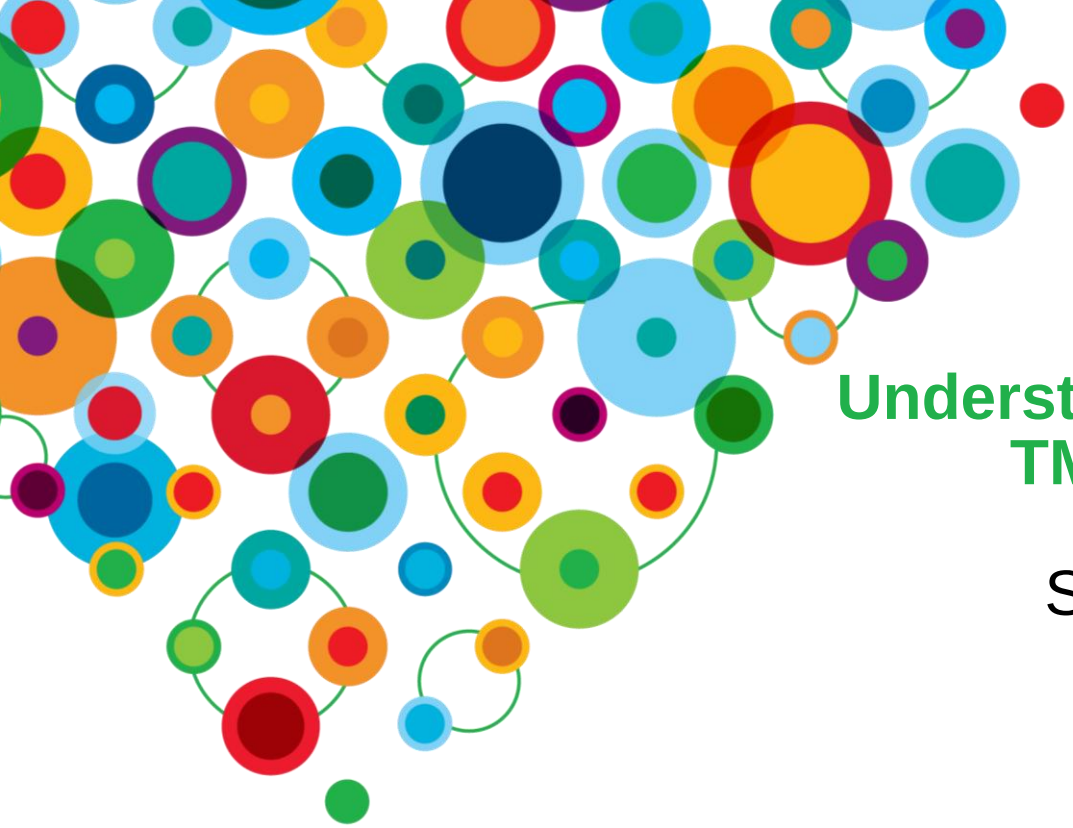


Understanding IBM Cognos TM1 Performance and Scalability

Session Number 3180

John van den Berg, IBM

Information On Demand 2010

The Premier Forum for Information & Analytics

Gain Insight. Optimize Results.

Disclaimer

.

Goals for Today

1. Categorize TM1 application types and performance components
   - Identify roadblocks to high performance
   - Describe how to go over, around or through roadblocks by:
     - Leveraging released TM1 9.5.x performance capabilities
     - And, deploying model and process techniques

2. Illustrate high levels of TM1 performance being achieved today

3. Describe a methodology for achieving high performance in your TM1 environment

High-Lights recent TM1 Improvement

Agenda

- Follow an "*Amalgamated*" Case Study for improving query, contribution and operations performance across a TM1 application
  - Before – what were the symptoms and causes of bad performance?
  - After – What improvements could be achieved?
  - Take-Aways
- Present a general application methodology to achieve a high performance TM1 application

Amalgamated Case Study
Could this be you?

An all too common story. Things started out great…

- I built a POC model:
  - Got some specifications
  - Created Dimensions and Cubes and some rules
  - Wrote some TI processes to load some sample data
  - Created Cubeviews, Reports and Input screens
- Queries and Inputs: everything ran fast
- I worked with the team to set SLAs for response times and system availability
- And the implementation began

Amalgamated Case Study Symptoms

But as the Go Live date approached…

- As we neared our implementation target performance started to lag (a lot):
  - For some users, some of the time, queries that used to be instant were now running 5, 10, 15 or many more seconds.
  - Looking in TM1Top I saw my planning users began stacking up behind these slow queries.
  - Some users query and update performance was always much slower
  - Server startup and Nightly processing are extending too long, I'm not meeting my availability SLA
  - My users are losing faith, my manager is not happy.
  - Oh yeah, the application's memory consumption has really grown, do I have sufficient RAM on my server?
- What happened? Why? What do we do?

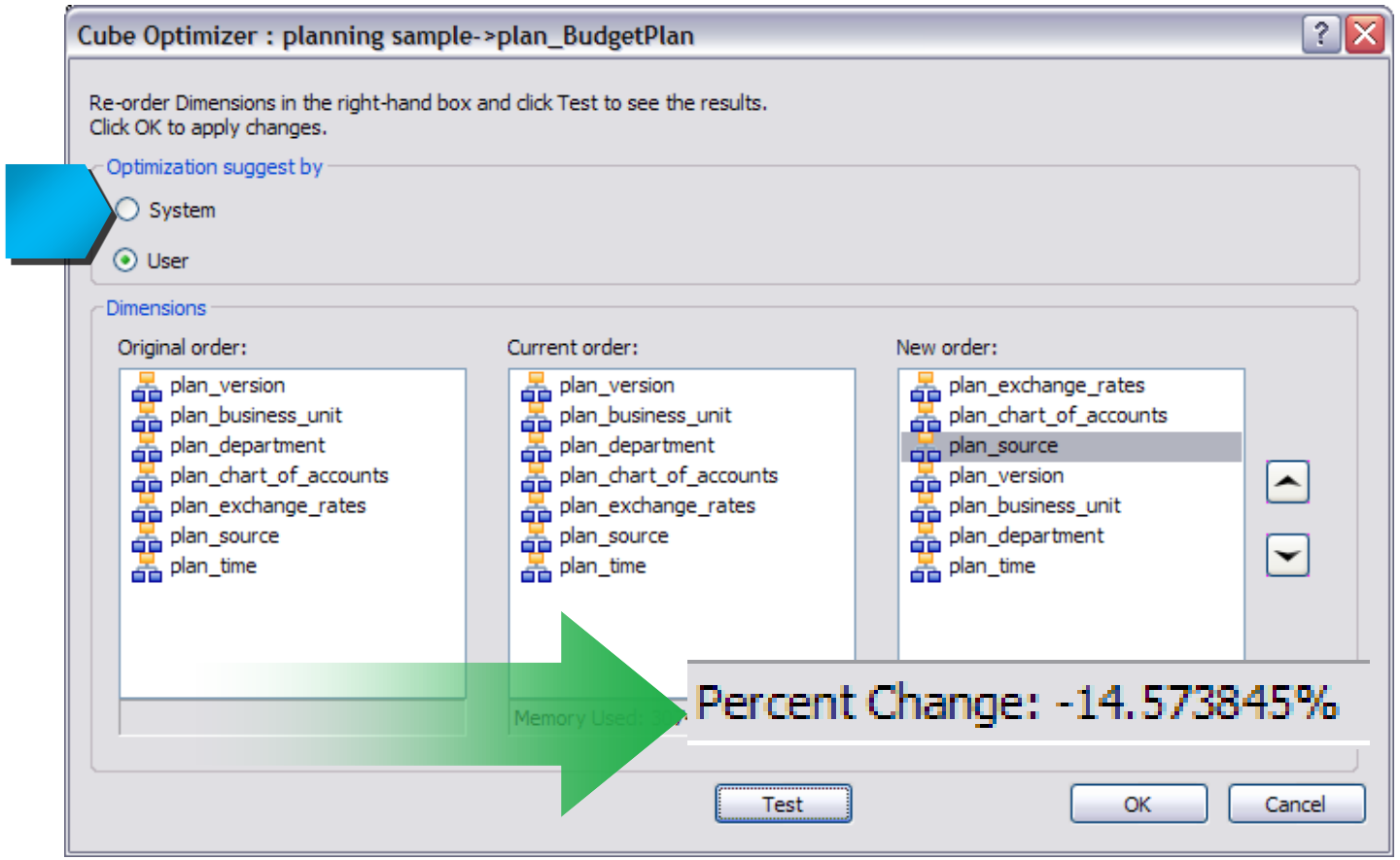
Amalgamated Case Study
First Step to Better Performance – Fix Query Times

Fully loading the model with data has exposed several design inefficiencies. Indicative symptoms are significantly slower query times and significantly increased memory consumption.

- Likely Causes:
  - Non-optimal cube dimension order
  - OVERFEEDING
  - Unnecessary calculations

Amalgamated Case Study
Fix Query Times – Optimize Dimension Order

- Reordering cube dimensions may give you an easy one time significant memory reduction and increased query efficiency



Amalgamated Case Study
Fix Query Times - OVERFEEDING

- OVERFEEDING = Many, many (, many) useless, wasted calculations
- Simple sample:

```
FEEDERS;

['Sales']=>DB('ProfitLossCube', !product, !timeperiod,
    'All Geographies', 'Sales Amount');
```

Danger

- Feeder statements should be as precision as possible:

```
FEEDERS;

['Sales']=>DB('ProfitLossCube', !product, !timeperiod,
        ATTRS('Customer', !customer, 'Geog'),
        'Sales Amount');
```

Amalgamated Case Study
Fix Query Times - Unnecessary calculations

- TM1 rules are fast, but straight dimensional aggregation is much faster.
- High level queries can require millions of low level calculations.
- Whenever possible remove or replace unnecessary calculations & consolidations:
  - For example:
    - If your time dimension contains years and periods before or after your useful data (actual or planning) remove the extraneous time periods.
    - If you have useless top level consolidations (often ‘All Years’ in a time dimension serves no purpose) remove them. They just invite unneeded aggregation and force unneeded calculation.
  - Replace “one time only” rules calculations that will never be changed by user input with “stored” calculations where possible. Removing these on-demand calculations can improve some query times by orders of magnitude!

Take Aways & Tips

The fundamental key to TM1 performance is query speed. Always consider the following:

- Size of Model (dimension size and order, data volume)
- Nature and Complexity of calculations

Keys to Performance

- Always Optimize Dimension Order
- Optimize Rules
  - Avoid using Rules for “static” calculations
  - Beware OVERFEEDING!

Amalgamated Case Study
Second Step – Deal with Object Contention

Inconsistent query times. Readers blocking Writers. Writers blocking each other. Strangest of all Readers blocking Readers…

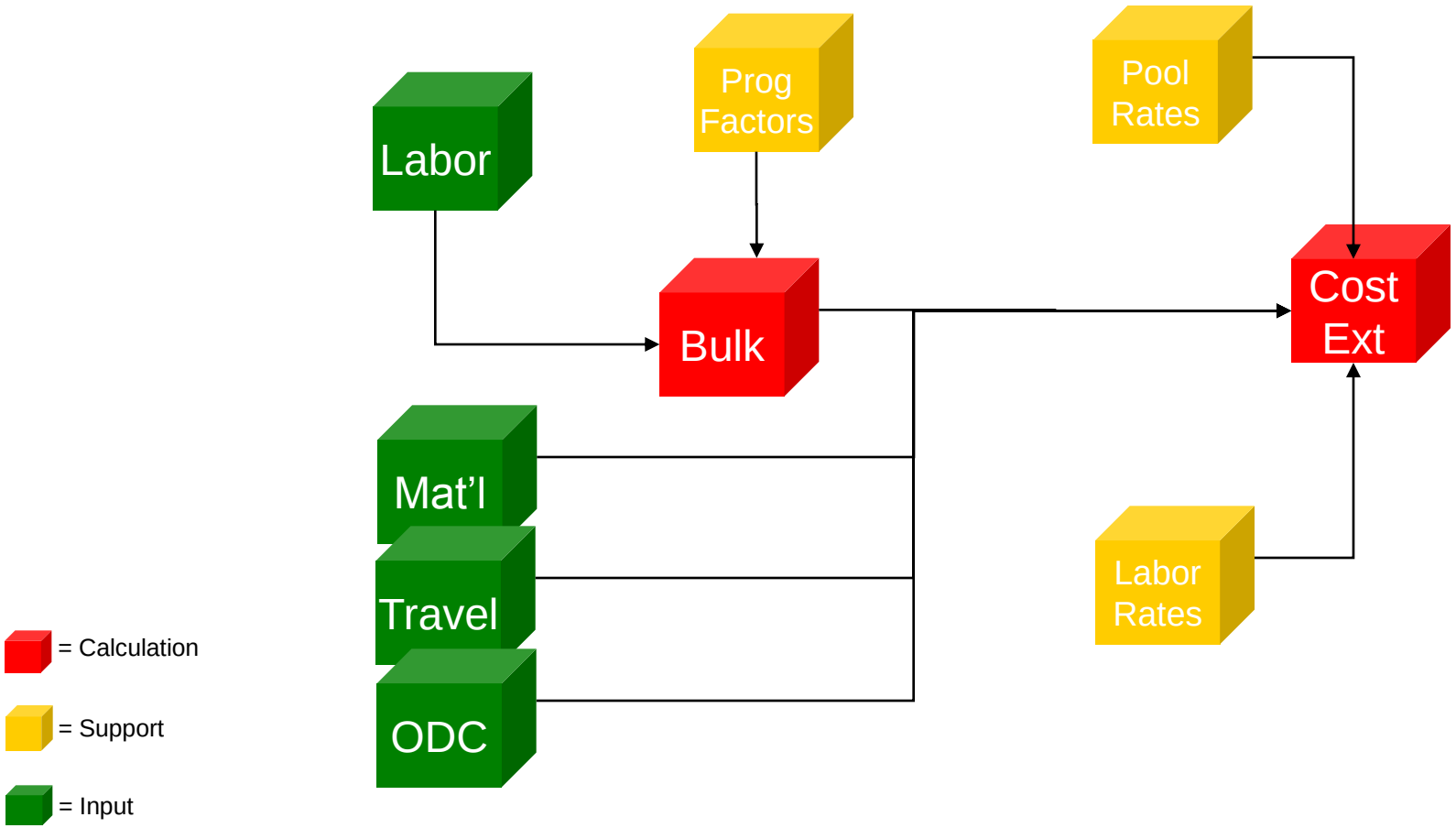
- Understand the TM1 locking and object dependency model
  - Readers block writers at the Cube/Dimension level
  - TM1 Cube Rules create multi-cube dependencies and multi-cube locks
  - Some Read operations require Write locks
- Work around locking by separating components that will be accessed concurrently by many users.
  - Separate cubes by logical function (employee plan, capital plan, driver based revenue…)
  - Use separate cubes to separate read/only users from contribution users
  - Separate writers from one another and other readers
    - Sandboxes/Personal Workspace capability

Concurrency That Has Been Achieved

- US Financial Services ~1,000 Users
- Australian Financial Services ~1,600 Users
- Large Manufacturer: ~1,500 Users
- Large Electronics Company: ~2,500 Users

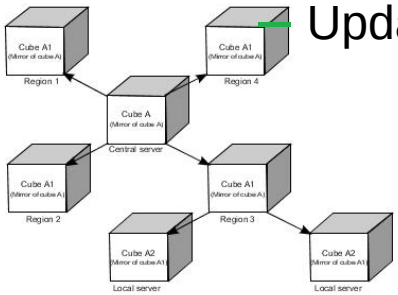
Amalgamated Case Study
Deal with Object Contention – Separate Cubes By Function

- Separate cubes by logical function (employee plan, capital plan, driver based revenue…)

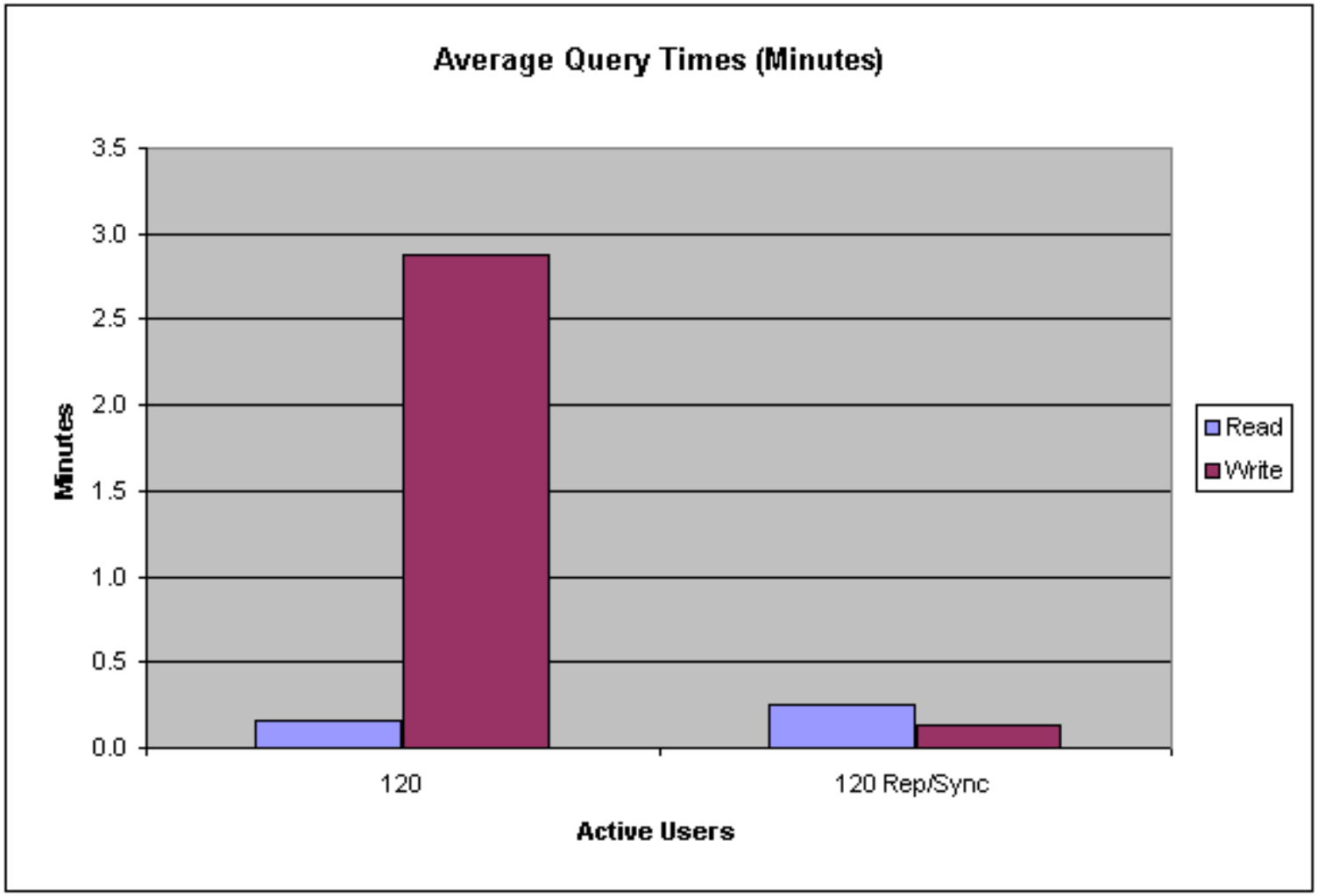


Amalgamated Case Study
Deal with Object Contention – Read/Only Cubes

- Use separate cubes to separate read/only users from contribution users
  - Create Read/Only Cubes on separate servers using Rep/Sync
    - Built into server functionality
  - Create same server Read/Only Cubes
    - Limit Cube rules to a minimum by pre-calculating everything at the leaf level on load. This will produce huge speed improvements for high level queries and drastically limit the negative impact of query cache invalidation.
    - Update Frequency depends on workflow requirements:
      - Update nightly from Input Cubes
      - Update slices on demand as part of a workflow submission.

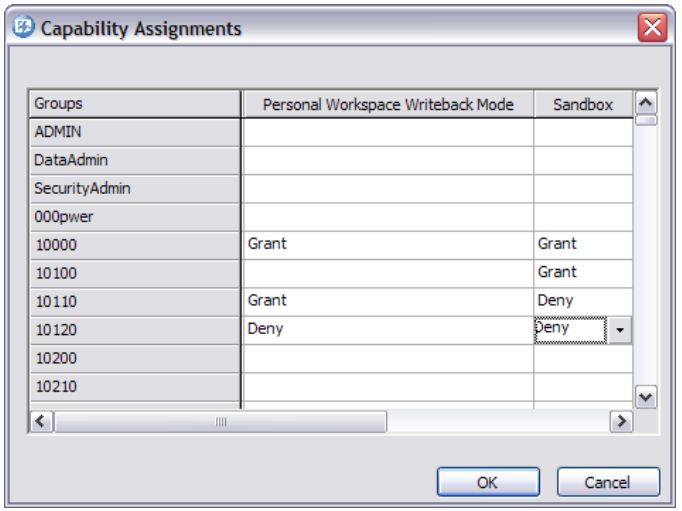


Effect of Read/Only Separation



Amalgamated Case Study
Deal with Object Contention - Separate writers from one another

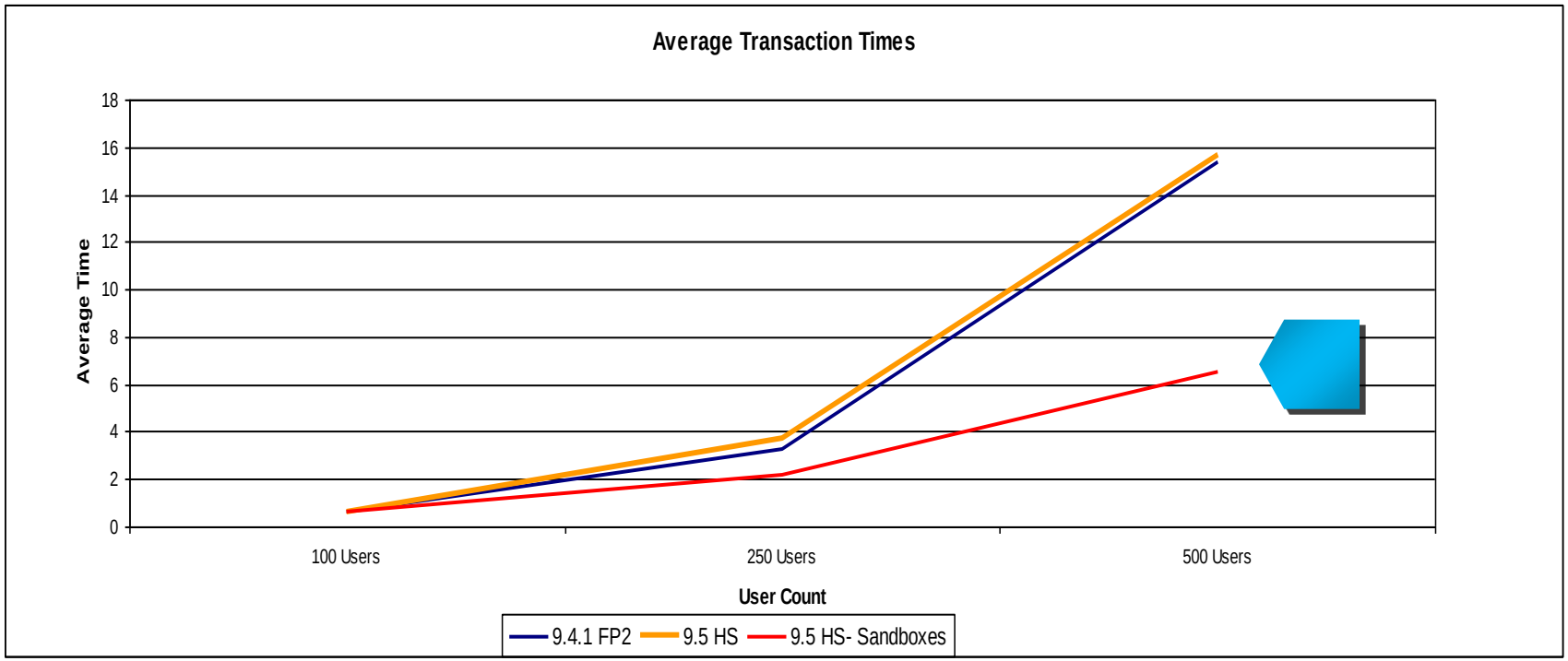
- TM1 9.5.x introduces Sandbox/Personal Workspace. Great new contribution functionality and writers don't lock "base" cubes until they commit their changes.
  - TM1 9.5 introduces Sandboxes capability.
  - TM1 9.5.1 augments with Personal workspace capability and Job Queuing



- Prior to TM1 9.5 writer separation can be achieve by "partitioning" input cubes.
  - Complex to implement but significantly decreases object contention and increases concurrent user activity.

Effect of Separating Writers with 9.5 Sandboxes

- Contribution into sandboxes avoids cube contention until users merge their sandbox numbers into the base cubes. Queued sandbox commits (9.5.1) further increase response times.
- Significantly better, more predictable performance is achieved at high user counts.



Amalgamated Case Study
Deal with Object Contention – Readers Blocking Readers

Beware of deployment of TM1 objects that require write locks for read operations:

- Dynamic Subsets
  - Require a write lock to evaluate
  - Are in constant need of re-evaluation (particularly in contribution applications)
  - Can be used for nightly updates of “semi-dynamic” subsets.
- User Defined Consolidations and }Rollups
  - Queries evaluating UDCs and Rollups can block Logins due to private object registration write locks
- Where possible, avoid deploying.

Take-Aways & Tips

Understand TM1's object locking model and its impact on user concurrency. Design for concurrency.

- Account for Object Locking
  - Understanding what activities block other activities
  - Sandboxes and Personal Workspace
    - Job Queuing
  - Partitioning of Writers
    - Application oriented (Employee Cube vs. CAPEX cube…)
    - Physical cube partitioning

Keys to Performance

- Leverage Calculation and Query Cache
  - Understanding when cache is invalidated
  - Use of "Read Only" cubes
    - Replication/Synchronization
    - Turbo Integrator based
- Beware Dynamic Subsets and UDCs

Amalgamated Case Study
Step Three – Operations Performance & Environmental Factors

With basic query performance corrected and object contention issues dealt with we still have various operational/environmental issues to deal with:

- Server Startup Too Slow
- Nightly Loads Too Long
- Do we have the correct Server environment?
  - TM1 Server
  - TM1Web
- Why are some clients always so slow?
  - WAN/LAN
  - TM1Web/Contributor
  - Excel

Amalgamated Case Study Operations - Server Startup

Keys to Performance

- Multi Threaded Startup (for pre 9.5.1)
  - There are some caveats (no conditional feeders)
- Persistent Feeders (TM1 9.5.1)!

  - Can decrease startup time by a factor of 10
    - Depends on size/use of Feeders
    - You still want to optimize your feeders!
  - Cautions
    - Disk considerations, .FEEDER files can be quite large.
    - Major rules file updates will still require a “full” restart.

Amalgamated Case Study Operations – Nightly Loads

Keys to Performance

- Data & Meta Data Import
  - "Multi Threaded" data loads
    - Imported data must be separate for each thread
    - Imported data must overwrite existing cells (no accumulate)
    - Batch Mode (BatchUpdateStart, BatchUpdateFinishWait)
  - Turn Off Transaction Logging (careful)
  - Bulk Load Mode for "big" dimension updates to avoid any chance of contention and rollback.

Amalgamated Case Study
Operations – Nightly Loads & Action Buttons

Some TI Processes may run concurrently and can't leverage "Batch" mode.

- Two or more long running processes that update/read the shared objects will contend with each other.
- Blocked processes will "Rollback" all operations prior to the block wait and then retry. This creates serious CPU drain and a locking Log Jam.

The solution – ""

- For TI processes that may contend (launched from Action Buttons or possibly conflicting Chores schedules) introduce an immediate locking cube write action to limit rollback.
- Create a small cube with 2 dimensions (semaphore.cub)
- In line one of the TI prolog write to the semaphore cube:

```
CellPutS('somestring', 'semaphore', 'element1', 'element2');
```

Amalgamated Case Study
Operations – Server Environment

TM1 Server - Keys to Performance

- For higher user concurrency, more CPU cores are important.
  - Beware object contention. Extra CPUs will only help if users are not blocked on object locks.
- For Larger Models more memory is required
  - NO SWAPPING TO DISK!
  - NOT currently NUMA aware
- Virtualization
  - Optimized Virtual Environments work
  - You STILL need sufficient CPUs and Memory

Hardware Configuration Examples

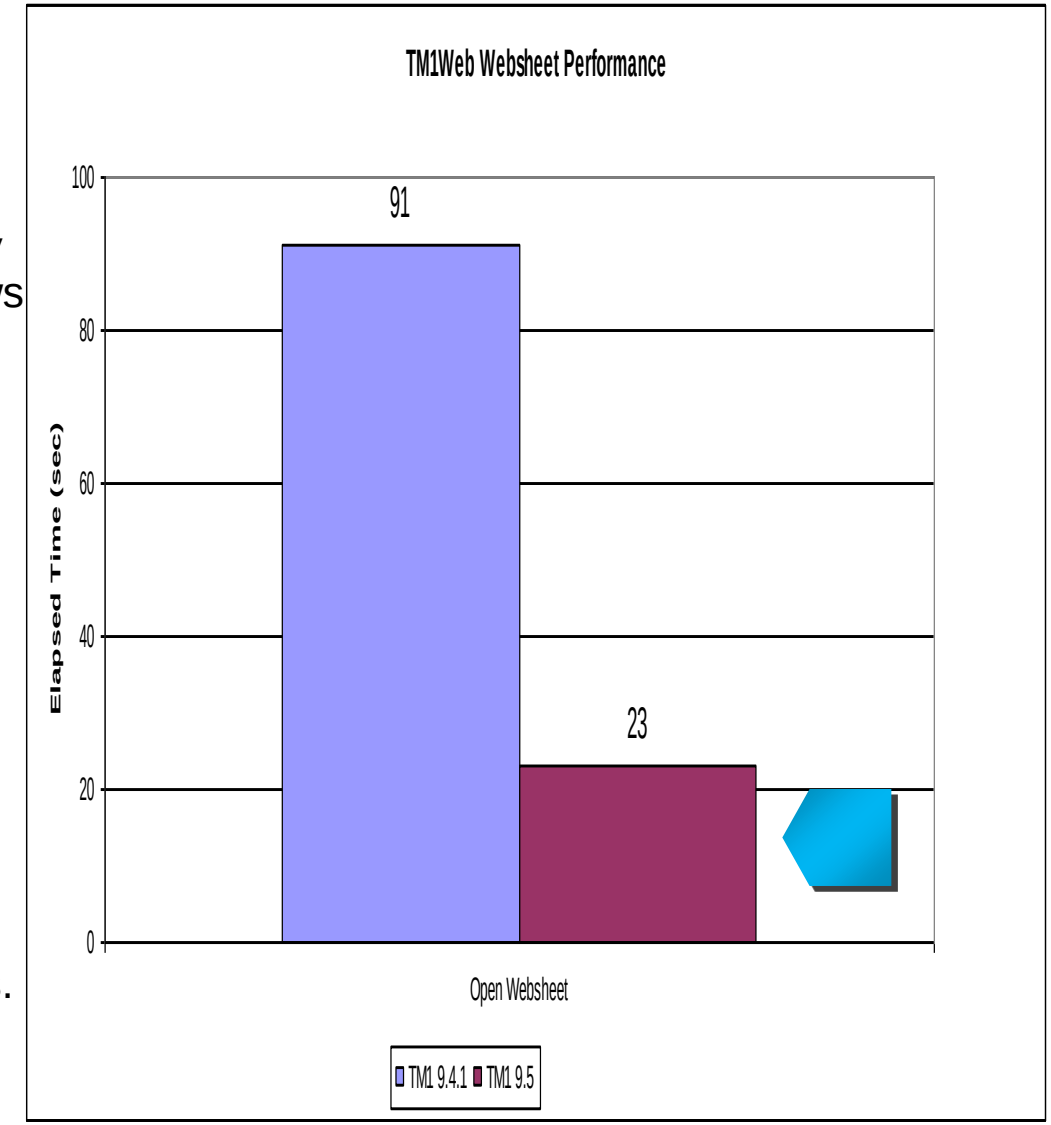
- **Large Manufacturer**
- Users: ~1,500
- TM1 Server:
  - 16 Cores: 8 dual-core Intel Xeon x64 @ 2.93 GHz
  - 128 GB of RAM
  - Windows 2003 Server Enterprise Edition 64-bit
- Web Servers (3):
  - 16 Cores: 8 dual-core Intel Xeon x64 @ 2.4 GHz
  - 32 GB of RAM
  - Windows 2003 Server Enterprise Edition 64-bit

- **Large Electronics Company**
- Users: ~2,500
- TM1 Server:
  - 16 Cores: 8 x Dual Core Intel Xeon x64
  - 128 GB of RAM
  - Windows 2003 Server Enterprise Edition 64-bit
- Web Servers (4):
  - 4 x Dual Core Intel Xeon x64
  - 64GB RAM
  - Windows 2003 Server Enterprise Edition 64-bit

Amalgamated Case Study Operations – Server Environment

TM1Web Server (IIS) - Keys to Performance

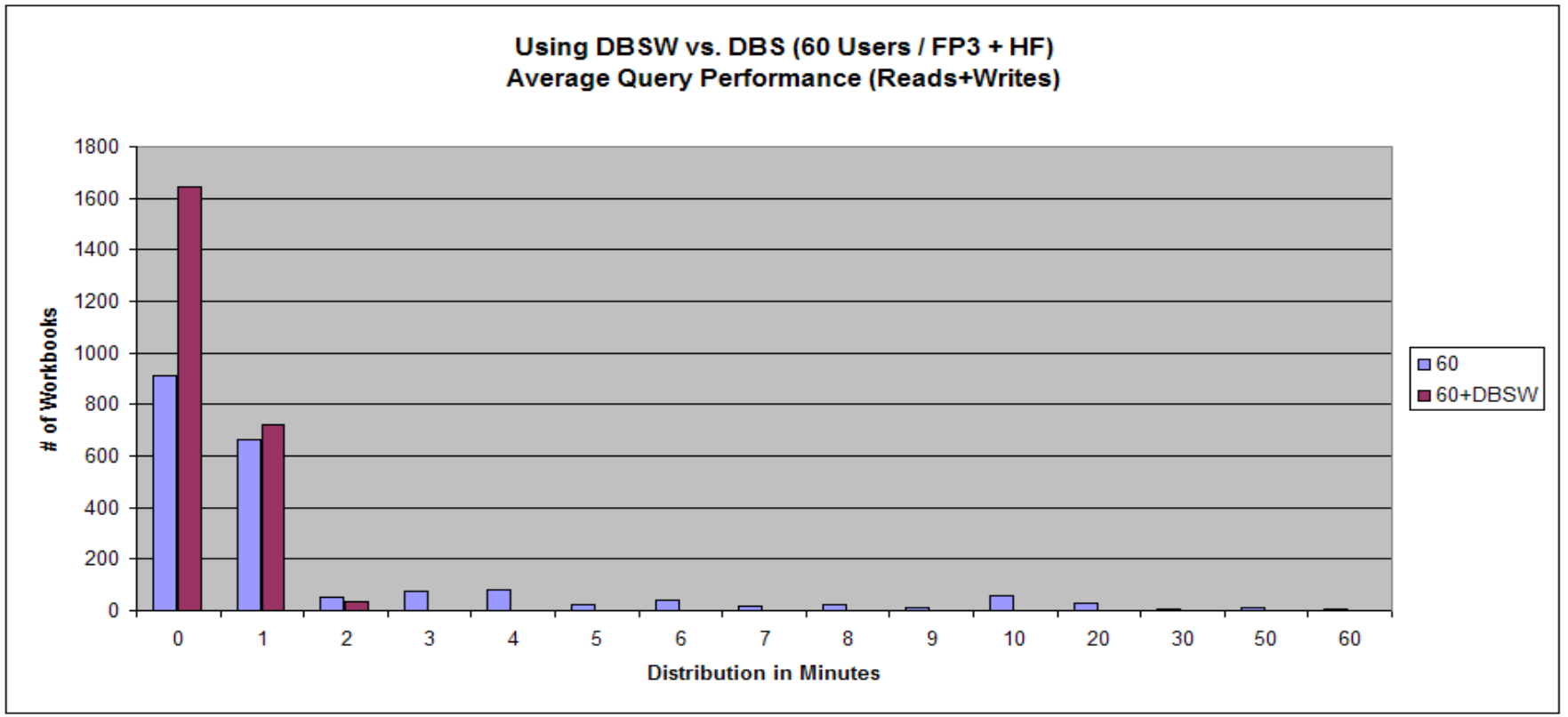
- 64 Bit Windows allows more memory which in turn allows more users
- Websheets typically require significantly more TM1Web resource than Cubeviews
- Websheet Paging to optimize (9.5.1)
- Locate the TM1Web server (IIS) on the LAN with the TM1 Server to avoid WAN related performance issues.
- Multiple Web servers
  - Allows horizontal scaling of web environment
  - Standard load balancers can be used but do need to support a “sticky” connection.
  - Multiple Virtual Directories on server hosts can help, particularly with Websheet based applications.



Amalgamated Case Study
Operations – Client Environments

TM1 formula optimization for Excel sheets

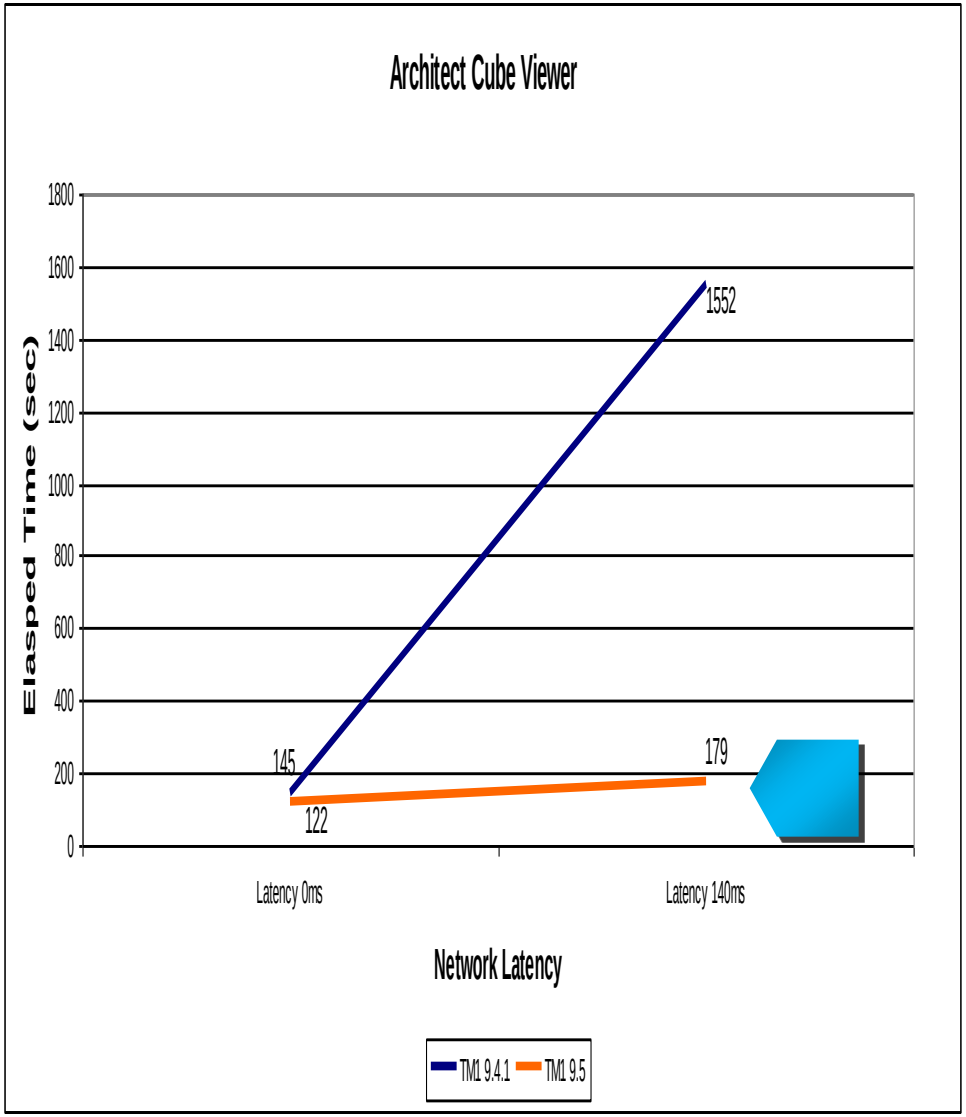
- Use the VIEW() formula instead of direct Cubeview name.
- Use DBRW() instead of DBR()
- Use DBSW() instead of DBS()



System Operation Optimizations
Client Environments

WAN/LAN

- In a WAN environment many features of the TM1 Architect and Perspectives clients do not perform well.
  - We have a ways to go…
- Strategies to improve performance for WAN users:
  - Implement the application in TM1Web for WAN users if possible
  - Have remote Architect and Perspectives users run via terminal services
  - Optimize Perspectives sheet design



End of Amalgamated Case Study

Methodology for Performance Rules of Thumb

- Understand the likely performance issues in your application.
- Build cubes with efficient dimensional design
- Avoid the “rules” trap!
  - A model is fast with a small data sample and one user may have significant issues under a full data load or when hit by multiple users.
  - For calculations that don’t change or only change in sync with external data loads consider storing calculations via Turbo Integrator instead of Real Time calculation with Rules.
- Beware model components that can create contention:
  - Dynamic Subsets
  - User Defined Consolidations

Methodology for Performance Test for Speed

- Once the model or components of the model are build it is imperative to test and tune.
- Test with a full load of data.
  - If you have a full set of “live” data, use it.
  - If you don’t have the real data, make it up.
  - Fully populated cubes can expose rule, feeder and dimensional inefficiencies that must be corrected before going live.
  - Test query times with un-cached cubes.
  - Test Turbo Integrator performance.
    - Make sure to run a full “overnight” scenario
  - Test startup time (with and without Persistent Feeders)

Methodology for Performance Test for Concurrency

- Once the single user performance has been properly established begin concurrency testing.
- Test with a full user load.
  - Once the application is built and populated run a series of multi-user tests exercising any/all functions of the system.
    - Record base line performance numbers with a single user.
    - Begin adding progressively more monitoring performance and TM1Top activity
- Automated Testing
  - If possible use a multi-user test automation tool to allow continuous test, improve, retest cycles without taxing your users' patience.
  - Appropriate tools depend very much on the nature of your application (web, Excel, custom…) and your budget.

Parting Thoughts…

IBM Cognos is continually considering how best to:

- Allow quicker, easier development of high performance applications
- Achieve even higher levels of performance

But For Now:

- Consider your application and performance pain points
- Implement one or more methods to improve performance
- Test/Adjust
- Repeat until happy

Information and Analytics Communities

- **On-line communities, User Groups, Technical Forums, Blogs, Social networks, and more**
  - Find a community that interests you at…
    - **ibm.com**/software/data/community
- Integration with TM1
- **Information Champions**
  - Recognizing individuals who have made the most outstanding contributions to Information Management communities
    - **ibm.com**/software/data/champion

So lets see a real life example

Information On Demand 2010
The Premier Forum for Information & Analytics

Gain Insight. Optimize Results.

So who is it?

Sorry. We can't reveal the retailer's identity as this project is still in early go-live / final UAT.

However, rest assured this is a real case study!

1. The reporting problem pre-TM1
2. Proof of concept and volume testing
3. Reporting requirements
4. Model design considerations
5. ***"Blackbelt"*** techniques to pull it off

- The source OLTP system ***(JDA ODBMS)*** ran the business well but fell down when it came to reporting
- The business's primary merchandise reporting tool was Excel
- Reports at the level of detail the business required were only available in the MCA reporting module at a low level. There was no concept of "drilling down" from above. This lead to:
  - Analysts having to run multiple reports with multiple sessions of MCA to dump out data then piece together and collate reports in Excel
  - Significant strain on the source system due to the volume of queries
  - Complicated Excel reports with macros and pivot tables that were subject to breakage
  - ... an "Excel Hell" scenario that we are all familiar with

- ***Proof of Concept***
  - 8 weeks of data at store by product by week level
  - 24 measures
  - 16 dimensional cube with many product attributes included as dimensions for slice and dice analysis
  - Excel cube viewer the primary interface
  - Around 30 sec response times on high level views
  - “Beauty contest” with another BI tool
  - TM1 selected due to ease of use and Excel integration
- Proof of technology
  - 3 days in HP data labs in Sydney
  - Expand data set to 100 weeks, 100+ GB
  - Prove stable at 100+ GB with no loss in query response time

- ***Hold 2 years of data at product/variation level by store by week***
- Hold 6 weeks of daily data at product/variation by store
- Be able to summarize and “slice & dice” measures based on product attributes (status, ranging, size, colour, season, vendor, etc.)
- Choose different levels in same report and able to traverse – drill up and down all levels of the product and location hierarchies
- Sub 30 sec query response times

- Eliminate reporting bottlenecks. Save time
- Eliminate use of ***MS Access***
- ***Availability of vs. LY and MAT calculations (plus many other calcs)***
- Exception based reporting
- Eliminate need to add or remove rows and adjust formulas when master data changes
- Provide common reporting language and source of truth. Central location for all reports. Centralise reporting team
- User friendly access to information

- Size! Over 1 million active products, hundreds of stores
- Billions of records. Initial calculations pointed to a model of ***100 – 200 GB***
- Managing query response times to user acceptable levels
- Managing data loads within very limited batch processing windows (2 – 7am)
- Managing many administration and “master data” type processes internally within TM1. E.g.
  - Backdated transactions
  - Product re-class events
  - Product status tracking

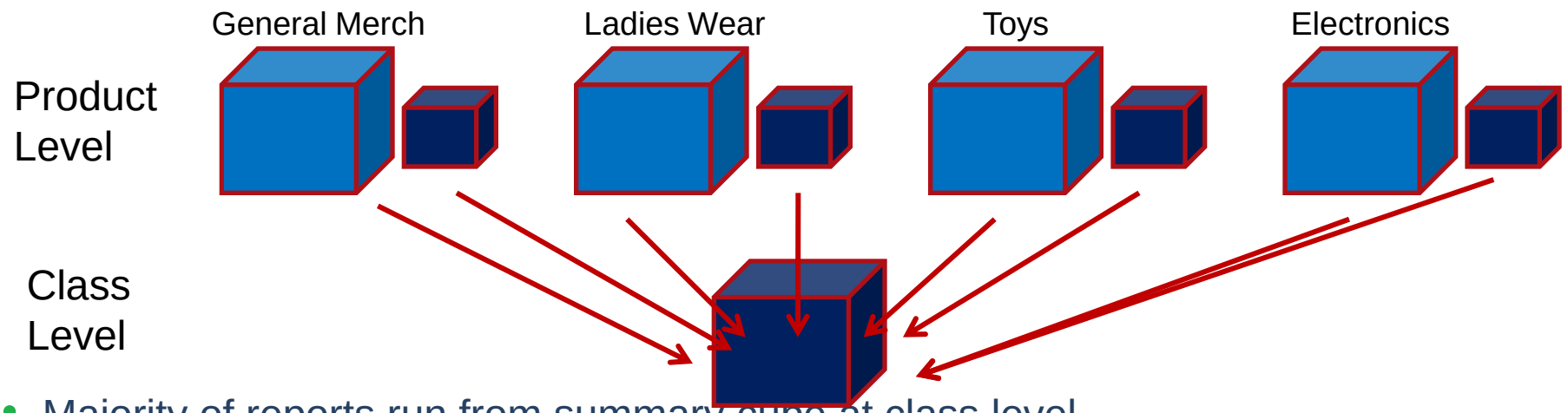
- "***Distributed" or "partitioned" model***
  - Break into smaller cubes based on business reporting structures not one mega cube
- Why?
  - Manage cube and dimension size for better query performance
  - Faster server load times on multiple threads
  - Parallel data loads for massive reduction in load times
- But still need to support business wide reports

- This is a BIG data model

Name	Memory Used	# Dimensions
Stock_and_Sales	25106742KB	21
Stock_and_Sales	21750503KB	21
Stock_and_Sales	17845430KB	21
Stock_and_Sales	16917111KB	11
Stock_and_Sales	12224182KB	21
Stock_and_Sales	11911798KB	21
Stock_and_Sales	10836662KB	21
Stock_and_Sales	8827894KB	21
Stock_and_Sales	7556791KB	8
Stock_and_Sales	7526710KB	21
Stock_and_Sales	7252199KB	21
Stock_and_Sales	2791294KB	11
Stock_and_Sales	2408766KB	11
Stock_and_Sales	2362110KB	11
Stock_and_Sales	2236478KB	11
Stock_and_Sales	2181110KB	21
Stock_and_Sales	2038014KB	11
Stock_and_Sales	1647038KB	11
Stock_and_Sales	1636414KB	11
Stock_and_Sales	1439614KB	8
Stock_and_Sales	1384830KB	11
Stock_and_Sales	1187326KB	8
Stock_and_Sales	1130622KB	11
Stock_and_Sales	942526KB	8

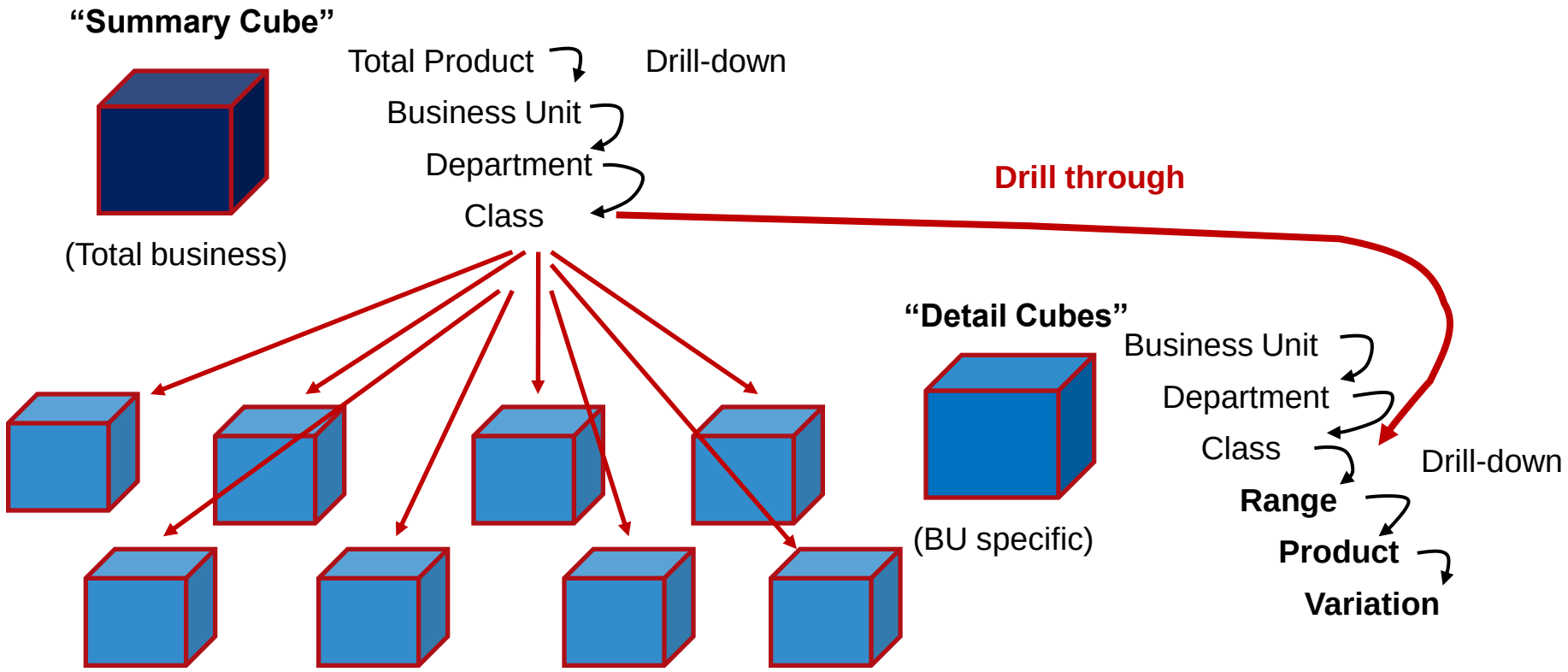
Name	Memory Used	# Subsets	# Elements
Product_	662538KB	11	1369316
Product_	177030KB	2	446532
Product_	155782KB	10	310390
Product_	145994KB	10	281091
LayBy_No	125766KB	1	463766
Product_	82694KB	10	167526
Product_	71878KB	10	148174
PO_No	63942KB	1	178461
Product_	57094KB	11	119329
Product_	32134KB	10	70111
Product_	28870KB	10	58136
Product_	22982KB	10	48476
Product_	20678KB	10	44167
Product_	10886KB	1	39299
Product_	6534KB	3	20132
Product_	5258KB	17	11094
Product_	3978KB	26	9409
Vendor	2826KB	18	6865
Time_SYS Date	2502KB	4	4749
Time_Date	2310KB	4	5414
Product_	710KB	9	1302
Product_	710KB	10	1270
Location	330KB	22	528
	330KB	17	444

- Model split by business unit (10 major BUs)

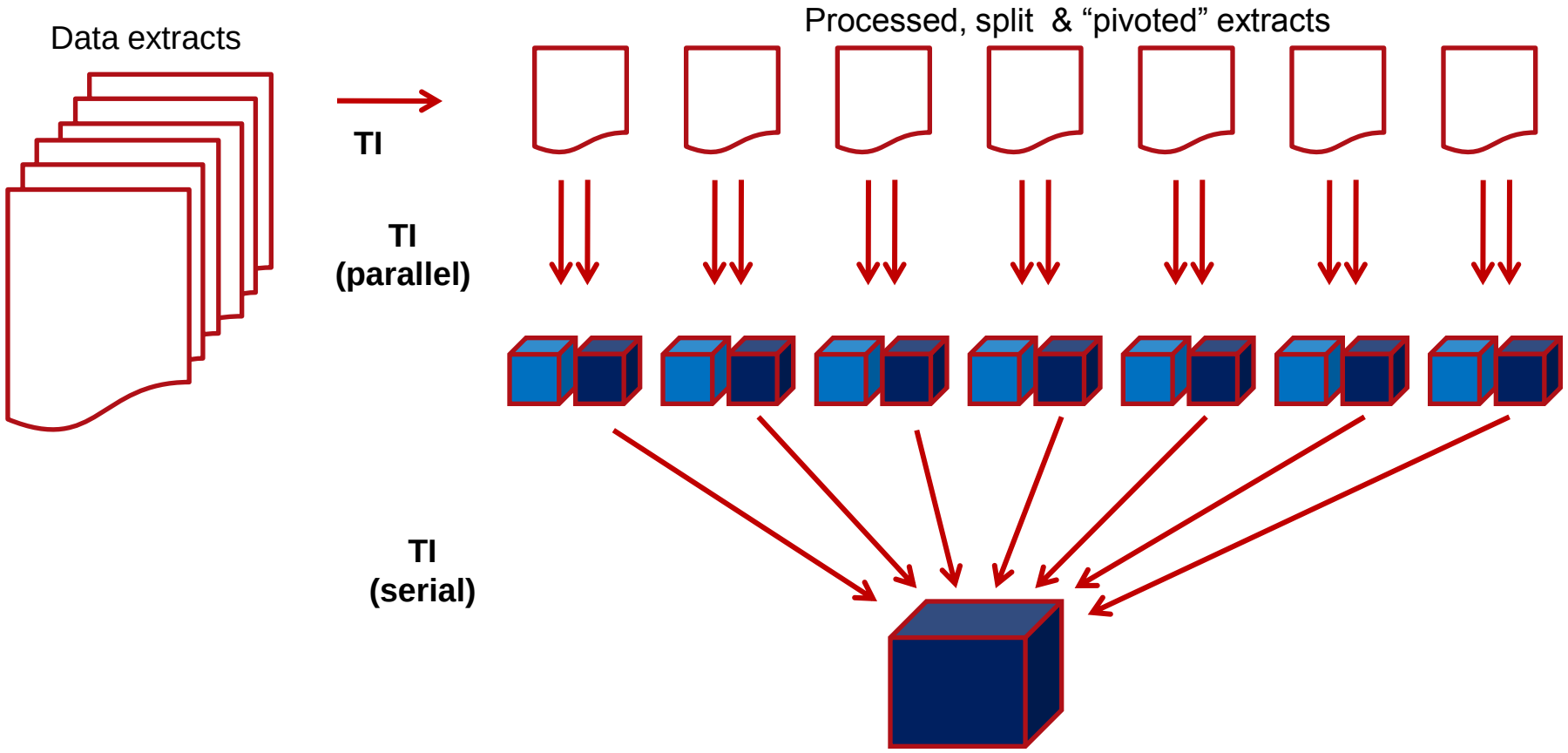


- Majority of reports run from summary cube at class level
- Final drill down to product level only available within business unit cubes

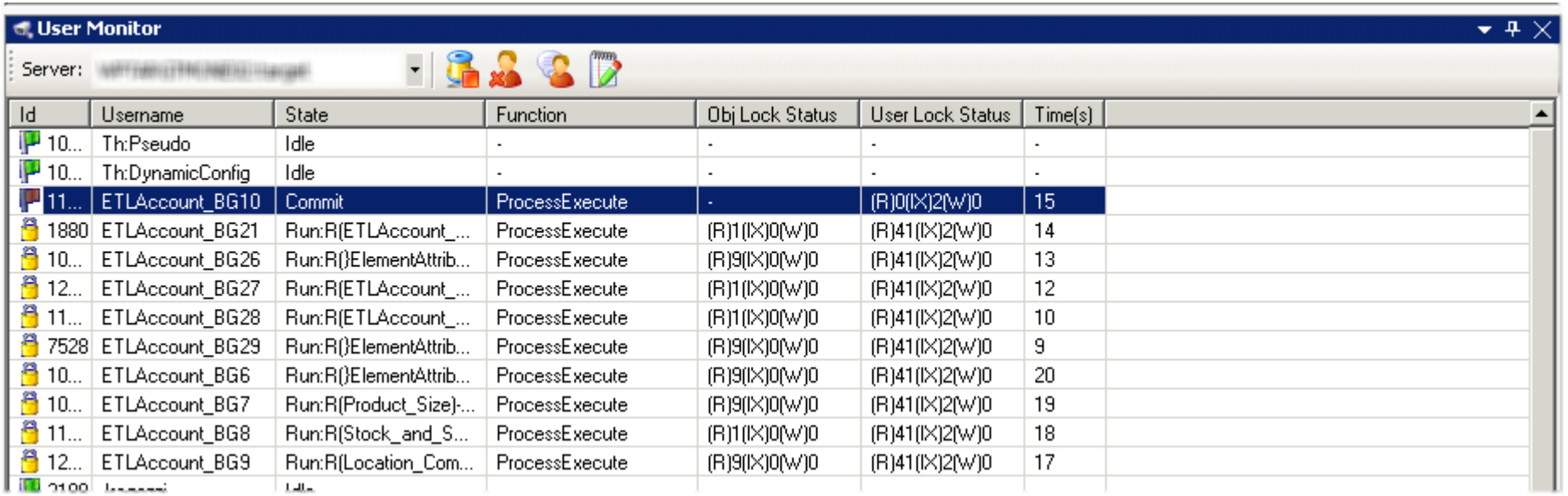
- The end user perspective ...



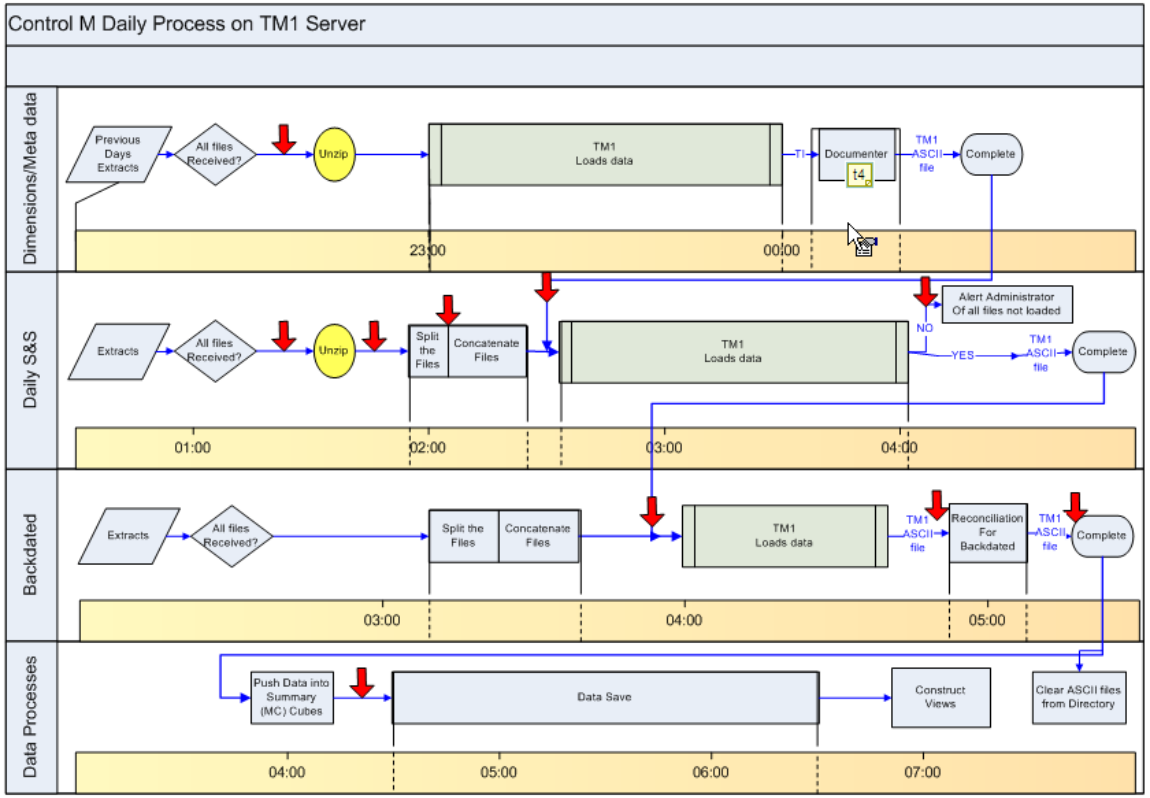
- Data integration perspective



- Model partitioning
- Multi-threaded data loads
- External job scheduling
- Feederless rules
- Avoiding locking
- Eliminating public dynamic subsets

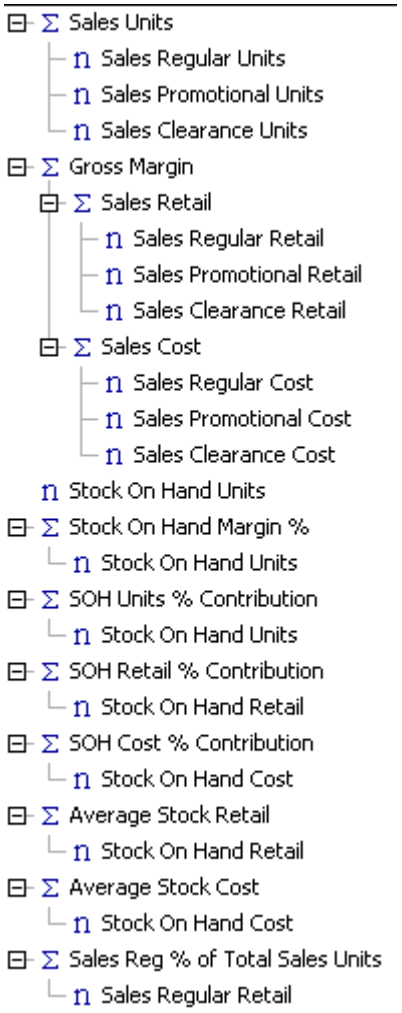


- External data loads from file extracts
  - Eliminates ODBC or source system as a bottleneck in TM1 processing
- Made possible by model partitioning
- Each cube loads on a different thread



- Control M is the enterprise scheduling tool
- When file transfer of extracts are complete TM1 jobs are called via a 3rd party command line tool TM1ProcessExecute.exe
- When TM1 jobs complete a confirmation file is placed in a monitored directory to signal downstream jobs can commence

- Allows much better and finer control than can be achieved via the TM1 chore scheduler

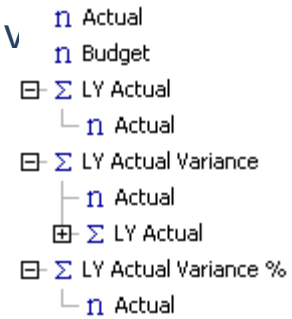


- Q: what are “feederless” rules?
- A: [illegible]ding!

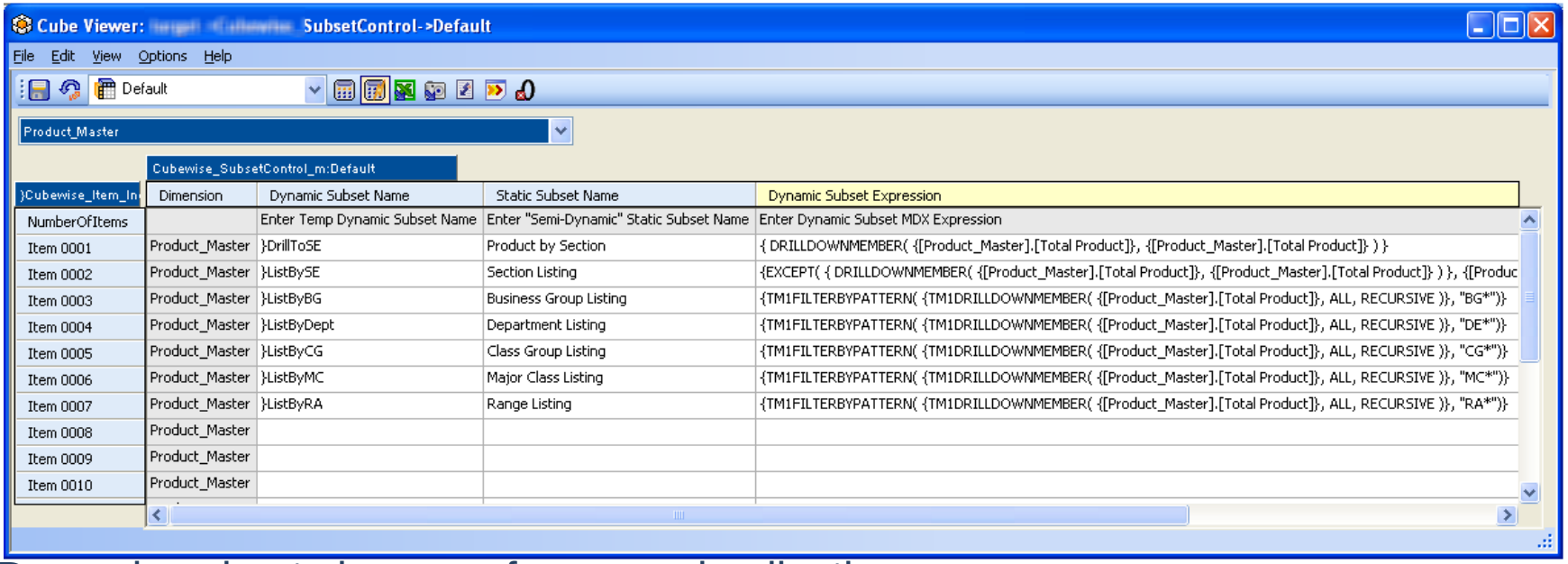
Regular hierarchies or rollups.
Elements consolidate or add/subtract

- [illegible]asures are defined as
- [illegible]ata to feed the rule rather than feeders
- Significant savings in model size and load time
- Example rule:
  [‘LY Actual’] = C: DB(‘Cube’, ‘Actual’, AttrS(‘Year’, !Year, ‘Prev Year’), ...);
- Ca[illegible]imensions (e.g. v

Artificial rollups for what would usually be N elements. The children are defined for the sole purpose of “feeding” the rule calculation



- To take advantage of multi-threaded loading need to make sure that each processes only obtains write locks on a unique set of objects
- Concept of “semaphores” or signalling cubes
  - So what’s a semaphore?
  - Any common object that needs to get written to. Most common example would be }CubeProperties cube with CubeSetLogChanges
- How to avoid
  - Handle logging before and after multi-threaded phase not in each process
  - Avoid multiple processes attempting to write to the same object (subsets, views, not just data)
  - Custom logging cubes: write out to file and process later
- Sometimes you want to lock ...
  - E.g. To force queuing to ensure one process has a “clear window”



- Dynamic subsets have performance implications
- Concept of “semi-dynamic” subsets
  - Nightly regeneration of dynamic subsets and conversion to static subsets
  - Get (most) benefits of dynamic subsets without the drawbacks

Thank You!
Your Feedback is Important to Us

- Access your personal session survey list and complete via SmartSite
  - Your smart phone or web browser at: ***iodsmartsite.com***
  - Any SmartSite kiosk onsite
  - Each completed session survey increases your chance to win an Apple iPod Touch with daily drawing sponsored by Alliance Tech

54