

- Disable auto configuration for the dense sparse settings
  - Your common sense will always be better
- Avoid unnecessary dimensions
  - Use attribute dimensions where ever there is a 1 to 1 relationship between sparse dimensions
  - Try to use aliases or concatenated member names to combine to dimensions (e.g. employee name, title, and employee number should not be 3 different dimensions)
  - Avoid inter-dimensional irrelevance
- Avoid enabling duplicate member names
  - It adds overhead
  - Adds complexity to reports
- Avoid flat sparse dimensions
  - You can improve performance for outlines with multiple flat dimensions by adding intermediate levels to the database outline.

Other Outline Tuning Items

- Data Blocks
  - 8K to 100K on 32 bit machines and 8K to 300K or larger on 64 bit machines
  - Large sparsely populated data blocks will not use the Data Cache effectively.
  - Large blocks can hurt parallel calculations
  - Large blocks can help calculations when all of the cross dimensional pointer dimensions was in a single data block (i.e. set to dense)
  - Data blocks smaller than 8K will cause the index to be too large and force Essbase to write to and retrieve the index from disk. This process slows calculations.
    - The index cache search time can be greater than the I/O time for extremely large indexes
  - If data blocks are larger than 100K, then Intelligent Calc does not work effectively.
  - 40K data blocks are a good size to try to for
- Avoid unnecessary stored members
  - Use Label Only members, Shared members, Dynamic Calcs (dense dimensions), and Dynamic Time Series

Introduction To

PRE-TUNING DATA LOADS

ClearLine Group

ORACLE GOLD PARTNER

Data Load Optimization

- There are two areas that may need to be reviewed for optimization during a data load.
  - Minimizing the time Essbase uses to read and parse the data source.
  - Minimizing the time Essbase uses to do I/O
- Most data load tuning is in the file layout
  - Avoid re-work by making sure that your initial file specifications include tuning guidelines

Data Load Optimization

- Sort sparse member combinations
  - This is one of the most important things you can do
- Position data in the same order as the outline
  - If the index cache is large enough to hold the index in memory, then this method does not speed up data loading.
- Reduce the number of rows being loaded
  - Use multiple data columns
  - Use a dense dimension for data column headers
  - Avoid using the *DATA* column
- Replace zeros with #MI in the data load rule
  - Improves block compression and creates few data blocks
- Load data file from the server instead of through the network
- Avoid unnecessary columns in the source data

ClearLine Group

ORACLE GOLD PARTNER

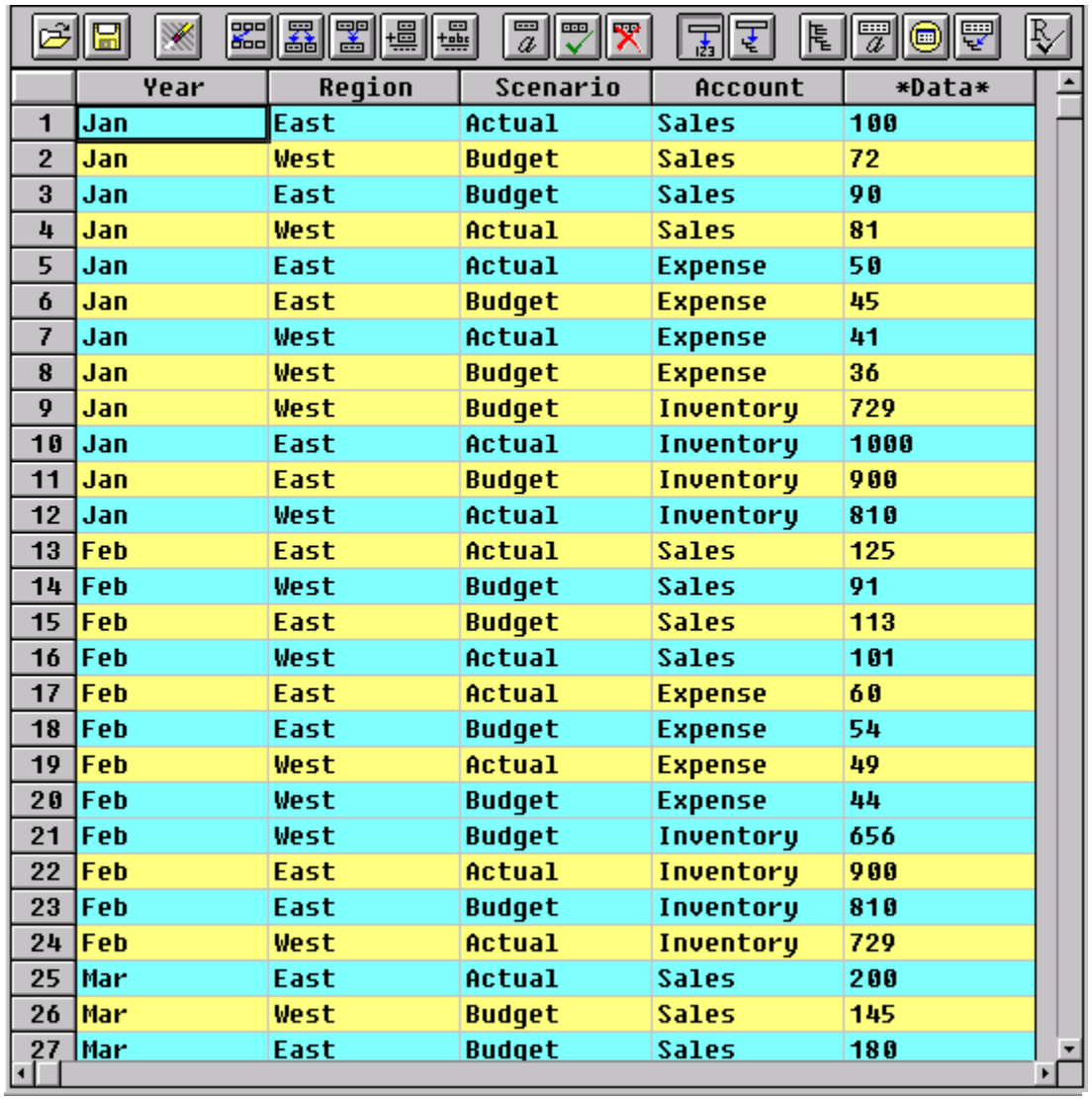
Data Load Example

- Assume that you would like to load a file contains 36 rows. The outline has the follow dense / sparse setting:
  - Account - Dense
  - Year - Dense
  - Region - Sparse
  - Scenario - Sparse

ClearLine Group

ORACLE GOLD PARTNER

Example Of A Poorly Tuned Data Load Rule



	Year	Region	Scenario	Account	*Data*
1	Jan	East	Actual	Sales	100
2	Jan	West	Budget	Sales	72
3	Jan	East	Budget	Sales	90
4	Jan	West	Actual	Sales	81
5	Jan	East	Actual	Expense	50
6	Jan	East	Budget	Expense	45
7	Jan	West	Actual	Expense	41
8	Jan	West	Budget	Expense	36
9	Jan	West	Budget	Inventory	729
10	Jan	East	Actual	Inventory	1000
11	Jan	East	Budget	Inventory	900
12	Jan	West	Actual	Inventory	810
13	Feb	East	Actual	Sales	125
14	Feb	West	Budget	Sales	91
15	Feb	East	Budget	Sales	113
16	Feb	West	Actual	Sales	101
17	Feb	East	Actual	Expense	60
18	Feb	East	Budget	Expense	54
19	Feb	West	Actual	Expense	49
20	Feb	West	Budget	Expense	44
21	Feb	West	Budget	Inventory	656
22	Feb	East	Actual	Inventory	900
23	Feb	East	Budget	Inventory	810
24	Feb	West	Actual	Inventory	729
25	Mar	East	Actual	Sales	200
26	Mar	West	Budget	Sales	145
27	Mar	East	Budget	Sales	180

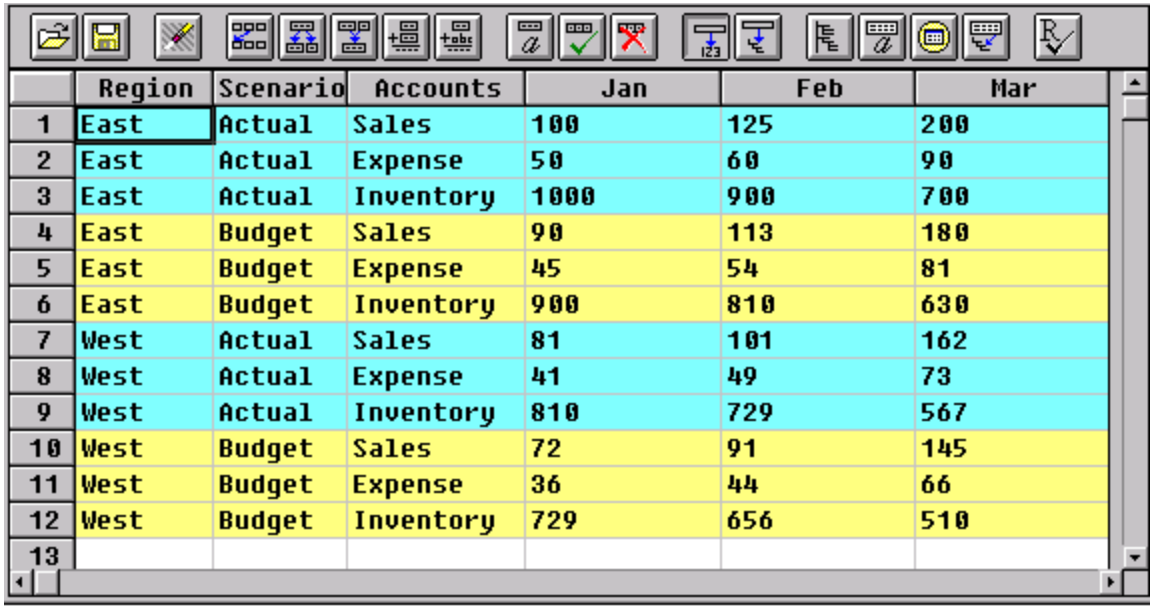
In the following example the system reads and writes to data blocks 33 times, but it only hits 4 blocks.

This will create excess I/O and it will fragment the database during the data load.

ClearLine Group

ORACLE GOLD PARTNER

Example Of The Data Load Rule After Tuning



	Region	Scenario	Accounts	Jan	Feb	Mar
1	East	Actual	Sales	100	125	200
2	East	Actual	Expense	50	60	90
3	East	Actual	Inventory	1000	900	700
4	East	Budget	Sales	90	113	180
5	East	Budget	Expense	45	54	81
6	East	Budget	Inventory	900	810	630
7	West	Actual	Sales	81	101	162
8	West	Actual	Expense	41	49	73
9	West	Actual	Inventory	810	729	567
10	West	Budget	Sales	72	91	145
11	West	Budget	Expense	36	44	66
12	West	Budget	Inventory	729	656	510
13						

The modified load only contains 12 rows with 3 data columns. The system only reads and writes each data block once after optimization

This will not create excess I/O or fragment the database during the data load.

ClearLine Group

ORACLE GOLD PARTNER

Introduction To

PRE-TUNING CALCULATIONS

ClearLine Group

ORACLE GOLD PARTNER

Calculation Optimization

- Default “Calc All” can be the most efficient way to calculate the database
  - Rarely used anymore – Consider ASO for simple “rack & stack” applications
- Place member formulas in dense dimensions
  - Make them dynamic where ever possible
- Largest sparse dimension should be the last dimension when leveraging the Calc Cache
- Unary calculations instead of formulas

Calculation Optimization

- Turn “AGGMISSG” on when possible
  - Do not use with loaded upper level data
- Simplify formulas with Hyperion Essbase functions
- Dynamic Calcs Non-Store and Dynamic Times Series where ever possible.
  - Dense dimension non-zero level members
  - Dense dimension zero level member with member formulas
  - Simple sparse calculation (E.G. Variance calculations that only hit 2 data blocks)
- Avoid using Dynamic Calc And Store

ClearLine Group

ORACLE GOLD PARTNER

“Two Pass” Optimization

- Calculate “Two Pass Calcs”
  - Always try to tag Two Pass Calculations as Dynamic Calcs.
  - Change the outline order to stop the requirement for a stored Two Pass Calculations

Simplify Calc Scripts & Business Rules

- Simplify the calculations where ever possible
- Reduce the number of passes
  - Group dense members together
  - Calculate dense members before sparse members
  - Remember “If Dense / Fix Sparse”
    - Reduce the number of passes through a database
    - Only calculate a sub-set of the database

ClearLine Group

ORACLE GOLD PARTNER

Calc Parallel

- Breaks each calculation pass into sub-tasks that can run simultaneously on up to 8 processors
- Works better with a Modified Hour Glass Design
- Large blocks can hurt parallel calculations
- It is not supported with incremental restructuring, committed access isolation level, numerous calculation functions (e.g. @VAR, @ARRAY, or @XREF) and will switch a back to serial processing
- Do not set “CALCTASKDIMS” too high
  - Determines the task schedule for the number of sparse dimensions
  - Default is 1 and will only use the last spares dimension in the task list
  - Calc performance can be reduced by asking Essbase to analyze too many dimensions

Intelligent Calculation

- "Intelligent Calculation" allows Essbase to remember which blocks need to be calculated.
  - Identifies which block need to be calculated after data is incrementally loaded.
    - Clean Block / Dirty Block
    - Dependent parents
  - Works well of the simple aggregations and "calc alls" after an incremental data loads and lock & sends.
  - Can cause incorrect results for complex calculations, calc scripts / business rules, and multiple pass calculations
- Use "Intelligent Calculation" only where appropriate

Introduction To

PRE-TUNING DATABASE SETTINGS & CACHES

ClearLine Group

ORACLE GOLD PARTNER

Buffered I/O Verses Direct I/O

- Buffered I/O
  - Uses the file system buffer cache.
  - Essbase uses Buffered I/O by default
  - Most databases use Buffered I/O
  - Does ***NOT*** use the Data File Cache
- Direct I/O
  - Bypasses the file system buffer cache and is able to perform asynchronous I/Os
  - Uses both the Data File Cache and Data Cache
    - Tuning should be focused on the Data File Cache
  - The OS should ***NOT*** be allowed to allocate memory to the file system cache.
  - This can "starve" non-Essbase applications and Buffered I/O applications.
    - It should only be used on dedicated Essbase servers where all of the applications are set to Direct I/O.
- Consider using Buffered I/O unless you have a compelling reason to change it
- All of the databases on a server should use the same I/O setting

Essbase Caches

Cache	Description
Index Cache	A buffer in memory that holds index pages. How many index pages are in memory at one time depends upon the amount of memory allocated to the cache.
Data File Cache	A buffer in memory that holds compressed data files (.pag files). Essbase allocates memory to the data file cache during data load, calculation, and retrieval operations, as needed. The data file cache is used only when direct I/O is in effect.
Data Cache	A buffer in memory that holds uncompressed data blocks. Essbase allocates memory to the data cache during data load, calculation, and retrieval operations, as needed.
Calculator Cache	A buffer in memory that Essbase uses to create and track data blocks during calculation operations.
Dynamic Calculator Cache	A buffer in memory that Essbase uses to store all of the blocks needed for a calculation of a Dynamic Calc member in a dense dimension (for example, for a query).

Essbase Caches

- Essbase provides default size settings for each cache; you ***WILL*** need to adjust these settings
- Appropriate cache size is affected by many factors, including database size, block size, index size, available server memory, and calculation functions being used (e.g. @allocate).
- Cache size settings can significantly affect database and server performance.
- When a data block is requested:
  - Essbase searches the data cache for the block.
  - If Essbase finds the block in the cache, it is accessed immediately.
  - If the block is not found in the cache, Essbase searches the index for the appropriate block number and then uses the index entry of the block to retrieve it from the appropriate data file on disk.
  - Retrieving a requested block from the data cache is faster and therefore improves performance.

Index Cache

- The most recently accessed index pages are held in the index cache
  - How much of the index can be held in memory at one time depends on the amount of memory that you allocate to the index cache
  - Note: The size of index pages is fixed at 8K.
- Index Cache Settings:
  - Minimum Value: 1,024K
  - Default Value Buffered I/O: 1,024K
  - Default Value Direct I/O: 10,240K
- Recommended Value
  - In theory, make it the same size as the index; otherwise, as large as possible.
    - Do not set this cache size higher than the total index size, because no performance improvement results.
  - The practical solution is to start with 50M to 100M
    - Try to achieve an Index Hit Ratio greater than 90% after you get data into the cube

Data File Cache

- The Data File Cache only works with Direct I/O
- Data blocks remain compressed in the Data File Cache
- In general, if you must choose whether to allocate memory to the data cache or to the data file cache, choose the data file cache.
- Data File Cache Settings:
  - Minimum Value: 10,240K
  - Default Value: 32,768K
- Recommended Value
  - In theory, make it the same size as the page file; otherwise, as large as possible.
    - Use the sizing spreadsheet for estimates during requirements and design
  - The practical solution is to start with 300M to 500M and adjust after full data loads are loaded and aggregated

Data Cache (Direct I/O)

- Data Cache Settings:
  - Minimum Value: 3,272K
  - Default Value: 3,272K
- Pulls data blocks from the Data File Cache
- Recommended Value
  - The recommended size is 12.5% of the value of data file cache size.
    - Approximately 40M to 65M
  - Increase value if any of these conditions exist:
    - Many concurrent users are accessing different data blocks.
    - Calc scripts contain functions on sparse ranges or functions requiring all the blocks be in memory
  - Data cache and data file cache settings may be larger than 4 GB 64-bit platforms.
    - You cannot specify settings larger than 4 GB in Essbase clients, you can enable larger settings using the MEMSCALINGFACTOR configuration setting.

Data Cache (Buffered I/O)

- Data Cache Settings:
  - Minimum Value: 3,272K
  - Default Value: 3,272K
- Pulls data blocks from the Page File
- Recommended Value
  - In theory, make it the same size as the page file; otherwise, as large as possible.
  - The recommended size is 12.5% of the page file size
    - Use the sizing spreadsheet for estimates during requirements and design
  - The practical solution is to start with 300M
    - Adjust after full data loads are loaded and aggregated
    - Try to achieve an Hit Ratio greater than 70% with a minimum goal of 30%

Calculator Cache

- The “Calc Cache” can improve calculation times by creating a place in memory to create and track new data blocks during calculation.
  - The Calc Cache is not a true cache.
  - It is not a “stash” of index pages or data blocks in memory
  - It tracks children and parents in support of block creation
  - The Calc Cache is the only Essbase cache that releases memory when a process is complete
- It is particularly useful when you calculate your database for the first time.
  - Subsequent calculations are often slower than the 1st time the calc is run
  - Subsequent calculations are more likely to use the data cache for blocks that are being recalculated instead of created
- Use the sizing spreadsheet to calculate the required Calc Cache Size

Dynamic Calculator Cache

- The Dynamic Calculator Cache is used to support queries
- There is a separate Dynamic Calculator Cache for each database
- The Essbase configuration file setting DYNCALCCACHEMAXSIZE is used to specify the size
- Default maximum size is 20M
  - May want to increase it to 40M during the initial build
- Review if there is a performance issue after full data loads are loaded and aggregated

DYNCALCCACHEMAXSIZE = CALCLOCKBLOCK * BLOCK SIZE * CONCURRENT USERS

ClearLine Group

ORACLE GOLD PARTNER

Introduction To

PRE-TUNING COMPRESSION

ClearLine Group

ORACLE GOLD PARTNER

Types Of Compression

- Bitmap
- RLE (Run-Length Encoding)
- Zlib
- Index Value Pair (IVP)

NOTE: The 72B Block Header can never be compressed

Compression Optimization

- Bitmap
  - Compresses #Missing
  - Good for non-repeating data
  - May allow for faster data loads and calculations
  - Use it unless you have a specific reason not to.
- RLE (Run-Length Encoding)
  - Compresses like values
  - Use on extremely sparse databases to compress repeating #Missings
  - Use on databases with loaded zeros
  - Use on planning cubes where numbers repeat
  - Consider updating the Modified Hourglass design with Time the first dimension in the outline

Compression Optimization

- Zlib
  - Essbase builds a data dictionary based on the actual data being compressed
  - Good for extremely sparse databases
  - Calculation and data loading are faster with direct I/O and zlib compression than with buffered I/O and zlib compression
  - Usually will shrink the database the most uses extra CPU Processing
- Index Value Pair (IVP)
  - Essbase applies this compression if the block density is less than 3%
  - Good for very large data blocks with sparse data
  - This compression algorithm is not selectable
  - It is automatically used whenever appropriate by the database.

Compression Optimization

- No Compression
  - Avoids excessive processing overhead that causes any type of compression to be less desirable than no compression at all
  - Only on a heavily populated database that does not have repeating values and has an average block density well over 50%.

Pre-Tuning

THINGS TO REMEMBER…

ClearLine Group

ORACLE GOLD PARTNER

Pre-Tuning Take Aways

- Outline
  - Use an Hourglass or Modified Hourglass design with an appropriate data block size
- Data Load
  - Sort your data
- Calculation
  - Simplify formulas, use dynamic calc non-store, remove passes calc scripts / business rules, and restrict the area being calculated
- Database
  - Apply appropriate cache settings
- Compression
  - Use Bitmap unless RLE fits a specific need

ClearLine Group

ORACLE GOLD PARTNER

Introduction To

SIZING HYPERION PLANNING & ESSBASE BSO

ClearLine Group

ORACLE GOLD PARTNER

Data Block Size

- Data block size is based on the stored intersections in the dense dimensions.
- The stored intersections represent the data cells within the data block.
- The data block size changes when:
  - Change the dense or sparse setting
  - Add or remove members to a dense dimension
  - Add or remove label only, dynamic calcs, or implied shares
- Data block size is 8n bytes, where n is the number of cells that exist for that combination of dense dimensions.
- There is a non-compressible 72B block header on each data block
- The optimum size range
  - 8K–100K for 32-bit
  - 8K–300K or larger for 64-bit servers
  - Many applications on 64-bit servers may have much larger blocks for some types of designs.

Calculating The Data Block Size

- Calculate the data block size with the following information:

Dimension Names	Outline Members		Dense / Sparse Setting
	Stored Members	All Members	
Measures (Accounts)	9	13	Dense
Period (Time)	66	396	Dense
Scenario	18	47	Dense
Fiscal Year	6	11	Sparse
Versions	5	6	Sparse
Channel	31	37	Sparse
Product	3,000	3,000	Sparse

  - Multiple the stored dense members together time 8 plus 72
  - ((9 * 66 * 18) * 8) + 72 = 85,608B
  - 85,608B / 1024 = 83.6K

Please note that the addition of the block header was included in this calculation on Essbase versions 3 through 7. The block header was dropped from the official calculation in System 9 to simplify the calculation, but the block header is still there and should be included to be more accurate.

Potential Number Of Blocks

- The potential number of data blocks is the maximum number of data blocks possible in the database.
  - If the database is already loaded, you can see the potential number of blocks on the Statistics tab of the Database Properties dialog box of Administration Services.
  - If the database is not already loaded, you must calculate the value.
- To determine the potential number of data blocks, assume that data values exist for all combinations of stored members

Calculating Potential Number Of Data Blocks

- Calculate the potential number of data blocks with the following information:

Dimension Names	Outline Members		Dense / Sparse Setting
	Stored Members	All Members	
Measures (Accounts)	9	13	Dense
Period (Time)	66	396	Dense
Scenario	18	47	Dense
Fiscal Year	6	11	Sparse
Versions	5	6	Sparse
Channel	31	37	Sparse
Product	3,000	3,000	Sparse

- Multiple the stored sparse members together
- (6 * 5 * 31 * 3,000) = 2,790,000 Potential Data Blocks

Calculating Max Database Size

- Multiple the data block size by the potential number of data blocks
  - Data Block Size ((9 * 66 * 18) * 8) + 72 = 85,608B
  - Potential Data Blocks (6 * 5 * 31 * 3,000) = 2,790,000
  - **(85,608 * 2,790,000) = 238,846,320,000B or 222.4G**
- This value assumes that every data block is created and that every data cell is populated.
- A more useful number will adjusts this value based on the sparseness of the data
  - Adjust for block compression on dense dimensions
  - Adjust for block creation on sparse dimensions

Estimating Sparseness

- Estimating Sparseness With Sample Data
  - Try to load a representative data set
  - Example:
    - 1 full month of data for 1 year, 1 scenario, and 1 versions with all dense/sparse combinations represented
    - Assuming that time is dense, estimate what the compressed block would be with 12 months of data.
    - Multiple this by the years, scenarios, and versions to come up with an estimated size
  - Unfortunately, this is usually not possible during requirements or design

Estimating Sparseness

- Estimating Sparseness Without Sample Data
  - Design an hourglass outline or modified hourglass with a reasonable block size
  - Calculation the block size and the maximum size of the database base on the outline
  - Apply estimates on compression and sparseness for a realistic size
    - This is the best way to design a cube without building it
    - It is also very useful for testing potential enhancements, controlling scope, and for documentation

Estimating The Calc Cache Size

DIMENSION NAME	TYPE	DECLARED SIZE	ACTUAL SIZE	DEPENDENT PARENTS
YEAR	DENSE	53	52	
ACCOUNTS	DENSE	150	148	
MARKET	SPARSE	30	20	
CHANNEL	SPARSE	25	20	TWO ALT ROLL-UPS
PRODUCT	SPARSE	50	50	
VERSION	SPARSE	55	50	
CUSTOMER	SPARSE	222	200	TWO ALT ROLL-UPS

A **single anchor multiple bitmap** is most effective Calc Cache. It can be calculated by first identifying the bitmap dimensions and the anchor dimensions. The anchor dimension is the last sparse dimension in the outline and it should also be the largest sparse dimension. The bitmap dimensions are the remaining sparse dimensions.

- Multiply the actual members of the bitmap dimensions together and divide by eight.

- (20*20*50*50)/8 = 125,000

- Remember, there are always two constant bitmaps for this option. Add two constant bitmaps to the total number dependent parents in the anchor and multiply this sum by the answer above for the total number of bytes needed.

- 2 constant bitmaps + 3 dependent parents = 5

- 5 * 125,000 = 625,000 bytes

ClearLine Group

ORACLE GOLD PARTNER

Estimating The Calc Cache Size

DIMENSION NAME	TYPE	DECLARED SIZE	ACTUAL SIZE	DEPENDENT PARENTS
YEAR	DENSE	53	52	
ACCOUNTS	DENSE	150	148	
MARKET	SPARSE	30	20	
CHANNEL	SPARSE	25	20	TWO ALT ROLL-UPS
PRODUCT	SPARSE	50	50	
VERSION	SPARSE	55	50	
CUSTOMER	SPARSE	222	200	TWO ALT ROLL-UPS

A **single anchor single bitmap** is second best, but uses less memory.
Multiply the bitmap dimensions together and divide by eight: (20*20*50*50)/8 = 125,000 bytes

A **multiple anchor single bitmap** is least effective, but uses lowest amount of memory. The second anchor dimension is the second to last sparse dimension in the outline.
Multiply the bitmap dimensions together and divide by eight: (20*20*50)/8 = 2,500 bytes

NOTE: The default calculator cache size is 200,000 bytes and the maximum calculator cache size that you can specify is 200,000,000 bytes.

NOTE: The minimum bitmap size is 4 bytes. If (member combinations on the bitmap dimensions/8) is less than 4 bytes, Essbase uses a bitmap size of 4 bytes.

You Don't Have To Do The Math!!!!

You can use the ClearLine Sizing and Calc Cache spreadsheet to do it for you!!!

ClearLine Group

Dimension Number	Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
		Stored Members	All Members		
Dimension Position 1	Measures (Accounts)	9	13	Dense	
Dimension Position 2	Period (Time)	66	396	Dense	
Dimension Position 3	Scenario	18	47	Dense	
Dimension Position 4	Fiscal Year	6	11	Sparse	BITMAP
Dimension Position 5	Versions	5	6	Sparse	BITMAP
Dimension Position 6	Channel	31	37	Sparse	BITMAP
Dimension Position 7	Product	3,000	3,000	Sparse	ANCHOR

Estimated Block Density	
Block Size After Compression	30%

Number of Dependent Parents in the Anchor Dimension	
Product	1

Estimate The Cube Size Instantly

Dimension Number	Dense Dimensions	Dense Stored Members	Sparse Dimensions	Sparse Stored Members
Dimension Position 1	Measures (Accounts)	9		
Dimension Position 2	Period (Time)	66		
Dimension Position 3	Scenario	18		
Dimension Position 4			Fiscal Year	6
Dimension Position 5			Versions	5
Dimension Position 6			Channel	31
Dimension Position 7			Product	3,000
	Product of Dense	10,692	Product of Sparse	2,790,000
Potential / Maximum Sizes				
Potential Block Size (in bytes)	85,608 B	84 K	(8 - 100K Optimum on 32 bit)	
Potential Number of Blocks	2,790,000 Blocks			
Potential Database Size (in bytes)	238,846,320,000 B	233,248,359 K	227,782 M	222.4 G
Data Sparseness	Database Size			
Potential Database Size (at 50%)	35,826,948,000 B	34,987,254 K	34,167 M	33.4 G
Potential Database Size (at 25%)	17,913,474,000 B	17,493,627 K	17,084 M	16.7 G
Potential Database Size (at 10%)	7,165,389,600 B	6,997,451 K	6,833 M	6.7 G
Potential Database Size (at 5%)	3,582,694,800 B	3,498,725 K	3,417 M	3.3 G
Potential Database Size (at 2%)	1,433,077,920 B	1,399,490 K	1,367 M	1.3 G
Potential Database Size (at .5%)	358,269,480 B	349,873 K	342 M	0.3 G
Potential Database Size (at .05%)	35,826,948 B	34,987 K	34 M	0.0 G
Potential Database Size (at .01%)	7,165,390 B	6,997 K	7 M	0.0 G
Potential Database Size (at .005%)	3,582,695 B	3,499 K	3 M	0.0 G
Potential Database Size (at .001%)	716,539 B	700 K	1 M	0.0 G

Estimate The Calculation Caches Instantly

Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
	Stored Members	All Members		
Measures (Accounts)	9	13	Dense	
Period (Time)	66	396	Dense	
Scenario	18	47	Dense	
Fiscal Year	6	11	Sparse	BITMAP
Versions	5	6	Sparse	BITMAP
Channel	31	37	Sparse	BITMAP
Product	3,000	3,000	Sparse	ANCHOR
Single anchor dimension multiple bitmap (best)				
Bitmap Dimensions				
Fiscal Year	6	11	Sparse	BITMAP
Versions	5	6	Sparse	BITMAP
Channel	31	37	Sparse	BITMAP
anchor dimension - 2 constant bitmaps				
Product	2			
anchor dimension - dependant parents				
Product	1			
Calc Cache =	**349 B**	**0 K**	**0.0 M**	**Under Max Size**
Single anchor single bitmap (second best)				
Bitmap Dimensions				
Fiscal Year	6	11	Sparse	BITMAP
Versions	5	6	Sparse	BITMAP
Channel	31	37	Sparse	BITMAP
Calc Cache =	**116 B**	**0 K**	**0.0 M**	**Under Max Size**
Multiple anchor single bitmap (third best)				
Bitmap Dimensions				
Fiscal Year	6	11	Sparse	BITMAP
Versions	5	6	Sparse	BITMAP
Calc Cache =	**4 B**	**0 K**	**0.0 M**	**Under Max Size**

Using The Sizing Spreadsheet

- You can use the ClearLine Sizing Spreadsheet to complete a lot of design work without having to build cube.
- You only need to type on the first tab called the input sheet
- You can only write to the input cells on the input sheet … all other cells and tabs are protected
- The “Sizing” and “Calc Cache” tab use conditional formatting
- You will need to enter in the:
  - Dimension name
  - Number of stored members
  - Total number of member
  - Dense / Sparse Setting
  - Block compression (100% mean no compression – the compressed block and the un compressed are the same size)
  - The dependent number of parents
  - The attribute dimensions are not needed for these calculations

ClearLine Group

ORACLE GOLD PARTNER

Using The Sizing Spreadsheet

- Change the "Block Size After Compression" to 90%

ClearLine Group

Dimension Number	Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
		Stored Members	All Members		
Dimension Position 1	Measures (Accounts)	9	13	Dense	
Dimension Position 2	Period (Time)	66	396	Dense	
Dimension Position 3	Scenario	18	47	Dense	
Dimension Position 4	Fiscal Year	6	11	Sparse	BITMAP
Dimension Position 5	Versions	5	6	Sparse	BITMAP
Dimension Position 6	Channel	31	37	Sparse	BITMAP
Dimension Position 7	Product	3,000	3,000	Sparse	ANCHOR

Estimated Block Density	
Block Size After Compression	90%

Number of Dependent Parents in the Anchor Dimension	
Product	1

Using The Sizing Spreadsheet

- The Calc Cache does not change, but the database size does
- Note the stop lighting on the size tab

Potential Database Size (at 50%)	107,480,844,000 B
Potential Database Size (at 25%)	53,740,422,000 B
Potential Database Size (at 10%)	21,496,168,800 B
Potential Database Size (at 5%)	10,748,084,400 B
Potential Database Size (at 2%)	4,299,233,760 B
Potential Database Size (at .5%)	1,074,808,440 B
Potential Database Size (at .05%)	107,480,844 B
Potential Database Size (at .01%)	21,496,169 B
Potential Database Size (at .005%)	10,748,084 B
Potential Database Size (at .001%)	2,149,617 B
Potential Database Size (at .0005%)	1,074,808 B
Potential Database Size (at .0001%)	214,962 B
Potential Database Size (at .00005%)	107,481 B
Potential Database Size (at .00001%)	21,496 B
Potential Database Size (at .000005%)	10,748 B
Potential Database Size (at .000001%)	2,150 B
Potential Database Size (at .0000005%)	1,075 B
Potential Database Size (at .0000001%)	215 B

Using The Sizing Spreadsheet

- Change the position of Products and Channel

ClearLine Group

Dimension Number	Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
		Stored Members	All Members		
Dimension Position 1	Measures (Accounts)	9	13	Dense	
Dimension Position 2	Period (Time)	66	396	Dense	
Dimension Position 3	Scenario	18	47	Dense	
Dimension Position 4	Fiscal Year	6	11	Sparse	BITMAP
Dimension Position 5	Versions	5	6	Sparse	BITMAP
Dimension Position 6	Product	3,000	3,000	Sparse	BITMAP
Dimension Position 7	Channel	31	37	Sparse	ANCHOR

Estimated Block Density	
Block Size After Compression	90%

Number of Dependent Parents in the Anchor Dimension	
Channel	1

Using The Sizing Spreadsheet

- This did not change anything with either the block size or the potential number of data blocks created.
- It did have an impact on the amount memory required for the Calculator Cache

Before Change

Single anchor dimension multiple bitmap (best)	
BITMAP DIMENSIONS	
Fiscal Year	6
Version	5
Channel	31
anchor dimension - 2 constant bitmaps	
Products	2
anchor dimension - dependant parents	
Products	1
Calc Cache =	349 B

Less memory is used with the original hourglass design

After Change

Single anchor dimension multiple bitmap (best)	
BITMAP DIMENSIONS	
Fiscal Year	6
Version	5
Products	3,000
anchor dimension - 2 constant bitmaps	
Channel	2
anchor dimension - dependant parents	
Channel	1
Calc Cache =	33,750 B

The non-hourglass design uses memory less effectively

ClearLine Group

ORACLE GOLD PARTNER

Using The Sizing Spreadsheet

- Add a dimension called Location with 30,000 stored members with 5 dependent parents, increase Channel to 1,310 stored members, and increase Product to 13,000 stored members

ClearLine Group

Dimension Number	Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
		Stored Members	All Members		
Dimension Position 1	Measures (Accounts)	9	13	Dense	
Dimension Position 2	Period (Time)	66	396	Dense	
Dimension Position 3	Scenario	18	47	Dense	
Dimension Position 4	Fiscal Year	6	11	Sparse	BITMAP
Dimension Position 5	Versions	5	6	Sparse	BITMAP
Dimension Position 6	Product	1,310	1,400	Sparse	BITMAP
Dimension Position 7	Channel	13,000	14,000	Sparse	BITMAP
Dimension Position 8	Location	30,000	52,000	Sparse	ANCHOR

Estimated Block Density	
Block Size After Compression	90%

Number of Dependent Parents in the Anchor Dimension	
Location	5

Using The Sizing Spreadsheet

- You will be at high risk for the database growing too large and you will not be able to use the best calc cache

Potential Database Size (at 50%)	590,451,217,200,000,000 B
Potential Database Size (at 25%)	295,225,608,600,000,000 B
Potential Database Size (at 10%)	118,090,243,440,000,000 B
Potential Database Size (at 5%)	59,045,121,720,000,000 B
Potential Database Size (at 2%)	23,618,048,688,000,000 B
Potential Database Size (at .5%)	5,904,512,172,000,000 B
Potential Database Size (at .05%)	590,451,217,200,000 B
Potential Database Size (at .01%)	118,090,243,440,000 B
Potential Database Size (at .005%)	59,045,121,720,000 B
Potential Database Size (at .001%)	11,809,024,344,000 B
Potential Database Size (at .0005%)	5,904,512,172,000 B
Potential Database Size (at .0001%)	1,180,902,434,400 B
Potential Database Size (at .00005%)	590,451,217,200 B
Potential Database Size (at .00001%)	118,090,243,440 B
Potential Database Size (at .000005%)	59,045,121,720 B
Potential Database Size (at .000001%)	11,809,024,344 B
Potential Database Size (at .0000005%)	5,904,512,172 B
Potential Database Size (at .0000001%)	1,180,902,434 B

Single anchor dimension multiple bitmap (best)	
BITMAP DIMENSIONS	
Fiscal Year	6
Version	5
Channel	1,310
Products	13,000
anchor dimension - 2 constant bitmaps	
Location	2
anchor dimension - dependant parents	
Location	5
Calc Cache =	***447,037,500 B***

Single anchor single bitmap (second best)	
BITMAP DIMENSIONS	
Fiscal Year	6
Version	5
Channel	1,310
Products	13,000
Calc Cache =	**63,862,500 B**

Multiple anchor single bitmap (third best)	
BITMAP DIMENSIONS	
Fiscal Year	6
Version	5
Channel	1,310
Calc Cache =	**4,913 B**

Using The Sizing Spreadsheet

- Now lets make a some of changes to see if we can get it back to a more reasonable size:
  - Remove the alternate rolls in Location and Product
    - Set the dependent parents back to 1
    - Set Product back to 3,000 stored and 3,000 members
    - Set Location 5,000 stored and 5,000 members
  - Change the block compression to 40%
  - Store only 12 months in time
  - Cut it back to 12 Scenarios

Using The Sizing Spreadsheet

- Now it should look like this…

ClearLine Group

Dimension Number	Dimension Names	Outline Members		Dense / Sparse Setting	Anchor / Bitmap
		Stored Members	All Members		
Dimension Position 1	Measures (Accounts)	9	13	Dense	
Dimension Position 2	Period (Time)	12	17	Dense	
Dimension Position 3	Scenario	12	25	Dense	
Dimension Position 4	Fiscal Year	6	11	Sparse	BITMAP
Dimension Position 5	Versions	5	6	Sparse	BITMAP
Dimension Position 6	Product	1,310	1,400	Sparse	BITMAP
Dimension Position 7	Channel	3,000	3,000	Sparse	BITMAP
Dimension Position 8	Location	5,000	5,000	Sparse	ANCHOR

Estimated Block Density	
Block Size After Compression	40%

Number of Dependent Parents in the Anchor Dimension	
Location	1

Using The Sizing Spreadsheet

- This is more likely to work

Potential Database Size (at 50%)	1,223,802,000,000,000 B
Potential Database Size (at 25%)	611,901,000,000,000 B
Potential Database Size (at 10%)	244,760,400,000,000 B
Potential Database Size (at 5%)	122,380,200,000,000 B
Potential Database Size (at 2%)	48,952,080,000,000 B
Potential Database Size (at .5%)	12,238,020,000,000 B
Potential Database Size (at .05%)	1,223,802,000,000 B
Potential Database Size (at .01%)	244,760,400,000 B
Potential Database Size (at .005%)	122,380,200,000 B
Potential Database Size (at .001%)	24,476,040,000 B
Potential Database Size (at .0005%)	12,238,020,000 B
Potential Database Size (at .0001%)	2,447,604,000 B
Potential Database Size (at .00005%)	1,223,802,000 B
Potential Database Size (at .00001%)	244,760,400 B
Potential Database Size (at .000005%)	122,380,200 B
Potential Database Size (at .000001%)	24,476,040 B
Potential Database Size (at .0000005%)	12,238,020 B
Potential Database Size (at .0000001%)	2,447,604 B

Single anchor dimension multiple bitmap (best)	
BITMAP DIMENSIONS	
Version	5
Channel	1,310
Products	3,000
anchor dimension - 2 constant bitmaps	
Location	2
anchor dimension - dependant parents	
Location	1
Calc Cache =	7,368,750 B

Single anchor single bitmap (second best)	
BITMAP DIMENSIONS	
Version	5
Channel	1,310
Products	3,000
Calc Cache =	2,456,250 B

Multiple anchor single bitmap (third best)	
BITMAP DIMENSIONS	
Version	5
Channel	1,310
Calc Cache =	819 B

ClearLine Group

ORACLE GOLD PARTNER

What is the bottom line?

- Think about the impact of your design and/or enhancement before you start building it.
  - Make sure that the proposed outline design can work BEFORE you build anything
  - Avoid making promises that you can not deliver.
  - A spreadsheet can be a great tool for communicating to clients and project managers who do not understand the impact of adding members and dimensions.

ClearLine Group

Steve Light Contact Information

slight@clearlinegroup.com

Please see me after the presentation if you would like a copy of it or of the ClearLine Sizing Spreadsheet.

ClearLine Group

ORACLE GOLD PARTNER

ClearLine Group

ClearLine Group, L.L.C.
250 Parkway Drive
Suite 150
Lincolnshire, IL 60069

Call:
855.ClrLine (855.257.5463)

Online:
slight@clearlinegroup.com
www.clearlinegroup.com