

IBM Planning Analytics & IBM Cognos TM1:

Introductory Guidelines and Proven practices for TM1/PA
Configuration, Architecture & Design

Or:

An introductory architecture and design guide for building
very large yet fast performing TM1/PA models

Andreas Kugelmeier
Executive Consultant, FOPM
Planning Analytics Architect
IBM Data and AI Expert Labs
Mobile Phone: +1-215-384-7302
Email: kugelmeier@us.ibm.com

Notices & Disclaimers

Copyright © 2019 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations and papers (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS document is distributed "AS IS" without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environment. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Bluemix Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.



Document Version History

Date	Version	Author	Description
02/25/2014	1.0	Andreas Kugelmeier	Version 1.0
05/28/2014	1.01	Andreas Kugelmeier	Minor changes to section 'Use Parallel Data Load for re-loading data for very large cubes'
06/25/2014	1.02	Andreas Kugelmeier	Minor additions to section on MTQ configuration
05/20/2015	1.1	Andreas Kugelmeier	<ul style="list-style-type: none"> Modify Section 'Use Parallel Data Load for re-loading data for very large cubes' Modify Section on TM1 with Cognos BI Misc. other modifications
06/11/2015	1.2	Andreas Kugelmeier	<ul style="list-style-type: none"> Improvements to MTQ section Additions to section on MTQ: additional notes on MTQ configuration parameters Minor additions to section on VMM configuration Misc. edits & improvements
07/23/2015	1.21	Andreas Kugelmeier	<ul style="list-style-type: none"> Add paragraph on MTQ parameter MTQ.OperationProgressCheckSkipLoopSize
07/24/2015	1.22	Andreas Kugelmeier	<ul style="list-style-type: none"> Additional information on caveats when using MTQ to speed up View generation via ViewConstruct
07/28/2015	1.3	Andreas Kugelmeier	<p>Add recommendation on</p> <ul style="list-style-type: none"> when to MTQ logging (only enable when needed) Persistent Feeders <p>Corrected/Improved section on MTQ Behaviour & Configuration Options as well as Recommendations on MTQ Configuration based on latest tests</p>
07/29/2015	1.31	Andreas Kugelmeier	Add recommendation to increase/re-optimize VMM when employing MTQ
07/30/2015	1.32	Andreas Kugelmeier	Corrected VMM default as per http://www-01.ibm.com/support/docview.wss?uid=swg27024354
08/05/2015	1.33	Andreas Kugelmeier	<p>Add material on</p> <ul style="list-style-type: none"> Guidelines & Samples for Rules vs. TI Dimension Sort Order Optimization ODBC vs. File-based load
08/12/2015	1.4	Andreas Kugelmeier	<ul style="list-style-type: none"> Update TI-Synchronization Utility zip file Add copy on use of Alias names in rules and feeders Misc. edits to section on Cognos BI Misc. edits to section on Misc. Recommendations and Guidelines Misc. edits to section on Dimension Sort Order Optimization Add section on Security
8/25/2015	1.41	Andreas Kugelmeier	<ul style="list-style-type: none"> Misc. minor updates
9/17/2015	1.42	Andreas Kugelmeier	<ul style="list-style-type: none"> Misc. minor edits to Section on Cognos BI & MTQ
10/2/2015	1.43	Andreas Kugelmeier	<ul style="list-style-type: none"> Correct tm1 server data log (not tm1server.log, but tm1s.log)
11/20/2015	1.44	Andreas Kugelmeier	Change title
1/7/2016	1.45	Andreas Kugelmeier	<ul style="list-style-type: none"> Add section 5.6 on MDX & Rowset logging
1/26/2016	1.46	Andreas Kugelmeier	<ul style="list-style-type: none"> Addtl. information in section Rules vs TI Fix typo: "If <u>rules</u> [not TI] are used (in ElementSecurity etc.), a security metadata change - for example due to a hierarchy change (with corresponding/resulting security changes for parent and/or child nodes) or due to a new element being added to a hierarchy (like a new archived version for which READ access now to be granted to all applicable groups) - will always require running the 'SecurityRefresh()' command in TM1"
2/2/2016	1.47	Andreas Kugelmeier	Add section on ViewConsolidationOptimizationMethod=TREE
3/18/2016	1.48	Andreas Kugelmeier	<ul style="list-style-type: none"> Additional information on Enabling Hypertreading where applicable Add section on Setting Windows Power Plan to High Performance
3/28/2016	1.49	Andreas Kugelmeier	<ul style="list-style-type: none"> Correction to section on VMM
4/23/2016	1.50	Andreas Kugelmeier	Edits to section on Dimension Sort Order Optimization
8/3/2016	1.51	Andreas Kugelmeier	Additions to section 'Optimization of Cube Logging and maintenance of TM1 log Files'
8/11/2016	1.52	Andreas Kugelmeier	Add commentary/footnote on the necessity to reprocess feeders for cubes cleared using CubeClearData
8/31/2016	1.53	Andreas Kugelmeier	Additions to section on VMM and VMT configuration
9/23/2016	1.55	Andreas Kugelmeier	Additions to section on VMM configuration (VMM defaults w/ Cognos BI, Max VMM value)
10/17/2016	1.60	Andreas Kugelmeier	Minor edits and more detailed recommendations on when to use DimensionElementInsertDirect and DimensionElementComponentAddDirect functions
10/20/2016	1.61	Andreas Kugelmeier	Addl. information on leveraging MTQ TI-processing of views
10/27/2016	1.65	Andreas Kugelmeier	Misc. additions to section on Security
11/4/2016	1.7	Andreas Kugelmeier	Addl. information on leveraging MTQ TI-processing of views
11/18/2016	1.75	Andreas Kugelmeier	Misc. minor edits and additions
1/9/2017	1.8	Andreas Kugelmeier	Remove section on TM1 with Cognos BI (now available in a separate paper, called



			'Guidelines & Proven Practices for IBM Planning Analytics & IBM Cognos TM1: Integration & Performance Optimization with Cognos BI'
01/03/2019	2.0	Andreas Kugelmeier	Misc. corrections and additions to MTQ section
02/01/2019	2.1	Andreas Kugelmeier	Add recommendation on PersistentFeeders with PA
02/27/2019	2.2	Andreas Kugelmeier	Add Caveat re adequate HW sizing with MTQ=ALL
04/04/2019	2.3	Andreas Kugelmeier	Misc. minor edits
11/21/2019	2.46	Andreas Kugelmeier	Addtl. Information re ForceReevaluationOffFeedersForFedCellsOnDataChange Misc. edits
05/11/2020	2.47	Andreas Kugelmeier	Minor updates to section on TM1s.cfg

Table of Contents

1. About this Document	7
2. Configuration Recommendations & Guidelines	8
2.1 The TM1s.cfg in IBM Planning Analytics: General Recommendations	8
2.2 Leverage Multi-Threaded Queries	9
2.3 MTQ Configuration	9
2.3.1 MTQ Behavior & Configuration Options	9
2.3.2 MTQ Configuration Scenarios, Considerations & Practices	10
2.3.3 Advanced MTQ configuration	13
2.4 MTQ Logging and Monitoring	14
2.4.1 MTQ Monitoring	14
2.4.2 MTQ Logging	14
2.5 Caching of MTQ results	16
2.6 Use of MTQ for TI-processing	17
2.7 Configure TM1 Parallel Interaction	19
2.8 Improving TM1 DB Startup time	19
2.8.1 Configure MaximumCubeLoadThreads (TM1 10.2.2)	19
2.8.2 Leverage MTCubeLoad, MTFeeders, and MTFeeders.AtStartup (TM1 11 / PA)	19
2.8.3 PersistentFeeders	19
2.9 Optimize Server handling of MDX Subsets	20
2.10 Set ViewConsolidationOptimizationMethod=Tree	20
2.11 Optimize the use of the TM1 Query cache via configuration of the VMM parameter	21
2.11.1 Query Cache configuration via VMM & VMT: Introduction	21
2.11.2 Configuration use cases & recommendations for VMM Query Cache configuration	21
2.11.3 Configuration use cases & recommendations for VMT Query Cache configuration	22
2.12 Optimization of Cube Logging and maintenance of TM1 log Files	23
3. General Design Recommendations	24
3.1 Recommendations on dimensionality	24
3.2 Recommendations on calculations via Rules	24
3.3 Rules vs. TI	26
3.3.1 Overview	26
3.3.2 Guidelines	26
3.3.3 Sample Scenarios	27
3.4 Recommendations on cube dimension order	28
3.4.1 Affect of Dimension Order on Performance	28
3.4.2 Initial Dimension Order	28
3.4.3 Optimizing dimension sort order	28

3.5	Security	31
3.5.1	Performance Considerations in TM1 Security	31
3.5.2	Avoid locking due to Security Changes	31
3.5.3	Optimize Security Model Performance (Query-Overhead due to Security)	32
3.6	TI-process Design	33
3.6.1	Employ a modular TI-process design methodology	33
4.	Recommendations for Data Load/Update procedures	34
4.1	Load data using TM1 ODBC interface	34
4.2	Optimizing Performance	34
4.2.1	Use CubeClearData() command for complete re-load of cube data	34
4.2.2	Separate TI processes for dimension maintenance and data load	34
4.3	Use TM1 functions DimensionElementInsertDirect and DimensionElementComponentAddDirect	35
4.4	Use Parallel Data Load for re-loading data for very large cubes	35
4.5	Minimizing Contention	36
4.5.1	Processing contention	36
4.5.2	Saving of Cube Data	37
4.5.3	Subsets & Views	37
4.5.4	Synchronization & Serialization of applicable TI processes	38
5.	Miscellaneous Recommendations and Guidelines	39

1. About this Document

This document is geared at introducing configurations, proven design practices & guidelines for

- TM1 Server and DB Configuration &
- TM1 Model Architecture/Design

The paper will provide the reader with the knowledge to

- significantly improve TM1 query processing speed,
- process data faster,
- avoid input, query & processing contention,
- build models that are not only faster & more scalable, but simpler and more user-friendly,
- have TM1 utilize HW more efficiently.

2. Configuration Recommendations & Guidelines

2.1 The TM1s.cfg in IBM Planning Analytics: General Recommendations

	Tm1s.cfg parameter	explanation
Recommended non-default parameters:	AllowSeparateNandCRules=T ViewConsolidationOptimizationMethod =Tree LoadPrivateSubsetsOnStartup=T ReduceCubeLockingOnDimensionUpdate=T PersistentFeeders=F vs. T	<ul style="list-style-type: none"> • Default is F. It is the proven practice to separate N- and C-level rules • Array is default (legacy). Tree is typically faster • Default is F. T will improve performance for end users with private subsets • Default is F. T will significantly improve scalability and reduce contention • Default is T, consider/test setting to F and using MTFeeders (see below). For new TM1 databases, set to F and only turn on if/when needed.
Optional recommended non-default parameters	MTCubeLoad=T MTFeeders=T MTFeeders.AtStartup=T MTCubeLoad.UseBookmarkFiles =T IndexStoreDirectory=<IndexStoreDirectory>	<ul style="list-style-type: none"> • Default is F. Allows TM1 to use multiple threads/cores for Cube Load on Startup. Will likely allow for faster database startup. Click here for more information • Default is F. Allows TM1 to use multiple threads/cores for feeder processing. Will likely result in significantly faster 'refresh' & startup of database instances. Click here and here for more information. Note: use of MTFeeders functions is typically OK with conditional feeders that are based on attributes or regular lookup cubes. • The use of Bookmarks/Bookmark files can speed up startup time significantly. It is recommended to compare startup time with or without use of Bookmarkfiles (after bookmarkfiles were created)
New default to be handled with Caution	MTQQuery=T (default) vs. MTQQuery=F	<ul style="list-style-type: none"> • Default=T. MTQQuery will use MTQ to query and 'pre'-cache C-level views used by TI-processes. While this typically leads to performance improvements, it can also significantly increase RAM consumption for larger views. That is because TI with MTQQuery=F / without MTQQuery will not load a view into RAM, but instead 'cycle' through a view in smaller blocks, not really increasing RAM usage much. Yet with MTQQuery=T, TM1 will attempt to query and load the entire C-level view into RAM prior to processing it via TI <ul style="list-style-type: none"> ⇒ MTQQuery=T (default) can increase RAM use significantly ⇒ only use IF (i) size of views is managed (known to be small enough) and (ii) sufficient RAM is available, otherwise, use MTQQuery=F; with MTQQuery=T, use DisableMTQViewConstruct () for Tis for which you want to disable MTQQuery
Other	ForceReevaluationOffFeedersForFedCellsOnDataChange → Leave at F (default), i.e. do not set to T	<ul style="list-style-type: none"> • Using ForceReevaluationOffFeedersForFedCellsOnDataChange=T can very significantly degrade TI processing and input contribution performance. • Only use ForceReevaluationOffFeedersForFedCellsOnDataChange=T if you have conditional feeders that change based on regular data inputs <u>and</u> if the continuous re-evaluation is needed. Otherwise, do not use this parameter. • Enabling this parameter is not necessary even for use of TM1 Applications/Contributor. The parameter is only needed if TM1 Performance Modeler is used for cube rule modeling (it is not recommended to use TM1 Performance Modeler for cube rule modeling or TI-process modeling).

Note: Temporary parameters (such as parameter changes prompted by support) should be commented as such and should include an explanation of why they were added (logical reason, not just 'as told by IBM support').

2.2 Leverage Multi-Threaded Queries

As of TM1 version 10.2, a new feature known as “multi-threaded queries”, or MTQ, is available. MTQ allows TM1 to take advantage of multiple processor cores to significantly increase the speed at which TM1 can perform calculations and queries by allowing queries to split into multiple processing threads, effectively using a parallel processing regime to resolve queries significantly faster.

The performance improvement of Multi-Threaded-Queries is in an approximately linear relationship to the # of CPU cores that the TM1 query engine is allowed to utilize: Approximate TM1 10.2 Query Time = TM1 <10.2 Query Time / # of CPU Cores utilized for Multi-Threaded Queries.

Note that for smaller TM1 models with high rule complexities, MTQ may not result in significant performance gains => it is not a good idea to count on MTQ to ‘offset’ the effects of poor rules or cube designs.

2.3 MTQ Configuration

2.3.1 MTQ Behavior & Configuration Options

- MTQ is configured and enabled via corresponding entries in the tm1s.cfg file (MTQ=N, with N=# of processing Threads)
- The MTQ parameters in the tm1s.cfg are dynamic, i.e. the TM1 Server process does not have to be restarted after a change in the MTQ settings.
- The MTQ value specifies the total # of threads that TM1 can leverage for MTQ.
- If MTQ=1 or MTQ=0, MTQ is disabled
- TM1 10.2: If there is no MTQ entry in the tm1s.cfg file, MTQ is disabled.
- TM1 11 (Planning Analytics): If there is no MTQ entry in the tm1s.cfg file MTQ is set to the # of processors on the machine (MTQ=ALL)
- To set the value to the maximum number of cores available on a server, the setting MTQ=ALL can be used.
- Setting MTQ to a negative number will (MTQ=-N) will result in the # of MTQ threads being determined as follows: $T=M-N+1$ (where T= # of threads to be used by MTQ, M= # of CPU cores available to TM1). For example, if your computer has 64 cores and you set MTQ=-10, the # of MTQ Cores will be $T = 64-10+1 = 55$
- **The MTQ value (MTQ=X, like MTQ=4) is the total thread pool for all users that the TM1 Server makes available for Multi-Threaded-Queries.**
 - ⇒ Example: If you set MTQ=4 on an 8-CPU machine, only 4 threads will be used in Total for MTQ. If two users run a query that leverages 4 MTQ cores, one user will get 1 non-MTQ thread, and the other user will get 1 non-MTQ Thread plus 4 MTQ Worker Threads.
 - ⇒ MTQ is not the MTQ pool per user, but for the TM1 Database server in total.
- An MTQ pool thread becomes ‘available’ to other users once it has finished.
 - ⇒ Example: If a query initially gets assigned 4 MTQ worker threads, this does not mean it will keep all those threads. One thread may finish and the query will continue with 3 worker threads. Or 2 threads will finish, but the TM1 MTQ engine will determine that the next ‘part’ of the query is best served with an additional 8 threads, resulting in 10 threads being used...

- ⇒ As a particular MTQ worker threads is finished (right after 'commit' in TM1Top), the thread becomes available again, and it may be used by the same query (for a new worker thread), or by a different user query.
- MTQ very effectively leverages Hyper-Threaded Cores. If the CPU and OS supports Hyper-Threading, we strongly recommend to enable Hyper-Threading (some hardware requires Hyper-Threading to be enabled via a Bios setting) and include the hyper-threaded cores in the MTQ configuration consideration.
- **MTQ is NOT dependent/reliant on the # of CPUs on the machine!**
 - ⇒ **MTQ will leverage multi-core processing capabilities by splitting and resolving queries via parallel worker threads**
 - ⇒ If you set MTQ to ALL or leave it at the default (ALL), TM1 will set an MTQ thread pool equal to the number of cores on the machine.
 - ⇒ You can set MTQ to a value higher than the number of cores on the machine. You will see more worker threads than CPU cores. This is acceptable, even though on most hardware, it is typically not going to result in performance gains, yet at lower concurrency typically will also not lead to performance degradation.
- The CPU time and resources given to a MTQ worker thread are entirely handled by the Operating System.
- An MTQ worker thread will ONLY occupy an entire CPU core if the core is not busy otherwise.
- TM1 simply initiates a parallel thread. Where and how this thread is being processed is handled by the Operating System.

2.3.2 MTQ Configuration Scenarios, Considerations & Practices

- a) Generally, the best practice is to set the MTQ value such that the maximum available processor cores are used, i.e. MTQ=All or MTQ=-1 or MTQ=M (with M= # CPU cores incl. hyper-threading cores).
- ⇒ it is typically best to leave MTQ at its MTQ=ALL (default for PA) and hence have it leverage as many worker threads as there are CPU cores. This ensures that hardware is leveraged at its maximum.
 - ⇒ If you have two or more TM1 databases on the same machine, it is still recommended to set MTQ to ALL, because: An MTQ worker thread does not occupy a CPU core. MTQ worker threads will share CPU cores where needed and where applicable. It is the Operating System that handles CPU time and resources.
 - ⇒ If you have two databases on an 8 core machine, each set to MTQ=8, and on each a user runs a query leveraging 8 MTQ threads, the 16 MTQ threads are balanced among the 8 cores on the machine, utilizing the HW in an optimal way.
 - ⇒ If only one user runs a query leveraging 8 MTQ threads, the HW is still used in an optimal way.
 - ⇒ But if you set MTQ=4 and just one user runs a query, the HW is utilized at only 50%.
 - ⇒ If we extend these simple examples to many users and dozens to hundreds of threads (like in a typical, large TM1 environment), we can argue that every TM1 database should be configured to run queries as efficiently as possible, leveraging the available HW as much as possible.
 - ⇒ Balancing HW utilization is the job of the Operating System

Caveat: While MTQ=ALL ensures that hardware is leveraged at its maximum, it is only a recommended setting provided that the HW-sizing is adequately matched with the demands of the TM1 Database(s) that are running on the machine. Operating TM1 databases with MTQ=ALL yet on machines with insufficient HW capabilities may over-tax the HW and lead to performance degradation.

- b) On very powerful HW (high # of CPU cores) on smaller to mid-sized databases, it may not make much of a difference to set MTQ to a value that is lower than the total number of cores. I.e. on a 36 core machine, an MTQ=32 or MTQ=36 may not make not much of a difference if the data volume is not sufficiently high to allow performance gains via additional processing threads.
- c) On very large Databases, a high MTQ # typically does make a big difference
- ⇒ It is a good practice to start with MTQ=ALL (= default).
 - ⇒ To evaluate if different MTQ settings improve performance, adopt a holistic testing approach:
 - **MTQ Worker Thread ≠ CPU Core** => An MTQ Worker Thread is 'just' a TM1 Thread.
 - What is the peak load on each database?
 - What is the Avg CPU utilization at Peak?
 - Do we currently max-out our HW capabilities? (HW should be used as much as possible; HW idle time is not good, because it means 'things' could be running faster)
 - How do the # of cores affect our end-user queries? This depends on the database/cubes: are they very large, do they leverage rules, ...
 - ⇒ Analyze!
 - ⇒ Is there an MTQ setting where using additional worker threads do not matter much anymore? If Yes, then this should be your max MTQ setting (i.e. do not go higher than where you see a tangible difference)
- d) Some additional considerations when evaluating MTQ Settings:
- One could argue that setting MTQ to a slightly lower value than the # of cores will allow more CPU time to be available for TI-processing (for example). **But:** This is **only** true IF the total concurrency on the database(s) is sufficiently low (compared to the # of cores). Example: The Operating System is balancing 200 concurrent TM1 users (and their TM1 threads) among its 36 cores. In such a context, setting MTQ to 36 or 34 or 30 will not really make a difference, because the CPU will be operating at maximum capacity regardless of the MTQ setting.
 - ⇒ When considering to lower MTQ, take into account the overall concurrency on the database at peak time and the data volume that you are querying.
 - ⇒ If concurrency is high and the data volume high, a higher MTQ value typically results in better performance in that end-user queries will be faster.
 - ⇒ When you operate multiple Databases on one server and each database has a high concurrency, lowering MTQ will not necessarily free up CPU time for processing etc., because the overall load on the environment is already high.
 - ⇒ Lowering MTQ in environments where multiple TM1 Database Instances share the same hardware will only have a positive impact on performance if the concurrency on each database is sufficiently low such that HW resources are kept available for other TM1 databases (and if those other databases need the free CPU time).
 - ⇒ When setting MTQ, do not optimize it for low concurrency times. Optimize it for when the database is used most.
- e) By employing a 'give and take' approach to assigning and re-assigning processing units, TM1 will balance and re-balance MTQ thread pool utilization between query requests: if all MTQ pool threads are available and a particular query can leverage all threads, TM1 may assign the max # of worker threads available and hence start MTQ processing with all threads. If additional queries are run during that time, hence 'requesting' query time, and once one or more MTQ worker threads have finished (committed), TM1 will assign the now free threads to other queries.

- f) Keep in mind that you can change MTQ settings while the server is running and while users are running queries. The MTQ cores will be re-balanced according to your new MTQ configuration.
- g) The use of MTQ will increase memory (RAM) usage. Subsequently, the use of MTQ typically will require an increase in VMM size. If VMM cache is set too low, even queries that were cached without MTQ use may not be cached anymore once MTQ is enabled. In such cases – to avoid unnecessary re-execution of MTQs - increase the VMM value until repeated query execution will not trigger MTQ activity anymore (indicating the cache is used). See <Caching of MTQ results> below.

2.3.3 Advanced MTQ configuration

- Per [IBM support flash](#), Customers who are applying MTQ to models with reasonably complex rules (i.e. to rules that have cross-cube references with recursiveness depth more than 2) who are on an early TM1 10.2 release (prior to 10.2 FP1) could intermittently suffer from incorrect calculations unless they have disabled MTQ for single cell consolidation via parameter `MTQ.SingleCellConsolidation=false`. This parameter should only be used in such Pre 10.2 FP1 environments, once an upgrade to the fixpack or higher version has occurred, the parameter should be removed.
- The dynamic TM1s.cfg parameter `MTQ.MultithreadStargateCreationUsesMerge=TRUE` can speed up the creation of large stargate views of >100MB. The default value of this parameter in TM1 10.2.2 is FALSE.
- The dynamic TM1s.cfg parameter `MTQ.CTreeWorkUnitMerge=TRUE` speeds up concurrent population of calculation cache kept in the CTree, thereby improving performance of queries that cause a very large cache (re-)population. The default value of this parameter in TM1 10.2.2 is FALSE. Note that `MTQ.CTreeWorkUnitMerge=TRUE` could lead to redundant work in MTQ threads when the same calculation for exactly the same cell is re-computed per MTQ thread that needs it, whereas with `MTQ.CTreeWorkUnitMerge=FALSE` a computed cell would be published faster into a global CTree cache and then re-used by all MTQ threads. In cases where `MTQ.CTreeWorkUnitMerge` is enabled (= set to TRUE), the additional parameter `MTQ.CTreeRedundancyReducer=TRUE` (also a dynamic parameter) may be used to reduce query redundancies if applicable.
- In TM1 10.2 Version prior to TM1 10.2.2. FP2 HF9, the (default) `MTQ.CTreeWorkUnitMerge=FALSE` setting could lead to rules not being calculated in certain scenarios where TI-processes were used to (re-)populate/refresh data. We therefore recommend to upgrade to TM1 10.2.2 FP2 HF9 or higher (10.2.2 FP3 and 10.3) when using MTQ. If an upgrade is not feasible at the given time and issues with rule calculations are encountered after TI-processing, enabling `MTQ.CTreeWorkUnitMerge` (`MTQ.CTreeWorkUnitMerge=TRUE`) will also solve the issue.
- The dynamic TM1s.cfg parameter `MTQ.OperationProgressCheckSkipLoopSize=<N>` specifies the number of cells to be processed before checking whether multi-threaded splits are needed. The default value is 10000. Is overridden by `MTQ.ImmediateCheckForSplit=T` which will effectively set `MTQ.OperationProgressCheckSkipLoopSize` to 1.

2.4 MTQ Logging and Monitoring

2.4.1 MTQ Monitoring

MTQ activity is best monitored via use of the TM1 Operations Console. The parent thread and each MTQ worker thread will all be visible as separate threads in TM1 Operations Console.

In the following screenshot, thread ID 14744 ran a query (a view) that subsequently spawned 8 worker unit threads:

ID	User	Context	State	Function	Type	Object	Info	Time (s)
6840	Th:Pseudo	-	Idle	-	-	-	-	0
6112	Th:Dynamic Config	-	Idle	-	-	-	-	0
7484	Admin	PM Hub	Idle	-	-	-	-	0
14744	Admin	Architect	Run:R	Get/viewByHandle	Rule	FX	-	57
13260	> Work unit for 14744	-	-	-	-	-	0	
3136	> Work unit for 14744	-	-	-	-	-	0	
13656	> Work unit for 14744	-	-	-	-	-	0	
13748	> Work unit for 14744	-	-	-	-	-	0	
13632	> Work unit for 14744	-	-	-	-	-	0	
12816	> Work unit for 14744	-	-	-	-	-	0	

2.4.2 MTQ Logging

To generate logging information on multi-threaded queries, the following entries can be made in the tm1s-log.properties file (located in the same location as your tm1s.cfg file):

To generate logging information on multi-threaded queries, the following entries can be made in the tm1s-log.properties file (located in the same location as your tm1s.cfg file):

- To capture Stargate creation times: log4j.logger.TM1.Cube.Stargate=DEBUG

=> look for tm1server.log entries like

```

10148 [3] DEBUG 2015-07-27 20:42:05.666 TM1.Cube.Stargate.Reference Stargate 0x00000000E6967210 referenced : RefCount=1->2
10148 [3] DEBUG 2015-07-27 20:42:05.666 TM1.Cube.Stargate.Reference Stargate 0x00000000E6967210 released : RefCount=2->1
10148 [3] DEBUG 2015-07-27 20:42:05.689 TM1.Cube.Stargate.Catalog Adding stargate: 0x00000000E6969410 to catalog: 0x00000000E004BF58 of cube:
0x0000000006993010.
10148 [3] DEBUG 2015-07-27 20:42:05.689 TM1.Cube.Stargate.Reference Stargate 0x00000000E6969410 referenced : RefCount=0->1
10148 [3] DEBUG 2015-07-27 20:42:05.689 TM1.Cube.Stargate.Reference Stargate 0x00000000E6969410 referenced : RefCount=1->2
10148 [3] DEBUG 2015-07-27 20:42:29.298 TM1.Cube.Stargate dbstgCreate: dbstgCalculate (0x00000000E6969410) in 23609ms SGSize=24701Kb
workPoolSize=0Kb hPoolSize=128Kb.
10148 [3] DEBUG 2015-07-27 20:42:29.299 TM1.Cube.Stargate.Definition Axes Consolidation Subsets:
Dimension: <Dimension Elements as on columns and rows>
10148 [3] DEBUG 2015-07-27 20:42:30.059 TM1.Cube.Stargate dbstgCreate: dbstgCalculateConsolidationLevel_UseTree (0x00000000E6969410) in 760ms
SGSize=24701Kb workPoolSize=1055Kb hPoolSize=128Kb.
10148 [3] DEBUG 2015-07-27 20:42:30.063 TM1.Cube.Stargate dbstgCreate: dbstgStoreConsolidationsBack_UseTree (0x00000000E6969410) in 4ms
SGSize=24701Kb workPoolSize=1055Kb hPoolSize=128Kb.
10148 [3] DEBUG 2015-07-27 20:42:30.063 TM1.Cube.Stargate dbstgCreate: CalculateAxesConsolidations Stargate (0x00000000E6969410) in 765ms
10148 [3] DEBUG 2015-07-27 20:42:30.063 TM1.Cube.Stargate New Stargate Created (0x00000000E6969410) in 24374ms SGSize=24701Kb workPoolSize=0Kb
hPoolSize=128Kb.
10148 [3] DEBUG 2015-07-27 20:42:30.063 TM1.Cube.Stargate.Reference Stargate 0x00000000E6969410 released : RefCount=2->1
10148 [3] DEBUG 2015-07-27 20:42:30.357 TM1.Cube.Stargate.Reference Stargate 0x00000000E6967210 released : RefCount=1->0
10148 [3] DEBUG 2015-07-27 20:42:30.357 TM1.Cube.Stargate Destroying Stargate 0x00000000E6967210





```

- To capture work unit splitting: log4j.logger.TM1.Parallel=DEBUG

=> look for tm1server.log entries like

```
10148 [3] DEBUG 2015-07-27 20:30:52.187 TM1.Parallel.Splits ...
10148 [3] DEBUG 2015-07-27 20:35:20.175 TM1.Parallel.SplitPlan Wrote split plan for StgConsOP[<Cube>:( [ <DimensionA>].[ <DimensionAElement>],
[ <DimensionB>].[ <DimensionBElement>],...)] to file '<LogFileDirectory>\<jsonfilename>.json'
10148 [3] DEBUG 2015-07-27 20:35:20.178 TM1.Parallel.Operation ...OP:... TM1ParallelOperation::MergeWorkUnits in 3ms
```

(see the json files for additional, very detailed debugging information on the split plans, for example:

-  splitPlan_StgConsOP_0x0000000008CAC690.json
-  splitPlan_StgCalcOP_0x0000000008AC59E0.json
-  splitPlan_StgConsOP_0x0000000008AB6AE0.json
-  splitPlan_StgCalcOP_0x00000000064F4210.json

- To capture the event of operation threads picking work units: log4j.logger.TM1.OperationThread=DEBUG

=> look for tm1server.log entries like

```
13064 [] DEBUG 2015-07-27 21:08:53.279 TM1.OperationThread Executing work unit:StgBuildWU:0x0000000008C318D0 [1/(52):(80)]OP:0x0000000006626CD0
3360 [] DEBUG 2015-07-27 21:08:53.279 TM1.OperationThread Executing work unit:StgBuildWU:0x0000000008C2F9C0 [1/(17):(36)]OP:0x0000000006626CD0
14636 [] DEBUG 2015-07-27 21:08:53.279 TM1.OperationThread Executing work unit:StgBuildWU:0x0000000008C356F0 [1/(37):(51)]OP:0x0000000006626CD0
13064 [] DEBUG 2015-07-27 21:09:32.580 TM1.OperationThread Finished execution of work unit:StgBuildWU:0x0000000008C318D0 [1/(52):(80)]OP:0x0000000006626CD0
13064 [] DEBUG 2015-07-27 21:09:32.601 TM1.OperationThread Executing work unit:StgBuildWU:0x0000000008C318D0 [1/(33):(36)]OP:0x0000000006626CD0
14636 [] DEBUG 2015-07-27 21:11:46.369 TM1.OperationThread Finished execution of work unit:StgBuildWU:0x0000000008C356F0 [1/(37):(51)]OP:0x0000000006626CD0
14636 [] DEBUG 2015-07-27 21:11:46.373 TM1.OperationThread Executing work unit:StgBuildWU:0x0000000008C356F0 [1/(14):(16)]OP:0x0000000006626CD0
```

Note that those logging flags typically are only useful for advanced debugging & troubleshooting. We recommend to only enable MTQ logging temporarily when/if needed or if prompted by IBM technical support.

2.5 Caching of MTQ results

Query caching behaviour is configured per cube via the VMM¹ value in the }CubeProperties cube, where the VMM value defines the maximum amount of memory to be used for caching per cube (i.e. the memory pool in RAM that is made available for caching). In many cases it is therefore a good practice to optimize/increase memory reserved for caching Stargate views by increasing the VMM value in the }CubeProperties Cube to a significantly higher value than the default of 128kb.

The use of MTQ will increase memory (RAM) usage. Subsequently, the use of MTQ typically will require an increase in VMM size. If VMM cache is set too low, even queries that were cached without MTQ use may not be cached anymore once MTQ is enabled. In such cases – to avoid unnecessary re-execution of MTQs - increase the VMM value until repeated query execution will not trigger MTQ activity anymore (indicating the cache is used).²

For larger to very large environments/cubes (in the double digit GB range or with hundreds of GB), an increase of the VMM cache to many MBs or even a few GBs may be useful, particularly for environments that are not highly volatile. For larger environments & where MTQ is leveraged heavily & with high concurrency, reducing the VMT threshold may be useful to reduce unnecessary (high) CPU utilization: With MTQ, a query that may a minute to run using one logical processor may run within let's say 3 seconds when using 24 cores. Particularly in high concurrency environments, such a query should be cached to not re-engage the 24 cores again (even if just for 3 seconds), hence freeing up computing resources for new queries or longer/larger queries

¹ TM1 setting thresholds and maximum cache memory for Stargate views. A Stargate view is a calculated and stored subsection of a TM1 cube that TM1 creates when you browse a cube with the Cube Viewer, Web-Sheet or In-Spreadsheet Browser. The purpose of a Stargate view is to allow quicker access to the cube data. A Stargate view is different from a TM1 view object. The Stargate view contains only the data for a defined section of a cube, and does not contain the formatting information and browser settings that are in a view object. A Stargate view that TM1 creates when you access a cube contains only the data defined by the current title elements and row and column subsets. TM1 stores a Stargate view when you access a view that takes longer to retrieve than the threshold defined by the VMT property in the control cube }CubeProperties. A Stargate view persists in memory only as long as the browser-view from which it originates remains unchanged. When you recalculate the browser view, TM1 creates a new Stargate view based on the recalculated view and replaces the existing Stargate view in memory. When you close the browser view, TM1 removes the Stargate view from memory. Stargate View caching behavior/thresholds (by cube) can be configured via changing the VMT and VMM values in the }CubeProperties cube: For each cube, the VMM property determines the amount of RAM reserved on the server for the storage of Stargate views. The more memory made available for Stargate views, the better performance will be. You must, however, make sure sufficient memory is available for the TM1 server to load all cubes. If no VMM value is specified the default value is 128KB. The valid range is 16 - 42934943296 kb (16-2³² kb).

² For larger cubes, increasing the VMM value to a value of between 1MB (1000) and 5MB (5000) is often a good start.

2.6 Use of MTQ for TI-processing

For TM1 10.2.2 <FP6 the following workaround can be used to leverage MTQ:

- a) Build the view in a separate process & use the ViewConstruct() function to 'query'/'construct' the view. This will trigger MTQ.
- b) Then, in a separate process, you may leverage the cached stargate view for processing. Note that this separate process may also build/rebuild this view and could do so using a different view name. What matters is that the views (the cached one from (a) and the new one from (b)) have the same content, i.e. that the corresponding stargate is the same.

Notes on this pre- FP6 workaround:

- The ViewConstruct and the data-processing TI-operations need to be separate. Using a trigger/master process to call two TI's will not work.
- You may use a chore to trigger both processes, but you have to use multi-commit mode
- The result of the ViewConstruct will need to go into the TM1 cache in order for the 2nd processes to leverage it => For large views, a significant increase in the VMM value may be required
- Only the visible part of the view will be cached when using ViewConstruct. It follows that if you are using views in TI that do not have all dimensions assigned as rows (ViewRowDimensionSet), the view needs to be re-build accordingly for leveraging MTQ
- Some views may be too large to warrant caching via ViewConstruct (VMM too high or not enough memory).
- Processing Leaf-level views may not always provide a performance gain (unless the leaf levels are driven by rule-based consolidations for example)

[As of TM1 10.2.2 FP6, the above workaround \(via ViewConstruct\) is no longer required in order to employ MTQ for TI-processing](#)

(see underlying hyperlink for more information on conditions & limitations).

- As of FP6, MTQ will be triggered in the context of the TI data tab itself.
- For individual TI processes, this auto-MTQ behaviour can be disabled by placing the function DisableMTQViewConstruct() in the prolog of the TI.
- Performance with this approach is slightly faster than if employing the pre FP6 workaround.
- MTQ for TI-views is only used for C-level views (Leaf-level views will not leverage MTQ unless a leaf value is derived per rule pointing to a consolidation).
- If the stargate was cached previously, MTQ will not be leveraged. In other words: The FP6 enhancement is cache-aware.
- The new TM1s.cfg parameter MTQQUERY can be used to enable/disable the use of MTQ for Views in TI processing
- The Planning Analytics Default is MTQQUERY=T
- MTQQuery will use MTQ to query and 'pre-cache C-level views used by TI-processes. While this typically leads to performance improvements, it **can also significantly increase RAM consumption for larger views**. That is because TI with MTQQuery=F / without MTQQuery

will not load a view into RAM, but instead 'cycle' through a view in smaller blocks, not really increasing RAM usage much. Yet with MTQQuery=T, TM1 will attempt to query and load the entire C-level view into RAM prior to processing it via TI

- ⇒ MTQQuery=T (default) can increase RAM use significantly
- ⇒ It is recommended to only use IF (i) size of views is managed (known to be small enough) and (ii) sufficient RAM is available, otherwise, use MTQQuery=F

2.7 Configure TM1 Parallel Interaction

Parallel Interaction is a TM1 database server configuration that improves response time of writing data by removing lock contention resulting from concurrent cube data access (when the system reads data or when the system writes data). When enabled, Parallel Interaction changes the Cognos TM1 Object Locking model so that write operations are not blocked by concurrent data read or data write operations to the same cube (or dependent cubes based on rules). Read operations always includes the most current write activity as of the time the read operation begins. Please see [Best Practices on TM1 Parallel Interaction](#) for details.

2.8 Improving TM1 DB Startup time

The following configurations can be used to shorten TM1 DB Instance Startup-Time if needed:

2.8.1 Configure MaximumCubeLoadThreads (TM1 10.2.2)

Enabling the MaximumCubeLoadThreads TM1 database server configuration parameter allows for multiple processor cores to be utilized on start-up of the TM1 database. When this parameter is not enabled, the TM1 server utilizes a single processor core for start-up. The optimal number of cores to use on start-up depends on a customer's specific implementation. Using more cores for start-up will consume more memory (due to duplicate feeder processing), but can result in significant start-up time improvements. IBM recommends starting with 50% of the number of available processor cores for your server and testing the trade-off of additional memory consumption and start-up time to determine the optimal setting for this configuration parameter.

To enable multi-threaded start-up, a configuration parameter must be added to tm1s.cfg file: MaximumCubeLoadThreads=<#ofThreads>.

For additional information, please see [IBM Cognos TM1 MaximumCubeLoadThreads](#).

2.8.2 Leverage MTCubeLoad, MTFeeders, and MTFeeders.AtStartup (TM1 11 / PA)

MTCubeLoad=T

- Default is F.
- Allows TM1 to use multiple threads/cores for Cube Load on Startup. Will likely allow for faster database startup. Click [here](#) for more information
- With MTCubeLoad=T, MaximumCubeLoadThreads will be disabled
- MTCubeLoad typically results in faster startup time than MaximumCubeLoadThreads AND does not increase startup memory like MaximumCubeLoadThreads does.

MTFeeders=T

MTFeeders.AtStartup=T

- Default is F.
- Allows TM1 to use multiple threads/cores for feeder processing. Will likely result in significantly faster 'refresh' & startup of database instances. Click [here](#) and [here](#) for more information. Note: use of MTFeeders functions is typically OK with conditional feeders that are based on attributes or regular lookup cubes.

2.8.3 PersistentFeeders

To improve reload time of cubes with feeders, set the PersistentFeeders configuration parameter to true (T) to store the calculated feeders to a .feeders file. The feeders will be saved to a *.feeders file (per cube) when the data is saved or rules are edited. On Startup and with PersistentFeeders=T, TM1 will load

feeders from the saved feeders file which reduces the time normally taken to recalculate those feeders. For installations with many complex feeder calculations persisting feeders and then re-loading them at server startup will improve performance. Note that for simple feeders, the time taken to read feeders from disk may exceed the time to re-calculate the feeders.

Only use persistent feeders if needed, i.e. if server startup time needs to be improved and after one has optimized rules & feeders.

2.9 Optimize Server handling of MDX Subsets

We recommend enabling the TM1 database server parameter `UseLocalCopiesforPublicDynamicSubsets` (in `TM1s.cfg`) to eliminate locking when using dynamic (MDX) subsets by using 'local' copies of the MDX subset when possible. Note that by default, i.e. if the parameter is not present in the `tm1s.cfg` file, `UseLocalCopiesforPublicDynamicSubsets` is enabled.

With `UseLocalCopiesforPublicDynamicSubsets` enabled, MDX subsets (instead of static subsets) should generally be used for all applicable queries and TI processing needs. Compared to static subsets, MDX subsets are typically faster to construct and easier to maintain due to their dynamic character.

2.10 Set `ViewConsolidationOptimizationMethod=Tree`

As per <http://www-01.ibm.com/support/docview.wss?uid=swg27044309>, the `tm1s.cfg` parameter **`ViewConsolidationOptimizationMethod`** defines the method used to achieve view consolidation optimization when the `ViewConsolidationOptimization` parameter is enabled (default) on the TM1 server.³ There are two methods that `ViewConsolidationOptimization` can use to calculate and store consolidations: `ARRAY` or `TREE`. The `ARRAY` method stores consolidations in a temporary array. The `TREE` method stores consolidations in a tree. `ViewConsolidationOptimizationMethod` should be set to `TREE` in most circumstances. The setting `ViewConsolidationOptimizationMethod=TREE` provides the best performance in normal operations.⁴

³ See https://www-01.ibm.com/support/knowledgecenter/SS9RXT_10.2.0/com.ibm.swg.ba.cognos.tm1_inst.10.2.0.doc/c_viewconsolidationoptimization_1.html%23ViewConsolidationOptimization_1

⁴ In rare instances, using the `TREE` method can result in a degradation of performance. In such an instance, try setting the parameter to `ARRAY`. For example, in the uncommon circumstance when dimensions have just a few leaf elements rolling up to many consolidations, `ViewConsolidationOptimizationMethod` should be set to `ARRAY`.

2.11 Optimize the use of the TM1 Query cache via configuration of the VMM parameter

2.11.1 Query Cache configuration via VMM & VMT: Introduction

Query caching behaviour is configured per cube via the VMM value in the }CubeProperties cube, where the VMM value defines the maximum amount of memory to be used for caching per cube. The 'legacy' VMM default value is 128KB and hence only allows caching of smaller views and a small number of views (because the 128kb threshold will be reached relatively quickly).

In many cases it is therefore a good practice to optimize/increase memory reserved for caching Stargate views by increasing the VMM value in the }CubeProperties Cube: TM1 allows to set thresholds and maximum cache memory for Stargate views. A Stargate⁵ view is a calculated and stored subsection of a TM1 cube that TM1 creates when you browse a cube with the Cube Viewer, Web-Sheet or In-Spreadsheet Browser. The purpose of a Stargate view is to allow quicker access to the cube data. A Stargate view is different from a TM1 view object. The Stargate view contains only the data for a defined section of a cube, and does not contain the formatting information and browser settings that are in a view object. A Stargate view that TM1 creates when you access a cube contains only the data defined by the current title elements and row and column subsets. TM1 stores a Stargate view when you access a view that takes longer to retrieve than the threshold defined by the VMT⁶ property in the control cube }CubeProperties.⁷ A Stargate view persists in memory only as long as the browser-view from which it originates remains unchanged. When you recalculate the browser view, TM1 creates a new Stargate view based on the recalculated view and replaces the existing Stargate view in memory. When you close the browser view, TM1 removes the Stargate view from memory. Stargate View caching behavior/thresholds (by cube) can be configured via changing the VMT and VMM values in the }CubeProperties⁸ cube: For each cube, the VMM property determines the amount of RAM reserved on the server for the storage of Stargate views. The more memory made available for Stargate views, the better performance will be. You must, however, make sure sufficient memory is available for the TM1 server to load all cubes. If no VMM value is specified the default value is 128KB. The valid range is If no VMM value is specified the default value is 128KB. The valid range is 16 - 42934943296 kb (16-2³² kb)⁹.

2.11.2 Configuration use cases & recommendations for VMM Query Cache configuration

- (a) End-user queries: For applicable cubes, it is a good practice to increase the VMM value to a new 'default' value of between 1MB and 10MB or even higher (depending on the cube's average stargate size and the number of expected queries/stargates to be requested in between data updates). This will allow caching of relatively large as well as many views and improve performance accordingly.
- (b) Note that when using Cognos BI to query a TM1 Cube, the default VMM value (unless defined otherwise) will be increased to 10MB. This default will remain in place until the TM1 instance is

⁵ [TM1 documentation on Stargate Views](#)

⁶ VMT: For each cube, this property defines the time threshold, in seconds, beyond which the algorithm that stores TM1 Stargate views is triggered. If the time required to calculate a cube view surpasses the specified threshold, TM1 attempts to store a Stargate view. If there is not enough memory available to store the Stargate view, TM1 purges the oldest Stargate view that is not currently in use, and continues to purge views in this manner until sufficient memory is made available. If no VMM value is specified the default value is five seconds. The valid range is 1 - 259,200 seconds. A value of 5 seconds is a good starting point for this threshold value.

⁷ If a VMT value is not explicitly defined, a Stargate view is generated when a view takes longer than five seconds. This is the default threshold when VMT is not specified in the }CubeProperties control cube.

⁸ [TM1 documentation on }CubeProperties](#)

⁹ going over the max value will cause overflow. Ensure that (i) sufficient memory is available to cover VMM cache pool RAM requirements and that you do not configure the cache values above the max.

restarted. In other words: if VMM is not specified, the default is 128kb until a BI query is run which will increase the default to 10MB.

- (c) When VMT is set to 0, the default VMM value (unless defined otherwise) will be increased to 10MB.
- (d) Running and monitoring representative sample queries via TM1 Operations Console will indicate when the Query Cache is leveraged (short query time and no MTQ activity) vs. if the cache is not being used (long query times & MTQ activity). If the VMM value(s) are too low, A query may not be cached. In such a scenario, running the same query repeatedly will cause repeated query execution & calculation vs. retrieval from the cache. If such scenarios are common, i.e. if certain common queries will not be cached due to low cache size, increase the cache until the query result is retrieved from cache.
- (e) The use of MTQ typically will require an increase in VMM size. If VMM cache is set too low, even queries that were cached without MTQ use may not be cached anymore once MTQ is enabled. In such cases – to avoid unnecessary re-execution of MTQs - increase the VMM value until repeated query execution will not trigger MTQ activity anymore (indicating the cache is used).
- (f) For larger to very large environments/cubes (in the double digit GB range or with hundreds of GB), an increase of the VMM cache to many MBs or even a few to many GBs may be useful, particularly for environments that are not highly volatile. => If you have sufficient RAM reserves, do not shy away from maximizing cache. You may assign as much cache pool memory as you see fit (for as long as the aggregate cache memory pool plus the server memory utilization falls within the overall RAM capacity of the environment/machine.

2.11.3 Configuration use cases & recommendations for VMT Query Cache configuration

- (a) For smaller environments, where MTQ is not leveraged heavily, leave the VMT at its default of 5 seconds (= empty)
- (b) For larger environments & where MTQ is leveraged heavily & with high concurrency, reducing the VMT threshold may be useful to reduce unnecessary (high) CPU utilization: With MTQ, a query that may a minute to run using one logical processor may run within let's say 3 seconds when using 24 cores. Particularly in high concurrency environments, such a query should be cached to not re-engage the 24 cores again (even if just for 3 seconds), hence freeing up computing resources for new queries or longer/larger queries

2.12 Optimization of Cube Logging and maintenance of TM1 log Files

For as long as a large cube is used for analysis/reporting purposes only and updated on a scheduled basis, it is a good practice to disable cube logging for that cube as this will speed up data load performance. Because a read-only cube cannot be accidentally overwritten by users, cube logging is not needed for audit purposes. For Plan/Budget and other input cubes etc., cube logging should however be enabled to allow for proper auditing to occur.¹⁰

Once data is saved via the SaveDataAll() or CubeSaveData() functions¹¹, the corresponding log entries will be removed from the TM1s.log file and the logs will be 'archived' in new log files named tm1s<DateTimeStamp>.log such as tm1s20140210172014.log.¹²

Furthermore, we recommend using the LoggingDirectory parameter in the TM1s.cfg file to direct TM1 to create and maintain log files in a directory separate from the Data directory. Doing so will simplify archiving of log files and TM1 database objects.¹³

For further information on TM1 logging and on interpreting TM1 transaction log files, please refer to [TM1 Transaction Logging](#)

¹⁰ to build/change a TI process that automatically & temporarily will disable cube logging during data load one can implement the following logic:

```
Prolog:
    nLogFlag = 0;
    sOldCubeLogChanges = CUBEGETLOGCHANGES(sTargetCube);
    CUBESETLOGCHANGES(sTargetCube, nLogFlag);
Epilog:
    CUBESETLOGCHANGES(sTargetCube, sOldCubeLogChanges);
```

Caveats:

- If such a TI process does not complete (due to processquit or a fatal error or being cancelled by an administrator), or if the TM1 server is shut down while such a TI process is running prior to the process epilog being run, logging will be DISABLED for the cube.
- The subsequent absence of logging could then lead to a data loss in the case of unplanned server outages or crashes.
- To prevent such a scenario from occurring (where logging is turned off inadvertently), it can be a good practice to set up a chore that will periodically check if logging for crucial cubes is enabled, and if not, re-enable cube logging.

¹¹ TM1 Data that is loaded into a cube is not automatically committed to disk but retained in-memory only. Use the functions CubeSaveData()¹¹ to commit one cube's data and/or the SaveDataAll()¹¹ to commit all data to disk. Note that SaveDataAll() and CubeSaveData() will acquire a write lock on the cube/database. It is hence not recommended to use SaveDataAll() and CubeSaveData at the end of each TI load process (because that will cause lock-contention with other load processes that load data to any cube (SaveDataAll()) or also load data to the same cube (CubeSaveData()) in a parallel load scenario (see below). There should be only one TI process that calls the SaveDataAll() function. Use a stand-alone, single, distinct chore to execute the SaveDataAll operation.

¹² Note that Undo operations will evaluate all applicable TM1 logs in the TM1 Log directory. If a large number of log files can be found in the Log directory, the UNDO operation may take a long time (and may lead to a significant lock on the corresponding cube). It is therefore recommended to archive older log files to a separate archive directory (for example: every day, schedule a job that will move the prior days log files to a separate directory)

¹³ Example:

```
DataBaseDirectory=.\TM1Data\
LoggingDirectory=.\TM1Logs\
```

3. General Design Recommendations

3.1 Recommendations on dimensionality

Particularly for large cubes, try to limit the # of dimensions in the cube to the dimensions that are really needed. An additional dimension (regardless of sparsity) can increase total memory consumption of the cube by up to 50% due to the additional storage required for the data element. Therefore, for large cubes, do not create a specific 'measures' dimension unless it really is needed. Evaluate if a 'regular' dimension can fulfill the technical purpose of the measures dimension (for example the currency dimension).

3.2 Recommendations on calculations via Rules

- Avoid the use of Alias Names in Rules: Calculations that reference an Alias (such as 'Current Month' for example) will have changing elements as the basis for the calculation. While the calculation statement (the rule) will not change, and underlying data may not have changed, the data being referenced will. Such Alias-based calculation rules are dependent on the alias mapping as defined in the attributes cube - hence a dependency is needed to perform invalidation (or stale data can persist after version submission.) Therefore, dependencies to the attribute cubes are established once such alias-based rules are encountered. The establishment of those dependencies can cause a temporary lock: In normal operations, cube dependencies are established when data which crosses cube boundaries (such as data that is derived by a rule that references an external cube) is retrieved. To create the dependency information, the server must lock the cubes while the dependency is established, potentially maintaining the lock during a long view calculation. Since this is a 'write' lock, other users are prevented from accessing the cubes.
- Feeders that feed from or to an element via its alias will have to be re-processed when/if/once the alias changes. Therefore, only use alias names in feeders for rules where the alias does not change frequently. Example: It is an acceptable practice to use and feed from Alias Names such as ActualsMonth1, ActualsMonth2 etc. for the fiscal months of the 'current' fiscal year (2015 01, 2015 02, ...) because feeders will have to be re-processed once a year (when the current year changes and the alias names roll over to the next year). Even monthly feeder-reprocessing due to changing alias names may be acceptable depending on data volume.
- Aim for a dimension and cube design that does not need rule feeders for calculations of for example YTD, QTD, AVG, EOP, Variances, and that does not use specific measures for YTD, QTD, AVG, EOP etc. Such a design can often be achieved via building the time dimension such that time periods that require rule calculations are only run against C-Level time dimension elements instead of writing the rule against a separate measure such as AVG and YTD. Additional measures & metrics often are created as N-level elements and rules and hence will require the N-level rules to be 'fed' via feeder statements. Such rule feeders would (i) significantly affect (increase) server start-up time for large cubes and (ii) increase memory requirements and (iii) result in longer query times. While N-level rules and feeders often cannot be avoided or are beneficial in that they may allow / provide a desired user experience, many times TM1's native consolidation algorithm, possibly combined with 'feeder-less' C-Level rules can be used instead, resulting in significant performance gains: C-Level time dimension elements will automatically be calculated if only one N-Level time dimension descendant is <> 0. Feeders can hence be avoided, hence significantly limiting cube size (no feeder memory needs) and start-up time (no evaluation of feeders necessary).

- The same approach applies to elements against which to calculate % and ratios: In cases where the ratios or % values are directly based on the value of a particular (other) measure (as a numerator or denominator), it is – particularly for larger models - a good practice to create ratio, % & variance rules as so-called feeder-less C-level rules, where the ratio/variance measure/element is made an ancestor of the non ratio/variance base element(s). In other words: If a % or ratio account/cost element/measure etc. is to be calculated for every cell and if the calculation is based directly on another account/cost element/measure either in the numerator or denominator, then make the % or ratio element a C-level element – with the other element as its descendant. You will not have to feed the calculation because for as long as the N-Level descendant carries data (the numerator or denominator is <> 0), it will be calculated.
- If a rule contains multiple aggregations (like $x = (a + b + c - d) * e / f$) consider using a rollup (where applicable) for all or even just parts of the aggregations. I.e. if a calculation includes the aggregation of a number of accounts as part of its logic, create a corresponding account rollup, and leverage the account rollup in the calculation. TM1's native aggregation algorithm is much faster than a rule-based aggregation, hence making such rules perform much faster.
- For TM1 rules and feeders, specify the left side of the rule/feeder (the AREA) as detailed as possible. Rather than write a rule that applies to many cells like

```
[ ] = N: IF ( !Currency @= 'Local Currency' & ( !Version @= 'Working Budget' % !Version @= 'Working FCST'), ..., stet );
```

Write the rule such that the left side (the AREA) is as specific as possible (even if this means including many elements), like:

```
[{'Working Budget','Working FCST'},'local currency'] = N: ...
```

Such a design will ensure that when TM1 evaluates a query it will only evaluate rules against the intersections that are affected by the rule vs. evaluating each intersection ([]) against the rule. In other words: the rule

```
[ ] = N: IF ( !Currency @= 'Local Currency' & ( !Version @= 'Working Budget' % !Version @= 'Working FCST'), ..., stet );
```

will be evaluated against each N-Level data point and the evaluation will determine that the no rule is to be applied. But before TM1 comes to that conclusion it will evaluate each intersection. For smaller cubes, the duration of this evaluation is insignificant. For large cubes, it can add many seconds to even minutes to the query time. The better rule

```
[{'Working Budget','Working FCST'},'local currency'] = N: ...
```

will automatically only be applied to intersections with versions 'Working Budget' & 'Working FCST' and currency 'local currency'. All other intersections will immediately be ignored.

Further reading and Examples:

- For more information on how the above design concepts can be applied to Time Dimension designs, please refer to the document 'Proven TM1 Practices for Continuous Time Period Dimension Design and Time-related Analysis including Design Template'.
- Please refer to the document 'Proven Practices for TM1 Version and Scenario Management and Variance Analysis including design template' for further information on proven practices around version/scenario management and version/scenario-related variance analysis.
- The 'TM1 Rolling Forecast and Planning Artefact' combines the aforementioned documents & provides rule templates for FCST/Planning functionality.
- Click [here](#) for a guide outlining caveats on the use of Alias Names in rules and feeders.

3.3 Rules vs. TI

3.3.1 Overview

Particularly for large cubes, evaluate if a custom calculation can be performed faster or more efficiently via TI process, either via a separate TI process or on the context of a data-refresh. Note that replacing a rule with TI is not always the best option. A rule may only require minimal feeding or in some cases no feeding at all ('feeder-less' C-Level rules, see above) whereas processing via TI may incur significant processing time. On the other hand, simple metrics that can be derived via simple calculations during runtime of the initial data load can often be processed very rapidly. Example: if the original load file contains 'Sales Volume' and 'Unit Price', 'Sales \$' should be processed as part of the initial data load by simply multiplying values from the 'Sales Volume' & 'Unit Price' measures.¹⁴

3.3.2 Guidelines

1. Reporting & Analysis: if the data is available in the source system, use TI
2. If the data 'moves' according to a low-volatility submission or closing cycle, use TI
3. As of TM1 10.2 MTQ, consider rules where you previously did not dare to
4. Keep in mind that cross-cube rules will always be much slower than processing the data. Therefore, in situations where larger amounts of data are ruled from one cube to another cube and where query performance is slow for that data in the target cube, replacing the rules with a TI process (if applicable from a procedural perspective) typically leads to significant performance gains.
5. For large & volatile data-sets, TI re-processing may be too time-consuming
6. Are calculations via weighted consolidations/rollups an option (like the Time and Variance calculations in our previous examples)?
7. Are feeder-less C-Level rules an option (like for our % Variance calculations from before)?
8. For small cubes / data volumes, rules are often a better choice than TI, due to
 - insignificant performance difference,
 - higher user-friendliness (real-time calculations & fewer processing steps)
9. Use rules for calculations against highly volatile data (input/planning data)
10. For very large cubes (hundreds of millions to billions of populated leaves, try to avoid feeders (and hence leaf level rules), unless the cubes do not get updated often and the duration of feeder (re-)processing – even with PersistentFeeders – is manageable. Use feederless, C-level rules for FX, Variance, and other applicable calculations.
11. If in doubt, evaluate the trade-offs between Performance, Usability & Maintenance:

	Performance Impact	Usability & Maintenance Impact
TI-Process	Processing within acceptable timeframe? Y/N	Can processing be automated? Y/N Do end-users have to manually process? Y/N
Cube Rules	Good Calculation Performance? Y/N	Can cube rules be fed efficiently? Y/N What is the maintenance effort for the rules & feeders? Low/High

¹⁴ 'Margin' as in 'Sales' – 'COGS' or 'Margin %' as in 'Sales'/'COGS' is not a good candidate for TI: 'Margin' can be calculated via a simple rollup (with 'Sales' as a descendant of 'Margin' weighted at 1 and with 'COGS' as a descendant of 'Margin' weighted at -1), 'Margin %' should be a 'feeder-less' C-Level rule, with 'Sales' and 'COGS' as descendants of 'Margin %' and with a C-level rule calculating 'Margin %'

3.3.3 Sample Scenarios

Scenario	Calculation Method	Performance Impact	Usability & Maintenance Impact
FX conversion for Input Data	TI-Process	Processing within acceptable timeframe? -> TBD	Can processing be automated? -> No Do end-users have to manually process? -> Yes
	Cube Rules	Good Calculation Performance? -> Yes	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
Sales \$ Actuals based on Price & Volume	TI-Process	Processing within acceptable timeframe? -> Yes (superior. update along with Act update)	Can processing be automated? -> Yes Do end-users have to manually process? -> No
	Cube Rules	Good Calculation Performance? -> Yes (but feeding required)	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
FX conversion for Actuals	TI-Process	Processing within acceptable timeframe? -> FX conversion data in Source System => Yes	Can processing be automated? -> Yes Do end-users have to manually process? -> No
	Cube Rules	Good Calculation Performance? -> Yes	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
FX conversion for Actuals	TI-Process	Processing within acceptable timeframe? -> FX conversion data NOT in Source System, multiple FX conversions per record => No	Can processing be automated? -> Yes Do end-users have to manually process? -> No
	Cube Rules	Good Calculation Performance? -> Yes	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
Driver-based FCST, Planning & Analysis	TI-Process	Processing within acceptable timeframe? -> Yes	Can processing be automated? -> No Do end-users have to manually process? -> Yes
	Cube Rules	Good Calculation Performance? -> Yes (for large cubes, MTQ may be required)	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
Profit Center Allocations & Apportionments	TI-Process	Processing within acceptable timeframe? -> Yes	Can processing be automated? -> Yes Do end-users have to manually process? -> No
	Cube Rules	Good Calculation Performance? -> TBD	Can cube rules be fed efficiently? -> TBD What is the maintenance effort for the rules & feeders? -> High
Spreading of FCST/Plan (top-down) by YTD Act	TI-Process	Processing within acceptable timeframe? -> Yes	Can processing be automated? -> Yes Do end-users have to manually process? -> No
	Cube Rules	Good Calculation Performance? -> Yes (for large cubes, MTQ may be required)	Can cube rules be fed efficiently? -> Yes What is the maintenance effort for the rules & feeders? -> Low
=> Outcome depends on particular spreading logic => what is easier to implement and maintain?			

3.4 Recommendations on cube dimension order

3.4.1 Affect of Dimension Order on Performance

Because the order in which dimensions are assigned to a cube affects the way the OLAP data is organized/stored, the dimension-order in the cubes affects system performance. Fortunately, TM1 allows the changing of the 'internal' dimension order of a cube without requiring a change of the initial dimension order that was set when the cube was created. The initial dimension order is what end-users will see when browsing / querying the cube. Dimension sort order optimization hence can become a performance optimization task **after** a cube was created.

- The order of dimensions significantly affects cube memory requirements and query performance.
- The relationship between memory requirements and query performance is not linear. For example, a 20% reduction in memory requirements may in some cases be indicative of a x5 improvement in query speed (for some queries). At the same time, there is typically no optimal cube dimension sort order that works best for all types of queries against a cube. An improved dimension sort order can hence also cause query performance degradation for some queries. It is however rare that a significant memory reduction due to dimension sort order optimization leads to a general performance degradation.
=> After optimizing dimension sort order as described below, ensure that the main queries or query types do not perform slower than before. If they do, re-optimize and evaluate against a set of representative queries.

3.4.2 Initial Dimension Order

You do not have to worry about the performance impacts of the dimension order when creating a cube. You can optimize the internal dimension order later (next slide). Therefore, to improve usability it is a common & proven practice to determine and use a common general dimension order for all cubes of a TM1 model. We hence recommend to determine an initial dimension sort order based on the optimization guidelines above or based on query use-patterns (i.e. aligning the initial dimension sort order with how users interpret, perceive and use the dimensions of a cube) and to apply this same sort order to all cubes of a particular model. If for example the Account dimension was chosen as the first dimension of a model it shall be the first dimension for all other cubes with an account dimension. Additional guidelines:

- if a specific measure dimension is used (with measures such as 'Value' or \$ or Units) it shall always be the last dimension of a cube.
- Use the same grouping for non-model specific dimensions that may be part of most if not every cube.

Examples:

- last dimension = specific measure dimension (if it is needed for the cube),
- 2nd to last or last dimension): Currency
- 3rd or 2nd to last dimension): Time
- 4th or 3rd to last dimension): Version

3.4.3 Optimizing dimension sort order

It is likely that one will specify an order of dimensions during cube creation that results in less than optimal performance. Relative sparsity and relative density of data as it is stored in a cube (in alignment with the order of dimensions) typically changes over time yet often will 'stabilize' in the sense that a production environment that has been in use for a period of time typically contains a representative dataset and hence will give an architect a true indicator of data density and sparsity. To allow for easy optimization of the dimension order in existing cubes, TM1 features the 'Cube Optimizer': the cube optimizer lets you re-arrange & evaluate a different, internal order of dimensions in a cube and apply the

new sort order to the cube if desired, all without having to rebuild the cube.¹⁵ When you optimize the order of dimensions in a cube, TM1 does not change the actual order of dimensions in the cube structure. TM1 does change the way dimensions are ordered internally on the server, and because the cube structure is not changed, any rules, functions, or applications referencing the cube remain valid.

Step 1: System-Generated dimension sort order optimization

One can have TM1 itself evaluate the cube and dimension structures and determine a system-generated optimal dimension sort order. The system-generated 'optimal' dimension sort order should be the starting point for dimension sort order optimization. For small and already very fast cubes, a manual optimization may not be necessary. Yet for medium-sized cubes and even larger cubes, a further optimization may prove to be very effective: The system-generated dimension sort order optimization typically manages to optimize cubes only to a certain degree. The system-generated dimension sort order optimization will not consider the last dimension & does not analyze relative sparsity and density as efficiently as possible. For larger cubes and/or for rules-heavy cubes where performance improvement is desired, we recommend to further optimize performance via manual dimension sort order optimization:

Step 2: Manual optimization of dimension sort order. Guidelines:

- a) Divide the dimensions into two groups: sparse and dense dimensions.¹⁶
- b) Order the dimensions as follows: smallest sparse to largest sparse, followed by smallest dense to largest dense.
- c) Exceptions: It typically is better to put a very small, dense dimension before a very large but sparse dimension. For example, a dimension such as Version/Scenario (Act, FCST, Plan) that has only two or three elements is better positioned before a very large but sparse dimension, such as Product, which might have thousands of elements. => be flexible and experiment with different configurations.¹⁷
- d) If a dimension (such as a measures dimension) contains string elements, it needs to be the last dimension.
- e) Very often, moving the measures (or last) dimension (if it does not include string elements) upwards (from the last position) can result in significant memory savings and performance gains. The TM1 system-generated cube optimization will not consider moving the measures/last dimension. The impact of moving the measures/last will have to be evaluated manually. If a cube has a measures dimension with just a few or even just one element in it, such a very dense dimension is the perfect

¹⁵ From the TM1 Manual:

- "Significant memory resources are required for the TM1 server to reconfigure the order of dimensions in a cube. During the re-ordering process, the temporary RAM on the TM1 server increases by a factor of two for the cube that you are re-ordering. For example, a 50 MB cube requires 100 MB of RAM to reconfigure."
- "Re-ordering puts a read lock on the server, locking all user requests while the re-order is performed."
- "You must be a member of the ADMIN group to optimize the order of dimensions in cubes. The optimization option is only available for cubes on remote TM1 servers; you cannot optimize the order of dimensions in cubes on a local server. Also, when you optimize the order of dimensions in a cube, you should not move the string dimensions from the last position, nor move the string dimensions to the last position."

¹⁶ From the TM1 Manual: "A dense dimension has a high percentage of values for its elements. You can estimate the density by answering this question: If one element in the dimension has a value, keeping the elements of the other dimensions constant, what is the probability that the other elements in the dimension have values? For example, if you have a budget in January for a given account and region, you probably also have a value for the remaining months. Therefore, the Month dimension is probably dense. Similarly, if you have a budget value for a given month, account, and region, you probably also have an actual value, making ActVsBud a dense dimension. However, in a worldwide sales cube, you probably do not sell every product in every region. Therefore, you would treat Product and Region as sparse dimensions."

¹⁷ For some TM1 models/cubes manually moving a continuous time dimension to be the last or 2nd to last dimension many times actually results in memory reduction & performance gains.

candidate for guideline (c) above. Making such a dimension the first dimension of a cube in most cases renders significant query performance gains.

- f) If one finds that moving the measures dimension upwards will render significant performance gains but you would like to use string elements too (for comments etc.), one may consider building a separate cube just for string measures.
- g) The TM1 manual states that one “should optimize the order of dimensions in a cube only in a development environment while you are trying to determine optimal cube configuration”. This should be interpreted as follows: Determine the optimal cube order via tests in a production-like development environment (with Production Data Volumes). Then – at a time of zero-concurrency – implement the tested new dimension order in your production environment.

Note that relative sparsity and relative density of data as it is stored in a cube typically changes over time yet often will ‘stabilize’ in the sense that a production environment that has been in use for a period of time typically contains a representative dataset and hence will give you a true indicator of data density and sparsity.

3.5 Security

3.5.1 Performance Considerations in TM1 Security

- Particularly in larger environments with a high number of users, there can be an observable time difference between queries executed by an admin user vs. queries by 'secured' end-users. This is because TM1 will ignore the security layer for admin users, whereas for 'regular' users, TM1 will apply security on top of / for the requested query. Depending on how security is implemented, this may require moderate to heavier processing overhead for determining security. Heavier processing is typically invoked for cubes that are secured via cell security. While there will always be a small query performance difference between those two user types (because TM1 will ignore the security layer for admin users), the query response delta can most often be reduced drastically by
 - optimizing the cell security cubes (fewer dimensions),
 - by fine-tuning cell security rules and – where applicable –
 - changing default cell security behaviour.
- Furthermore, the security model shall be designed such that locking due to security-updates is minimized. In the following sections, we are providing some details on the aforementioned practice recommendations.¹⁸

3.5.2 Avoid locking due to Security Changes

It is not recommended to use rules particularly for }ElementSecurity. We recommend leveraging a security staging model¹⁹ to determine and then process Security Credential values instead:

Cube Security (')}CubeSecurity.cub'), Dimension Security (')}DimensionSecurity.cub'), Process Security (')}ProcessSecurity.cub'), and Element Security Data (')}ElementSecurity_<Dimension>.cub') should be processed via TI instead of cube rules. **If rules are used, a security metadata change - for example due to a hierarchy change (with corresponding/resulting security changes for parent and/or child nodes) or due to a new element being added to a hierarchy (like a new archived version for which READ access now to be granted to all applicable groups) – will always require running the 'SecurityRefresh()' command in TM1, effectively rendering all cached security settings invalid and hence renewing/refreshing all security credentials. A security refresh on large models will typically lead to a multi- to many minute lock of all user activity due to TM1 refreshing security access credentials for all active users and groups.**

If security is manually entered or processed via TI (and hence directly stored in the corresponding security cube), a security refresh is not necessary for such security changes. The security changes will propagate automatically and with only very short locks.

Note that Cell Security Cube rules are re-evaluated automatically once a user refreshes a query. It follows that a SecurityRefresh() is not required to refresh credentials established by cell security rules.

¹⁸ Guidance and recommendations pertaining to TM1 Workflow Security is available [here](#).

¹⁹ A Security Staging & Maintenance Model is designed to be

- a staging ground or mirror of the TM1 security model

- with additional cubes/features/external data loads that feed into the staging model to simplify security maintenance

We use the security staging & maintenance model to configure security and then push the data from the Security Maintenance Model to the TM1 Security Model

3.5.3 Optimize Security Model Performance (Query-Overhead due to Security)

3.5.3.1 Use the SKIPCHECK statement in CellSecurity Cubes and do not feed those rules

With skipcheck, CellSecurity rules will be evaluated regardless (feeders are not needed) yet testing of CellSecurity will result in much better performance and lower memory consumption.

3.5.3.2 Reduce the # of dimensions in CellSecurity cubes to only the dimensions needed to defined cell security

As of TM1 10.2, Cell Security performance can perform much faster because a cell security cube only needs to contain the dimensions needed for defining cell security. Example: if your securing a cube against Cost Center and Version only, your cell security cube only needs to include those two dimensions plus the }Groups dimension. For CellSecurity, it is recommended to only include dimensions in the }CellSecurity cube that are needed to define the restrictions and access credentials to determine access rights and restrictions. A new function named CellSecurityCreate is available to optimize CellSecurity cubes.²⁰

Note that ElementSecurity may also be used to supplement CellSecurity in case of dimensions that are not needed for CellSecurity Purposes (i.e. one could define cell security restrictions by omitting Scenario security, but implement scenario security per element security).

Optimizing the CellSecurity cube by reducing its dimensions will reduce security-related query response times by an order of magnitude for larger cubes because TM1 will be able to process and determine security much faster against a smaller cube.

3.5.3.3 Change CellSecurityDefaultValue in }CubeSecurityProperties for applicable cubes

More recent version of TM1 allow changing the CellSecurity default (if a cell in the cell security cube is empty). By default, an empty cell in CellSecurity will lead to Cube, Dimension and ElementSecurity being applied. Some CellSecurity rules or regimes however call for Security to be explicitly defined via CellSecurity. Rather than to define a default per rule, the default can be specified via }CellSecurityProperties (by cube).

For example, if cell security is used to determine READ access at the intersection level (like Department and Account combinations), and unless the user belongs to a group that gives her access to a particular intersection or group of intersections, the access will be NONE. Under normal (default) Cell Security behaviour, the rules would then have to be designed such that NONE access would be applied by rule wherever READ was not to be granted. By changing the CellSecurity default to NONE, the rules would only have to be written to provide READ access (because the default would be set to NONE). As a result, it may be possible to write simpler (as in more performance optimal) cell security rules. In addition, maintenance is simplified in that there will be one entry that ensures that default access is to be NONE (rather than needing possibly multiple rules with a NONE output or even one rule like `[]:=S'NONE'`; (which - per section <Recommendations on calculations via Rules> – should be avoided if possible).

²⁰ http://www.ibm.com/support/knowledgecenter/SS9RXT_10.2.2/com.ibm.swg.ba.cognos.tm1_ref.10.2.2.doc/r_tm1_ref_tifun_cellsecuritycubecreate.html

3.6 TI-process Design

3.6.1 Employ a modular TI-process design methodology

From the perspective of maintainability, it is a recommended practice in larger TM1 environments to build TI-processes that are more modular and can for example process a task (such as a zero-out or the copying/promotion/submission of an input version to a version iteration) for a set of cubes specified at runtime. Such modular TI-process environments benefit from lower ongoing maintenance cost because they share a common modular code base where every modular functionality ideally has to be maintained (and fixed) only once.^{21, 22}

	Non-Modular TI-processes	Modular TI-Process environments
Complexity	Higher complexity, primarily due to code repetition	Little to no code repetition => lower complexity, higher transparency
Development Time	Longer development time (due to copy, pasting & adjusting of code portions)	Shorter development time (due to applying and calling code modules)
Testing efforts	The copy, paste, adjust approach can easily lead to small mistakes (that could have a large impact) => higher testing efforts	Lower testing efforts because the code in the module components only needs to be tested once.
Maintainability	High effort (changes need to be made in every applicable process)	If changes affect a process module, changes only need to be made once
Consistency	Lower consistency (consistency depends entirely on thoroughness and accuracy of individual developer)	High consistency
Portability	Very low. Changes or enhancements have to be implemented per process	Very high

²¹ A modular TI-process design approach often necessitates implementing security checks at runtime to ensure the end-user has the rights to run a TI process against the specified (selected) cube(s) and/or dimensions and/or elements. Such a security regime can be put in place by creating a generic sub-process that is built to check a users' access credentials against security models/cubes. Sample methodology/code (based on an IBM TM1 asset utility) and assuming a check for Group access:

A) in the prolog of the TI-process that requires a security check, insert the following code:

```
#security checks (check if user is allowed to write)
sSecurityAccess = "";
sSecurityAccessViolationMessage = "";
IF ( SCAN ( 'R*', TM1USER() ) > 0 % CellGetS ( 'ClientGroups', TM1USER(), 'ADMIN' ) @= 'ADMIN' % CellGetS ( 'ClientGroups', TM1USER(), 'DataAdmin' ) @= 'DataAdmin' );
    # Chore (in which case TM1USER() returns 'R*<ChoreName>')
    # ADMIN or DATA ADMIN => allow
ELSE;
    # NON-ADMINS will have to determine a Group! ###
    IF ( pGroup @= " );
        sError = 'please specify a value for Group!';
        processbreak;
    # Now that we have ensured that a Group is entered, we can
    # Validate Group Security Access
    ELSEIF ( ExecuteProcess ( 'SYS_IBM_Security_Validate_Element_Access', 'pDimensionName', 'Group', 'pElementName', pGroup, 'pElementAccess', 'WRITE' )
    <> processexit(normal());
        sSecurityAccessViolationMessage = 'TM1 User does not have sufficient access credentials to "' | 'Group' | '" "' | pGroup | '".';
        sSecurityAccess = 'N';
        processbreak;
    ENDIF;
ENDIF;
```

Where SYS_IBM_Security_Validate_Element_Access' is the sub-process that checks security credentials.

B) in the epilog, insert the following code:

```
IF ( sSecurityAccess @= 'N' );
    itemreject ( sSecurityAccessViolationMessage | ' Processing was aborted.' );
ENDIF;
```

²² Highly modular TI-processes (not processes that just modularize View & Subset creation, but processes that employ sophisticated design methodologies in the data tab for fact data processing to 'any' cube) can result in a longer runtime. Yet even for mid- to larger sized models, the longer runtime typically is not materially significant, particularly for procedures that are not run very often.

4. Recommendations for Data Load/Update procedures

4.1 Load data using TM1 ODBC interface

General maintenance as well as the orchestration of parallel data load/update tends to be easier if SQL Queries are used rather than flat files. SQL queries can be changed faster and loads can be partitioned quicker via use of and orchestration of specific SQL queries rather than having to maintain a separate ETL file extraction system.

In addition, the utilization of ODBC as a means for TM1 to extract/load data eliminates the security risks and maintenance tasks associated with securing a file-based ETL infrastructure.

	ODBC	File
Speed (Throughput)	Low to Very High (depending on DB performance, driver & infrastructure) Very High: 23.000 records / second (against Oracle DB at Fin. Sector Client in Chicago)	Medium (for very large files) to Very High
Maintainability	Low Effort (SQL can be parameterized)	Medium to High Effort (DB ETL routines may have to be changed, multiple files may have to be extracted, ETL needs orchestration)
Complexity	Low	Low
Security	High	Medium

4.2 Optimizing Performance

4.2.1 Use CubeClearData() command for complete re-load of cube data

Use the `CubeClearData()` command instead of `ViewZeroOut()` to clear all data from a cube prior to re-loading for example adjusted historical data: `CubeClearData()` will clear all data from a cube and is significantly faster than the `ViewZeroOut()` command (which should be used for zero-out of sections of a cube).²³

4.2.2 Separate TI processes for dimension maintenance and data load

Instead of adding new dimension elements during a data-load TI-processes, create separate TI processes for dimension maintenance. Doing so will – via procedural separation of metadata and data-update procedures - allow the load and update of cube data using parallel load threads²⁴ (because with parallel interaction a cube lock is not established if only data is being updated), hence significantly improving cube load time.

In cases where new dimension elements need to be added in one TI process along with loading new data, the dimension update should occur

- in the TM1 data tab
- while using the TM1 10 functions `DimensionElementInsertDirect` & `DimensionElementComponentAddDirect`:

²³ Note that `CubeClearData` will also clear feeders so those will have to be re-processed if applicable for whichever cubes may feed into the cube cleared using `CubeClearData()`

²⁴ See section <Use Parallel Data Load for re-loading data for very large cubes> below

4.3 Use TM1 functions DimensionElementInsertDirect and DimensionElementComponentAddDirect

Where meta-data and data processing need to occur within one and the same TI process, use the new functions DimensionElementInsertDirect and DimensionElementComponentAddDirect instead of DimensionElementInsert and DimensionElementComponentAdd. DimensionElementInsertDirect and DimensionElementComponentAddDirect can be used along with data update in the data tab of a TI process and hence will shorten processing time because metadata-tab processing can be avoided.

Note that for very large dimensions and subsequently very many *Direct updates, using the *Direct functions may cause a decrease in performance due to data being assigned to a very large number of elements which are not yet 'committed' to the dimensions. Such large dimensions should be updated using separate processes for

- a) Master-data and hierarchies and
- b) Attributes

and generally be separated from fact-data processing.

For fact data load processes where new elements (that may not have been captured by Masterdata update processing) will be added and assigned to an orphans node (and hence only a relatively small number of elements will be added using *Direct functions), we generally recommend leveraging the *Direct functions.²⁵

4.4 Use Parallel Data Load for re-loading data for very large cubes

In a parallel data-load regime, data is loaded into one or multiple cubes via separate & parallel processing threads. Using a parallel data load framework, a cube can be loaded using as many CPU threads as available and applicable, hence providing significant performance gains of data-load operations. Due to the very fast general load speeds of TM1 Turbo integrator (TI) processes, Parallel Data Load can be particularly useful when updating in scenarios where very large cubes are to be updated at a high frequency (for example, billions of facts to be updated daily).

For very large cubes, we recommend applying a TI framework methodology that will allow loading one or more cubes using parallel data loads, allowing the reload of cubes (with adjusted historical data for example) using parallel load threads and hence significantly speeding up load time. For example, if the load of one month of data takes 10 minutes and one wants to load/update data for an entire year, one can leverage parallel data load to load 12 data sets (one for each month) in parallel, hence still resulting in a total load time of only 10 minutes instead of 120 minutes.²⁶

Methods to initiate and perform a parallel data load include:

- a) Launching multiple TM1 Architect/Perspectives or Performance Modeler sessions manually and starting a data load process in each session,

²⁵ If used in a (fact)data load process, we recommend parameterizing code execution of DimensionElementInsertDirect & DimensionElementComponentAddDirect as a parameter of each data load TI process (for example via a parameter pUpdateMetadata Y/N).

²⁶ Using a parallel data load regime/framework, TM1 can process upwards of 50.000 records per second per CPU core (on high-performance CPUs at >3GHz). Using 16 CPUs for parallel data load, this can result in an overall data-load/update speed of roughly 48 Mio records per second, or 2.88 Billion records (2.880.000.000) in one hour, or 1.000.000.000 in just 21 minutes!

- b) Using [TM1RunTI](#) or [RunProcess](#) to trigger multiple data load processes built to allow parallel execution and data load,
- c) Setting up [chores](#) (one for each thread) that will automatically trigger a data load process based on an execution flag/indicator.
- d) Leveraging the scheduling & orchestration capabilities of [IBM Cognos Command Center](#) in combination with TM1,
- e) Using Python or equivalent, and TM1 Rest API calls (Note: packages like [TM1py](#) make using Python with the TM1 Ret API easy. TM1py is a Python package that wraps the TM1 REST API in a simple to use library)

All the above methods require that applicable TI process(es) or other procedures (Python code) used to load the data (the processes that are initiated in parallel) will adhere to the following guidelines, conditions & restrictions in section <Processing contention> below

For further information, please refer to [Proven practices and design template documentation](#), [code templates](#), [code templates with examples](#).

4.5 Minimizing Contention

4.5.1 Processing contention

- When performing a parallel data load, the zero-out / clear data operation should also be parallelized where applicable. Note that zero-out operations against large data volumes are also performance intensive. If a complete refresh/load is required, the cube should be cleared using the CubeClearData() Command (CubeClearData() is significantly faster than ZeroOutView() in clearing ALL data from a cube).
- Parallel data-load/update processes may not update meta- & master-data that is in a dependency-relationship to the cube(s) that are updated. Any data-update-related master- & metadata updates shall occur
 - in a single thread per dimension and
 - prior to initiating the parallel fact data load.

In other words: master-data for one dimension can at this point not be updated in parallel. A dimension update will lead to a lock of the dimension and through the lock will prevent a parallel data load/update from occurring. Dimension updates are hence to occur prior to the fact data update.

- Any subsets and views that are created in the context of the data-load TI processes need to be unique subsets and views (unique within the TM1 instance) to avoid contention during subset and view creation and deletion.
- Cube-Dependencies need to be (re)established where applicable prior to parallel data load. Please refer to [Understanding Cube Dependency](#) for more information on identifying and handling dependency locking.
- Note that TM1 and Planning Analytics will create cube read-locks or write-locks after dimension edits. Such locks will have to be 'removed' prior to initiating parallel processing:

Planning Analytics lock behaviour after dimension edits: When a dimension is edited, all cubes that use this dimension and that leverage cube rules need to be locked (a short lock) to re-evaluate rule validity and consistency. In TM1 10.2.2 this lock was an IX lock,

with PA, it is by default an IX lock (but the behaviour in PA can be changed for [ReduceCubeLockingOnDimensionUpdate=T](#), such that first an RO lock is issued and then evaluated for necessity of an IX lock).

- ⇒ when a dimension is edited, subsequent parallel processing against a cube with that same dimension will lead to a lock and prevent parallel processing (all threads except one in WAIT mode).
- ⇒ Solution to prevent/release such locks:
 - A) Put [ReduceCubeLockingOnDimensionUpdate=T](#) in place (tm1s.cfg). With this setting, the parallel threads will initially initiate a RO lock, allowing parallel processing to start. Only the first Parallel Process to finish/commit will then convert the RO lock into an IX lock and go into a WAIT state. In other words: [ReduceCubeLockingOnDimensionUpdate=T](#) will generally reduce locking
 - B) To remove the READ lock, a common practice is to create a view (and then destroy the view) for affected cubes (prior to parallel processing). The View-Creation will (re)establish cube validity and remove read locks, allowing parallel write-back against the cube.

4.5.2 Saving of Cube Data

Data that is loaded into a cube is not automatically committed to disk but retained in-memory only. Use the functions `CubeSaveData()`²⁷ to commit one cube's data and/or the `SaveDataAll()`²⁸ to commit all data to disk.

Note that `SaveDataAll()` and `CubeSaveData()` will acquire a write lock on the cube/database. It is hence not recommended to use `SaveDataAll()` and `CubeSaveData()` at the end of each TI load process (because that will cause lock-contention with other load processes that load data to any cube (`SaveDataAll()`) or also load data to the same cube (`CubeSaveData()`) in a parallel load scenario (see below).

There should be only one TI process that calls the `SaveDataAll()` function. Use a stand-alone, single, distinct chore to execute the `SaveDataAll` operation.

4.5.3 Subsets & Views

When using Subsets & Views in the context of TI processing,

- Create the Subsets & Views (target view for zero-out, source view) within the TI process²⁹. We do not recommend using pre-built subsets and views as those may be inadvertently changed. Furthermore, using pre-built views will violate the following 'rule':
- Use unique object names for dimension subsets and cube views used in the context of TI-processing. Any subset and view (and the subsets used by that view) that is used by a TI process should have a name that is unique to the user executing the process and the time the process is executed. Using subsets and views with unique names ensures that
 - The Subsets and Views can - if needed - be isolated and debugged without affecting other processing (be it by the same process or by different processes)
 - No contention will occur between different TI processes creating and/or destroying subsets and views at the same time
 - No contention will occur between the same TI processes creating and/or destroying subsets and views at the same time.

²⁷http://pic.dhe.ibm.com/infocenter/ctm1/v10r1m0/index.jsp?topic=%2Fcom.ibm.swg.ba.cognos.tm1_ref.10.1.0.doc%2Fr_tm1_ref_tifun_cubesavedata.html

²⁸http://pic.dhe.ibm.com/infocenter/ctm1/v10r1m0/topic/com.ibm.swg.ba.cognos.tm1_ref.10.1.0.doc/r_tm1_ref_tifun_savedataall.html

²⁹ Ideally via a 'generic', re-usable sub-process

- Tip: When building (temporary) system views and subsets, it is a good practice to include the TM1 Client name in the object (subset and view) name because this will allow for easier troubleshooting by allowing faster identification of the responsible client. Note however that when a chore is run, the TM1 Client Name identified by the TM1USER() function will include characters that are invalid for use in view or subset names (such as *). Similarly, CAM or Active Directory user names may include invalid characters such as '/' or '\'. Any such characters would need to be parsed out of the string returned by TM1USER(). A corresponding 'plug & play' utility to parse a valid user name out of TM1USER() is available per [here](#).

4.5.4 Synchronization & Serialization of applicable TI processes

For applicable TI-processes, we recommend the implementation of a Process Synchronization Framework to serialize TI processing to prevent thrashing/rollback: The concurrent execution of Turbo Integrator processes that lock the same objects and block each other can lead to successive rollbacks and retries, also known as thrashing:

If two or more TI processes may perform an update of one and the same dimension, the first process to acquire the dimension 'lock' will block the other process from continuing, the 2nd process will hence do a rollback of its actions before encountering the lock and then will attempt to start anew (possibly hitting the same lock later should process 1 not have finished). Such two processes should be kept from being executed in parallel and instead should only be allowed to be run in serial to avoid thrashing and excessive rollback actions. The same applies to fact updates and dimension updates: if you are running a process that updates fact data you should not attempt to update the dimension master data using a different process as this will cause locking and rollbacks.

A semaphore logic can be used to synchronize certain (applicable) TI processes to run in serial execution mode only. The synchronization logic will ensure that a process is told to 'wait' at the very beginning of execution (before it would require a roll-back or data updates etc.) until the 'lock' that would prevent the process from continuing is released.

As of TM1 10, the SYNCHRONIZED()³⁰ function allows for easy serialization of TI processes if needed.

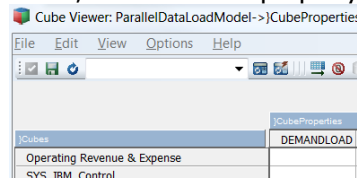
Recommended use of SYNCHRONIZED():

- i) For TI processes that are built specifically to updated a certain dimension, add the following code to the beginning of the TI process (TI prolog): SYNCHRONIZED (<DimensionName>);
- ii) For TI processes that are parameterized such that multiple or varying dimensions or objects could be accessed and locked, implementation of a dynamic Synchronization and Serialization Methodology and utility is recommended: Please click [here](#) for the utility documentation and [here](#) for corresponding TM1 components. The TM1 utility leverages the SYNCHRONIZED() function in combination with a configurable Lookup model to allow setting, maintaining and testing different serialization configurations in an ad-hoc/dynamic fashion.

³⁰ See [TM1 SYNCHRONIZED\(\) function](#) for details

5. Miscellaneous Recommendations and Guidelines

- a) If a cube performs poorly, first check if dimension sort order optimization may be applied and then check if cube rules & feeders can be performance tuned to improve performance.
- b) If a cube/model performs poorly under contention but fine for just one user, check for (re-)establishment of cube dependencies and other behavior that may cause temporary locking. Use a tool like Operations Console / TM1 Top / PAW monitoring and lock.exception debugging to test and monitor.
- c) To improve performance, set the MTQ parameter in the tm1s.cfg file to the highest # of CPUs that are available to the TM1 server process
- d) Only keep the needed history (# of years) in your main model/cubes (for example N years). Archive off old historical data by building an archive cube and loading historical data into the archive cube. Then zero-out the historical data from the Main cube models. Set the 'Archived' cubes to be loaded on demand only. DEMANDLOAD indicates if a cube is automatically loaded when the server starts or is loaded 'on demand' only when a cube value is requested. When a cube is loaded on demand, the value of the DemandLoad property is YES, otherwise the property value is NO.



- e) If data is frequently analyzed/queried in the same context by the same business users, attempt to keep/store/model the data in one cube as this will ease report as well as input template development. I.e. do not split Act and FCST data into separate cubes if Act vs. FCST analysis is frequently needed; do not split cubes into a Current Year and Prior Year cube if the two data sets frequently need to be compared for analysis etc.
- f) Create summary and detail cubes (with drill-through capabilities) if the detail data is not used frequently (see below: drill-through to transactional level detail) and/or only for separate detail analysis and/or the detail data is analyzed as part of different business processes.
- g) => only separate data (different cubes) if this makes sense from a logical or procedural standpoint. The focus should be on ease-of-use and user-friendliness.
- h) Use the same or similar 'visible'/'initial' dimension sort order for all cubes of a model. Dimension optimization shall be done independently for each cube to optimize the internal dimension sort order
- i) Let's say you have a drill-through detail cube that holds vast amounts of 'transactional' data and is only used for drill-through purposes, i.e. where the detail cube is used to get transactional detail for specific intersections of a summary level cube. A lot of the dimensions may be shared between the summary and the detail cube. At the same time you may – depending on HW capabilities etc. – want to prevent users from using the drill-through cube to perform high level aggregations (aggregations they could be doing on the summary cube). To improve performance in such a scenario, consider using separate 'dependent' dimensions for the detailed model rather than 'directly' sharing the 'master' dimensions from the summary cube. The 'dependent' dimensions can be derived from the

'master' dimensions, yet may have (i) different element weights such that a root level consolidation is not occurring (0 weights for descendants of the root element) and (ii) may limit the # of high-aggregation hierarchy levels. Example: while a Cost Center hierarchy in the overview cubes may and should allow consolidation of all cost centers along the cost center hierarchy, a 'dependent' cost center dimension used for very large detail cubes may be identical to the main cost center dimension in terms of its elements and hierarchies, but may not provide consolidation of cost centers at the lower branch levels. The same concept could be applied to a 'company' dimension: while the standard 'master' company dimension will have to allow consolidation across all companies, a 'dependent' company dimension may be built such that 'detail' data can only be viewed at the company level. Such dependent dimensions do not incur any significant maintenance overhead as they can be maintained automatically based on the 'master' dimensions. Note that security for such 'dependent' dimensions can be derived automatically based on the 'master' dimensions (because the 'dependent' dimension contains only elements from the 'master' dimension).

Please contact IBM for a corresponding TM1 asset for maintenance of master-dimensions and dependent dimensions – including security.

- j) For TM1/PA with Cognos BI and Cognos Analytics, please refer to the paper '[Planning Analytics with Cognos Analytics - Introductory Guidelines and Proven practices](#)'.