IBM TRIRIGA Application Platform
5.0

Best Practices for System Performance

Note

Before using this information and the product it supports, read the information in “Notices” on page 113.

This edition applies to version 5, release 0, of IBM TRIRIGA Application Platform and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Planning for optimized performance

Use these best practices to improve the performance of applications on the IBM TRIRIGA Application Platform.

Optimizing performance in your environment

These best practices provide optimal performance in the lab test environment, but your environment might require different settings. Use these settings as a guideline to monitor and tune to your IBM® TRIRIGA® environment.

Because customers who deploy TRIRIGA across large, global enterprises face some of the greatest challenges, this document focuses on improving performance in advanced enterprise configurations.

Virtualized environments

Deployments in virtualized environments such as VMware might not give the same performance benefits from these best practices. Using shared resources in virtualized environments does not provide optimal performance. Use dedicated CPU and memory resources instead.

Also, internal testing shows that putting the TRIRIGA database in a virtualized environment such as VMware can have significant performance impacts. These impacts increase rapidly with larger workloads.

For more information, see *Database server virtualization* in Database server tuning and maintenance.

Support for performance-related problems

Performance-related problems are often the result of environmental factors and outside the scope of support. Although IBM makes every effort to assist clients in understanding the cause of performance issues, performance-related issues are not officially something support can resolve. Understand the contents of this best practices document and recognize that failure to adopt the recommendations in this document prevent further investigation by support in resolving any performance-related concerns.

You need a good technical understanding of TRIRIGA to understand how TRIRIGA interacts with technology. The Java™ code in TRIRIGA is called the TRIRIGA Application Platform. This code supports the associated technologies such as databases, application servers, and browsers that are supported for each version. When the application platform is developed, the platform is used to develop TRIRIGA applications including Lease Management, Space Management, Reservations, Operations and Maintenance, and Project Management. These applications are stored as metadata objects that take advantage of the functionality that is built into the development platform. This metadata that creates applications is stored in the database.

To understand performance tuning with TRIRIGA, you must understand that there are core technology tuning tips that are related to the platform, and there are application-specific tuning tips that are related to the metadata coding that creates the applications. When the network, hardware, and middleware are optimally tuned, the next step is to tune the platform and finally the application. If clients customize workflows or extended formulas, IBM TRIRIGA Support focuses first on configurations that might change the use of database indexing, SQL queries, formula usage, data flow, or other aspects that are not consistent with best practices and are not tested. If custom configurations are identified as a cause of any issue in product functionality or performance, a developer, business partner, or IBM services engagement might be required to resolve the issue.

The scope of these best practices applies to all versions of IBM TRIRIGA 10 and 11 and all versions of TRIRIGA Application Platform 3 and 4, which are collectively referred to as TRIRIGA in the remainder of these best practices.

Identifying TRIRIGA versions

The platform version is the smaller number of the two numbers that represent TRIRIGA. The 5.x.x is the platform version and the 11.x.x is the application version. IBM uses a standard of Version, Release, Fix Pack (VRF) as its numbering system so that version 4, release 5, fix pack 3, is 4.5.3. The application version numbers are seven digits larger than platform versions. While many of the recommendations in this document also apply to earlier versions of the TRIRIGA platform and applications, this document is intended specifically for the latest versions.

The platform is always designed to be backward-compatible with applications that were previously developed. For example, an application is developed by using platform 4.5.0. Upgrading the platform to 5.0.0 maintains compatibility with the older application.

Product documentation

For more information about TRIRIGA, see the IBM TRIRIGA product documentation and IBM TRIRIGA Application Platform product documentation in IBM Documentation. Helpful information might be found in the latest Release Notes for your installed IBM TRIRIGA product version, as they might contain specific information that overrides topics in this document. In addition to the release notes, thoroughly review the Installation and Implementation Guide. For information about IBM TRIRIGA supported products and platforms, see the IBM TRIRIGA Supported Versions and IBM TRIRIGA Application Platform Compatibility Matrix.

Best practices as a cooperative effort

This information is the result of a cooperative effort between IBM employees, IBM Business Partners, and a core group of enterprise customers of TRIRIGA. This information was developed both in response to customers and in concert with them. The cooperative effort is ongoing. The goal of these best practices is to provide you with information that can help you to improve the experience of system performance for your users.

Quality assurance and testing

Each release of TRIRIGA provides new features for your users. IBM also strives to increase performance, reliability, and speed with each release. One way that IBM enhances performance is to provide tools and features that are designed to improve system performance.

The IBM Quality Assurance (QA) team tests TRIRIGA. Testing is performed during the development cycle and again before the product is released. Testing continues after TRIRIGA is released. Application and system functionality testing is performed as is system performance testing. The quality assurance team tries to emulate many of the system setups that you might use. However, a laboratory testing environment can never fully anticipate all the scenarios that the product encounters in your environments. The testing environment undergoes constant review to determine how best it can be scaled to meet the demands of large enterprise deployments.

Customer involvement and feedback are vital to improving quality assurance testing. You can provide feedback for these best practices in IBM TRIRIGA Chat.

Factors in system performance

The system performance depends on more than the applications and database. The network architecture affects performance. The application server configuration can impair or improve performance. The way that you deploy TRIRIGA across servers affects the way that the products perform. Many other factors are involved in the user experience of system performance.

This document addresses the following topics:

- System architecture setup
- Application server configuration
- Reporting

- Integration with other systems by using the integration framework
- Network issues
- Bandwidth
- Load balancing
- Database tuning and SQL tuning
- Client workstation configuration
- Miscellaneous performance improvement tips
- Platform tuning
- Application configuration and customization
- Troubleshooting

Performance testing and tuning in implementation project plans

When planning for a TRIRIGA implementation, schedule the following phases:

- Performance testing
- Performance tuning
- Contingency iterations of testing and tuning for issues that are discovered
- Fallback plans to mitigate the risk if performance testing uncovers issues

As a part of project management, certified Project Management Professionals (PMs) must be fully versed in project risk management. Project managers should develop a series of actions in case identified risks occur. Performance testing and tuning are inherent risks in any implementation. Poor preparation can lead to unrealistic deadlines and even project failure. It is important to add ample time to perform multiple iterations of performance testing and tuning and explain to the stakeholders what the risks are, in terms of project deadlines, if they do not accept that multiple iterations and extra time might be needed to tune the system and application.

It is also important to understand the difference between a contingency plan and a fallback plan. A contingency plan is executed only when identified, accepted risk events occur and is developed during the plan risk responses process. A fallback plan is developed to deal with risks if the primary contingency plan is not effective for the known identified risks.

For example, if, in the performance testing phase, a performance issue is identified with a particular process is uncovered. The tuning phase investigates the performance issue by examining the performance logs generated by running the process and analyzing the information for where the bottleneck or hot spot is. When the analysis is done, the application metadata can be updated, following best practices for application performance. If an application cannot be easily updated, then a contingency plan is necessary. A larger rearchitecting of the application would be done to both deliver the business requirements and solve the performance issues. The fallback plan is needed if the rearchitecting fails to improve performance, and the initial assumptions of the system size are brought into question, and new faster hardware is required to deliver acceptable performance.

Infrastructure pyramid

Visualize TRIRIGA at the top of a pyramid that consists of network, hardware, and software services. When troubleshooting any performance issue, it is crucial to consider the entirety of the stack as potential variables that must be diagnosed. If the top is tuned without consideration for the reliant levels below it, performance problems might continue. The pyramid comprises the following infrastructure within which TRIRIGA operates:

- TRIRIGA runs within the layer of software services. The entire system must be sufficiently robust and well-tuned.
- Software services such as application servers, report servers, and database servers run on the operating systems.

- Operating systems run on the hardware servers.
- Hardware servers operate within the network.
- Network architecture supports the entire system infrastructure.

Improving system performance requires attention to all elements of the infrastructure. Your databases can be tuned perfectly, but if the network is slow, you might still have system performance issues. Or, you might have a fast local area network, but if client workstations lack memory and processing power, users experience poor performance.

Customers report that there is normally more than one issue to address. Improving performance involves adjusting many things. Together, these changes can combine to yield significant performance improvement. Making improvements in one or two areas can boost performance. For example, upgrading user workstations can often greatly improve the application experience, even if other problems exist in the overall deployed environment. As you continue to improve different areas of the environment, system performance improves incrementally.

Figure 1 shows the interdependence of TRIRIGA and other elements in the system infrastructure pyramid. These best practices discuss performance considerations starting at the base of the pyramid and moving up to the top with specific recommendations for the TRIRIGA product.

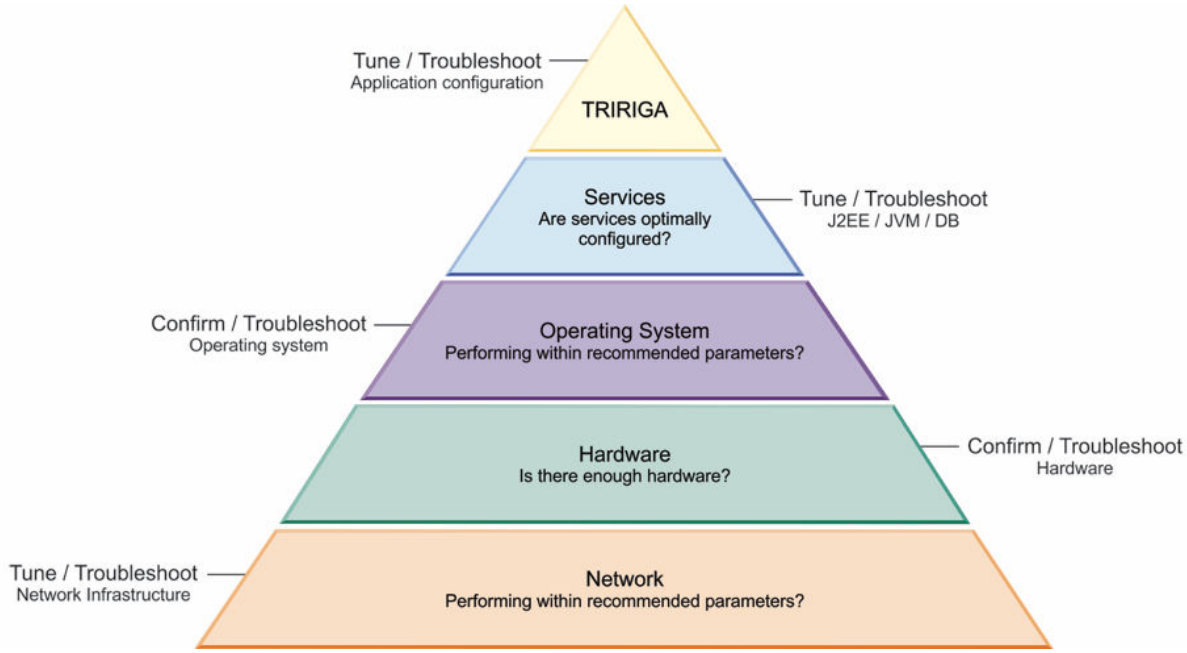

Figure 1. Infrastructure pyramid

Chapter 2. Tuning the network

Clients connect to the application over the network. The application also communicates with its various parts, such as the application server, database, and report server, over the network. If any segment of the network performs poorly, the user experiences a system that is slow and hard to navigate.

Overview

TRIRIGA is a web-based product that operates on a request and response basis. If the requests and responses are delivered slowly, TRIRIGA has no control over response time. If users complain that TRIRIGA "feels slow", as opposed to one particular process being slow, the network is often the root cause and must be investigated first.

Network speed throughput test

Sometimes, you need to understand the network speed throughput from different client locations. In the IBM TRIRIGA Administrator Console, the Performance Monitor manager tests different client locations and compares network speed throughputs. The network **Speed Throughput Test** tests the network speed from the server to the current user desktop. In the results, you can see your speed throughput in KB and an average comparison with other network structures. Information about upload and download speeds, speed to generate the data, and speed to upload and download to the database are included.

You can provide the network **Speed Throughput Test** to client users without the need for a login. For adequate response times, users must be able to obtain network throughput of at least 229 KB. Users might experience performance degradation if the network does not operate within these parameters. Assign a network engineer or IT administrator to investigate network traffic speed from that client to the server. Also, test the TRIRIGA server directly by circumventing any web servers and proxies to confirm if network speed improves.

Often, intermediary proxies contribute to poor network speeds. For example, if TRIRIGA is deployed to WebSphere Application Server Liberty on port 9080, the direct URL removing redirects, proxies, load balancers, web servers, firewalls and more might be `http://[webspherereserver.com]:9080/`

Direct URL:

The port number is the port that TRIRIGA is deployed to.

You might also have a configured context root which is after the port number and slash (/) of the URL. A context root is the direct connection to the application and can help determine whether the performance concern is related to the deployment infrastructure or the network infrastructure. Perform the test as close to the application server as possible and on the same network segment.

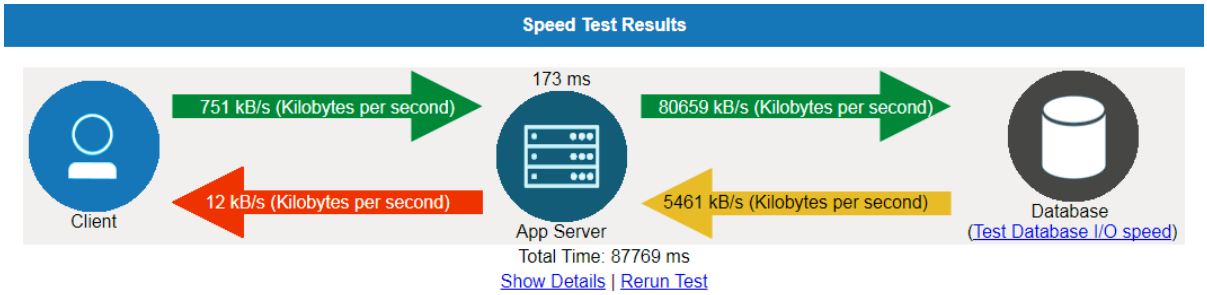

Figure 2. Network speed throughput test

Using Citrix or Windows terminal service

Some options for resolving low-bandwidth or high-latency network performance issues include services such as Windows Terminal Server and Citrix. These services can provide maximum performance between

the Windows Terminal Server or the Citrix client and the application server with a minimum traffic between the Windows Terminal Server or the Citrix server and the user.

A benefit of running TRIRIGA through Citrix is that Citrix traffic is treated as *business traffic*. In some customer environments, business traffic can take priority over nonbusiness traffic. Some customers get considerable improvements in network performance when they use a Citrix or Windows Terminal Server. IBM does not test or certify Citrix or other bandwidth tools.

You can use network caching, acceleration, and compression utilities to improve network performance.

Using compression techniques to improve performance

Use one of the following techniques to improve bandwidth or latency issues:

- HTTP compression
- Hardware compression

These options are mutually exclusive. If you are using hardware compression, then do not also use HTTP compression.

HTTP Compression

HTTP compression is a capability that can be built into web servers, such as IBM HTTP Server (IHS) and Apache HTTP Server (AHS) 2.x and later, and web browsers to make better use of available bandwidth, and to provide faster transmission speeds. HTTP compression affects all servers in a cluster. HTTP data is compressed before it is sent from the server. Hardware load balancers that bypass the HTTP Server cannot employ this method.

Compression-compliant browsers announce to the server which compression methods the browser supports, so the server can provide the compressed data in the correct format. Browsers that do not support compression download decompressed data. Data can be compressed by using a compression module such as Apache `mod_deflate`. The server software decides the compression scenario.

The following compression scenario is a good solution for low-bandwidth locations:

IBM HTTP Server
: Use Apache `mod_deflate` and set `DeflateCompressionLevel` to 3 to improve response time in environments that have low bandwidth and high latency. For information about configuring `mod_deflate`, see Apache Module mod_deflate.

Hardware compression by using network appliances

Network appliances, such as routers, provide compression and caching features. Network appliances can help compress data and optimize bandwidth. Customers report that network appliances can prove to be beneficial to system performance, especially in a high-latency or a low-bandwidth environment.

Hardware compression affects all servers in the configuration.

Chapter 3. Tuning system architecture and hardware

Maximize system performance by tuning your system architecture to account for your resource needs.

Overview

When sizing your TRIRIGA system, you must include a performance test phase as part of the implementation. Sample starting configurations are provided, but you must conduct performance testing analysis and tuning to reach the best possible configuration for your environmental and application circumstances before you sign off on the final production configuration.

For more details on system architecture and hardware configuration, see the IBM TRIRIGA Application Platform: Installation and Implementation Guide.

Planning capacity

As with any web-based platform, the application server and database capacity that is needed to deploy the TRIRIGA depends on the number of anticipated users and user requests.

At a minimum, the server capacity to satisfy the average load during a work day is required, with response times that are acceptable to the user base. If possible, aim to satisfy the volume of requests that is anticipated during peak intervals of high user activity. Hardware resources such as CPU, memory, I/O capacity, and network bandwidth are key to reducing response times. Unless you install TRIRIGA on a server or group of servers that can handle many transactions, users might experience slow response times.

Adding more servers and database capacity might improve performance, but that is not always the case. The application usage and system configuration can play a significant role in the balance of system performance.

Estimating sizing

Estimating the TRIRIGA system size is complex and error-prone, which is why it's an estimation and not a calculation. Use the following estimation methods to size a TRIRIGA implementation:

- Size by Example (SBE)
- Proof of Concept (POC)

SBE

An SBE approach requires a set of known samples to use as data points along the range of system sizes. By making more examples available for SBE, the intended implementation is more accurate. Several targeted SBE sizing solutions for prospective TRIRIGA customers are provided. These targeted solutions were compiled from internal deployments, performance benchmarks, and customers' external deployments.

POC

A POC or pilot-based approach offers the most accurate sizing data of all approaches and includes the following process steps:

- Test your implementation design.
- Test your chosen hardware platform.
- Simulate the projected load.
- Validate design assumptions.
- Validate IBM TRIRIGA.
- Provide iterative feedback for your implementation team.

- Adjust or validate the implementation decisions that are made before the POC.

However, two disadvantages of a POC are time and money. Running a POC requires personnel, hardware, and time to implement the solution, validate the solution, iterate changes, retest, and finally analyze the POC findings. However, a POC is always the best approach for any sizing exercise. It delivers results that are accurate for the unique implementation of the specific customer and that are as close to deploying the real live solution as possible, without the full capital spending on hardware and project resources.

Scalability

Many factors can affect the scalability of TRIRIGA. For example, the type of hardware that is used can determine how many active users per Java virtual machine (JVM) you can achieve.

The average workload that is applied can also have a large impact on the system. For example, you can support more users per JVM if those users are producing five transactions per hour instead of ten transactions per hour. In addition, heavily customizing TRIRIGA can result in fewer users per JVM because of the overhead in processing custom code.

Lastly, the architecture that is used in deploying TRIRIGA has a significant impact on the number of active users per JVM. Some clients achieve only 50-100 concurrent users per JVM when all transactions are running in the JVMs. Creating separate JVMs for user interface transactions allows for more active users per JVM. In an advanced system configuration, internal testing shows the ability to support up to 2000 concurrent users per user interface JVM.

Development system configuration

A development environment is typically used as a sandbox for developers to make and test changes. The sandbox environment is not built for performance or for many users. That environment is also known as a basic system configuration that typically has less than 20 concurrent users.

A typical configuration for this type of environment would have 1 or 2 tiers depending only on the database requirements. A single application tier or server is common to serve for multiple logical roles in this case, for example, application, process and Business Intelligence and Reporting Tools (BIRT) server.

The following hardware requirements are typical for the application tier that is used for this configuration:

- 2 to 4 CPU logical cores/threads
- 4 GB to 8 GB of memory with 4 GB allocated to the Java heap

Basic system configuration

A basic environment might be a configuration for production with less than 100 concurrent users or a staging environment that is not used for development work. This type of environment might be configured for production performance or for production function testing. This configuration is also known as a basic system configuration that typically has less than 100 concurrent users.

A basic configuration consists of TRIRIGA running on a single application server. The application server connects to a single instance of the database that is available on a database server. The application server can also connect to a report server.

Do not use the basic system configuration if you anticipate using IBM TRIRIGA Connector for Business Applications (Web Services). TRIRIGA applications that use IBM TRIRIGA Connector for Business Applications or other connector technology might require more server resources and therefore must not be used in a production environment if the basic system configuration is implemented.

Examples of connector-based products are TRIRIGA CAD Integrator, Reservation Management with Microsoft Exchange, ENERGY STAR Connector, Connector for ESRI/GIS and the TRIRIGA Integration Object.

A typical configuration for this type of environment has two tiers, one for the application server and one for the database. A single application tier is common to serve for multiple logical roles, for example, application server, and BIRT server. If given the choice of two different types of servers, the more

powerful server must be the database server. The most common way to improve the performance of a TRIRIGA system is to increase the CPU and memory of the database. The database server is the single most important server across the tiers.

The following hardware requirements are typical for the application tier that is used for this configuration:

- 2 to 4 CPU logical cores or threads
- 4 GB to 8 GB of memory with 4 GB allocated to the Java heap

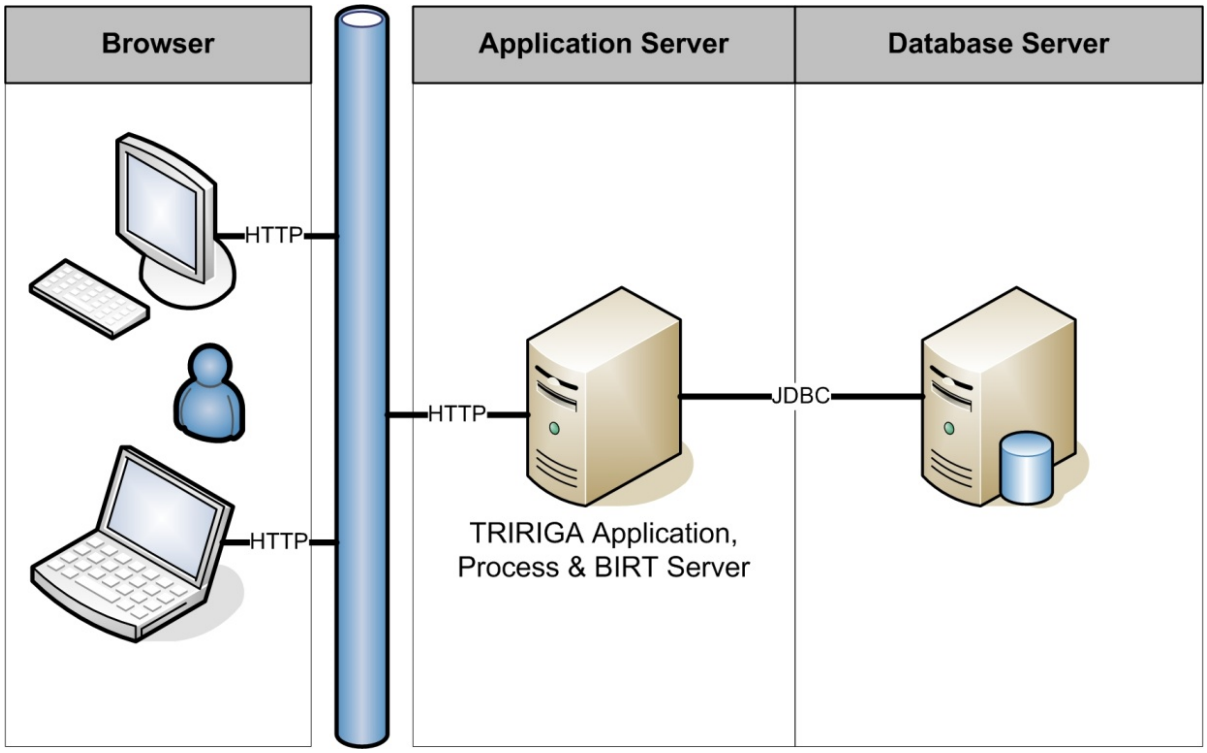

Figure 3. Example basic system configuration

Typical system configuration

A typical or medium environment might be a configuration for production with fewer than 1000 concurrent users or a staging environment that has fewer users than a production environment and does not have redundancy or high availability (HA) requirements.

A typical configuration for this type of environment might have 2 to 3 tiers, depending on infrastructure and security requirements. A web tier might be added for a DMZ requirement, but the typical configuration is to have one tier for the application servers and one for the database. The most common way to improve the performance of a TRIRIGA system is to increase the CPU and memory of the database. The database server is the single most important server across the tiers.

You can also improve performance by using separate JVMs for the logical application and process servers. TRIRIGA application and process servers have the same installation and the same Java Application Server requirements. They serve different roles for the system. The application server, single or multiple, is for user sessions and must be dedicated only to users with no background processes. The process server, single or multiple, must be dedicated to background processes, such as agents, integrations, and advanced reporting processing.

The following hardware requirements are typical for the application tier that is used for this configuration:

- 4 to 8 CPU logical cores or threads
- 8 GB to 16 GB of memory with 4 GB allocated to each Java heap
- 50 GB disk drive

The following hardware requirements are typical for the database tier that is used for this configuration:

- 8 to 16 CPU logical cores/threads
- 16 GB to 32 GB of memory
- 500 GB of fast Fibre Channel with15K RPM or faster disk or solid-state drives

Figure 2 shows a typical configuration to support a system that has fewer than 1000 concurrent users and no advanced system needs such as high availability (HA).

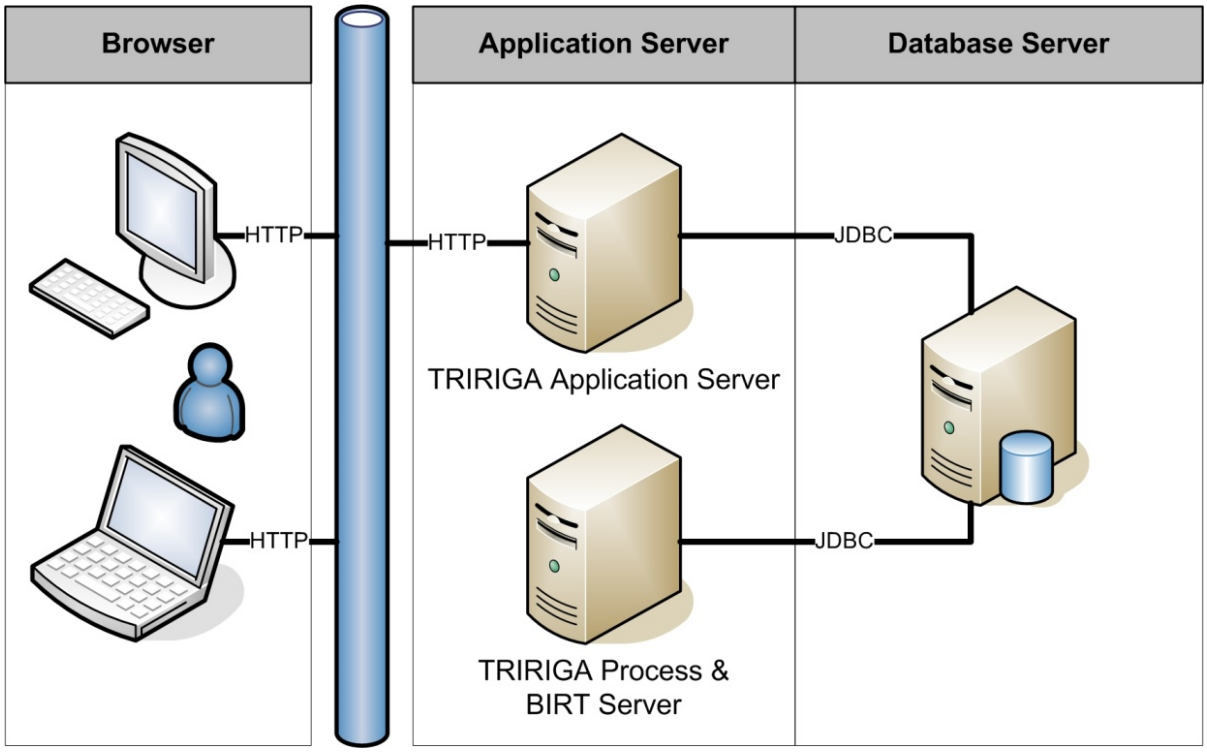

Figure 4. Example of a typical system configuration

Advanced system configuration

An advanced or large environment might be a configuration for production with up to several thousand concurrent users and when redundancy or high availability (HA) requirements are crucial to business needs.

An advanced system configuration might have one or more of the following requirements:

- HA
- Load balancing integrations, or
- Many concurrent users

This type of advanced system might be as simple as two application servers with a cold failover-clustered database system or as complicated as multiple servers at all tiers. Implementations of TRIRIGA for advanced system needs can be achieved in many ways depending on the requirements of your system. There are no specific requirements for the TRIRIGA Application Platform to achieve these types of advanced requirements.

HA requirements can be configured many ways. Hardware-based or software-based load balancers can be used in front of multiple web servers to communicate with multiple application servers. With the use of the IBM TRIRIGA Administrator Console, the logical roles of the servers can be changed dynamically at any time by turning background agents on or off with any server console.

Continually monitoring and evaluating TRIRIGA products for performance and using horizontal and vertical scaling is an effective way to distribute user load, improve overall system performance, and ensure optimal user experience. With WebSphere® Liberty Collectives, HTTP Session Failover can be used for both high availability and load balancing. Hardware-based or software-based load balancers can

distribute workloads among multiple web servers to communicate with multiple application servers. With load-balanced web servers, use session affinity. The client's request is always passed through the same server that originated the request, and the overhead that is associated with replicating session data can be eliminated.

The number and type of servers that are used in a high availability environment depends mostly on application usage patterns and production transaction volumes. The number and size of JV Ms that are configured depends in part on the overall hardware and software limitations of the environment and on the expected number of concurrent users. You can tune the setup based on traffic and performance numbers. For example, in situations in which many web services or integrations are exercised using APIs, a dedicated TRIRIGA integration server might be needed to maintain system performance. However, the database server must still be the largest and most important server in the environment. One application server might easily overload the database server.

The following hardware requirements are typical for the application tier that is used for this configuration:

- 8 to16 CPU logical cores/threads
- 16 GB to 32 GB of memory
- 50 GB disk drive

The following hardware requirements are typical for the database tier that is used for this configuration:

- 32 to 64 CPU logical cores or threads
- 64 GB to 256 GB of memory
- 2 TB to 4 TB of fast Fibre Channel SAN storage with 15K RPM or faster disk or solid-state drives

Figure 3 shows an example of a complex enterprise architecture that is required to support up to several thousand concurrent users when redundancy or HA requirements are crucial to business needs.

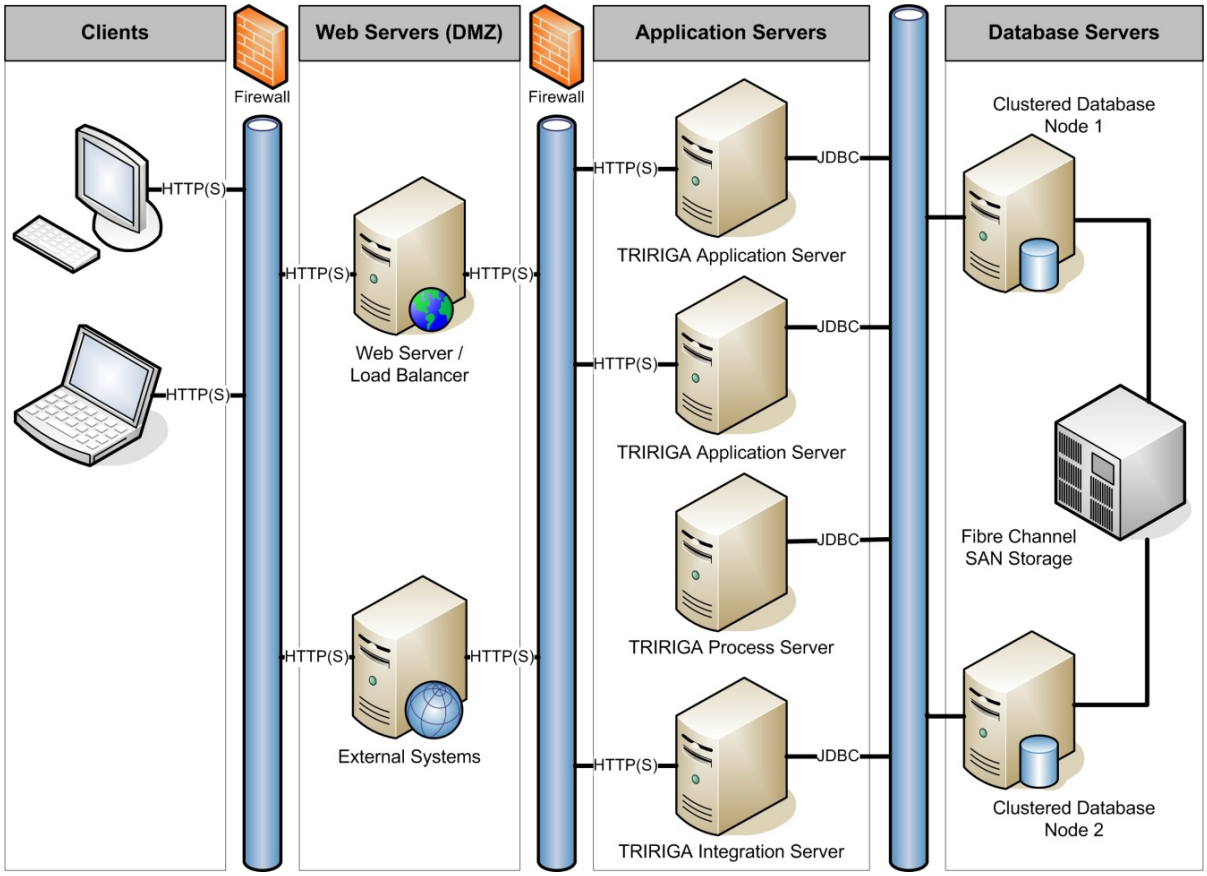

Figure 5. Example of advanced system configuration

Chapter 4. Tuning the operating system

Tune at the operating system (OS) level on servers and clients to optimize TRIRIGA performance.

Server operating system

Tune at the OS-level on servers to optimize performance. These settings optimize performance. However, consult with your systems and network administrators before you use them to ensure that they are compatible with your organization's standards.

Linux®

Default settings for tuning TCP and UDP on Linux are sufficient for TRIRIGA. However, modify the following settings for best performance:

For networking, use the following settings:

```
sysctl -w net.ipv4.ip_local_port_range="1024 65535"
```

For resources, use the following settings:

```
max user processes (-u) 8192
open files         (-n) 131072
```

For shared memory on Db2 servers, see Modifying kernel parameters (Linux).

For shared memory on Oracle servers, see Configuring Kernel Parameters and Resource Limits.

Windows

Set the following networking parameters, which are located in the Windows registry key:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`

```
TcpTimedWaitDelay         dword:0000001e (30)
StrictTimeWaitSeqCheck    dword:00000001 (1)
MaxFreeTcbs               dword:00011940 (72000)
MaxHashTableSize          dword:0000ffff (65535) T
cpWindowSize              dword:0000ffff (65535)
```

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces`

```
TcpAckFrequency           dword:00000001 (1)
```

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters`

```
EnableDynamicBacklog      dword:00000001 (1)
MinimumDynamicBacklog     dword:00000032 (50)
MaximumDynamicBacklog     dword:000003e8 (1000)
DynamicBacklogGrowthDelta dword:0000000a (10)
```

If some of these parameters are not in the registry, add them. For more information, see the Microsoft technical documentation.

For Windows Server, the default start port is 49152, and the default end port is 65535. Therefore, 16384 ports are available by default.

To view the dynamic port range, start the command prompt and use the `netsh` command:

```
netsh int ipv4 show dynamicport tcp
```

To change the dynamic port range for the maximum number of ports allowed, issue the following command:

```
netsh int ipv4 set dynamicport tcp start=1025 num=64510
```

The minimum start port is 1025, and the maximum end port cannot exceed 65535.

Client operating system

Tune at the OS-level on clients to optimize performance. Customers report that client workstation configuration is, initially, the most important area to focus on when users experience performance issues.

Hardware and software products constantly evolve. TRIRIGA can run on some older hardware and software platforms. The best performance is achieved by using the newest supported operating systems and the latest hardware. Robust workstations provide better performance. For example, client workstations must have at least the minimum RAM that is required by the operating system and by all other applications that it runs, but adding more RAM can boost performance. Similarly, using workstations with multiple CPUs and higher clock speeds also boosts performance.

- Limit or prevent some workstation activities and processes. Customers report that some user activities and workstation processes can degrade performance. Check for and monitor these activities and processes and respond as necessary.
- Monitor the network for streaming audio and video. Customers report that monitoring the network to prevent users from using streaming audio and video can noticeably increase the bandwidth that is available to the system.
- Have only one network link active. Users with both wired and wireless network links can cause system performance issues. Limit users to one active network link.
- Monitor for nonresponsive processes and applications. Processes and applications that are not responding use memory and can affect the performance of the client workstation. A workstation can have nonresponsive processes or applications that the user might not be aware of. Use system tools, such as the Windows Task Manager, to check for and end any nonresponsive processes and applications.

For more information on supported desktop client workstations, see the IBM TRIRIGA Compatibility Matrix.

Chapter 5. Tuning the database server

Tune your database server for optimal performance.

Overview

Your database performs the following functions:

- Stores all the data that the applications collect and calculate.
- Stores metadata for configuring and maintaining the database environment.
- Processes all transactions from the applications.
- Accesses the data in the database to generate documents.
- Generates resource-intensive management reports.

Co-location

To improve performance, co-locate the database server on the same subnet as the application and process servers. Do not use cloud database offerings such as IBM Db2 on Cloud Service or Oracle Database Exadata Cloud Service. However, hosting the entire TRIRIGA infrastructure in the cloud where the infrastructure is hosted on the same subnet is a viable configuration for good performance. In addition, customers report good performance when they use Oracle Exadata Database Machine in on-premises deployments.

Always check the IBM TRIRIGA Compatibility Matrix for the latest database versions and fix packs that are supported. Performance issues where the database server is not colocated with the application and process servers are not supported.

Converting IBS_SPEC_ASSIGNMENTS to Module Level Associations

TRIRIGA stores all associations between records in the IBS_SPEC_ASSIGNMENTS table. To improve performance, consider converting the IBS_SPEC_ASSIGNMENTS table to Module Level Associations (MLA) tables. MLA tables are smaller and therefore improve performance. MLA tables also reduce database contention in comparison to the IBS_SPEC_ASSIGNMENTS table. For more information, see Converting IBS_SPEC_ASSIGNMENTS to Module Level Associations.

Standard tuning techniques

Apply standard database tuning techniques and periodically monitor production databases during peak load. You can use the standard monitoring tools such as the tools on the database platform. If necessary, adjust parameters to resolve the bottlenecks as suggested by the monitoring tools.

Tuning the database

Use indexing and query tuning to tune your database for optimal performance.

Indexing

Indexes use key parts of data from a table in a binary structure to enhance searching capability. Each record in the table has data that is associated in the index. Regularly evaluate the following opportunities for indexing:

- Analyze the TRIRIGA logs.
- Use the Performance Analyzer for long-running queries.
- Use database tools such as Oracle AWR Reports and Db2 snapshots to identify long-running queries.

When you identify long-running SQL queries, use the database platform's query tuning utilities to identify potential indexes to improve the query response time. Oracle SQL Developer, IBM Data Studio, and Microsoft SQL Server Management Studio each provide SQL tuning utilities. These utilities operate similarly. You input the SQL query and select the option to tune the query. The utility then lists any potential indexes that can be created to improve query performance. For more information, see the documentation for your database platform.

Indexing increases search speeds. However, a drawback of indexes is that for each insert, update, or delete, the index must also be updated. Database administrators often apply many indexes to a table to enhance searching and then sometimes find that other activities are slower. Review and test all potential indexes to ensure that you have the optimal balance for searching and for updating tables.

TRIRIGA includes several built-in indexes against IBS_SPEC_ASSIGNMENTS. However, you must analyze IBS_SPEC_ASSIGNMENTS indexes to ensure that the indexes improve performance. Do not add indexes to IBS_SPEC_ASSIGNMENTS because they can cause issues in TRIRIGA.

Indexes depend on their configuration and data composition. Analyze the configurations and data for indexes that might be helpful for your database configuration. The indexes that are included with TRIRIGA help in most use cases.

Also, check existing indexes against new indexes to make sure that there is no overlap. For example, a new index might recommend an index on columns A, B, and C. However, there might be an existing index on columns A and B. Instead of having multiple indexes, modify the existing index to add column C after column B.

Customizing TRIRIGA can change the way that you select information from the database. Some customizations include more tables and columns. If you customize TRIRIGA, compare indexes to the user functions that use them. You might need to alter existing indexes to include new columns and create new indexes for new tables.

Use the Database Table Manager tool to view, alter, drop, and create database table indexes because they are then managed by the product. For more information, see Database tables.

Analyze the IBS_SPEC_ASSIGNMENTS table to understand your data

Module Level Associations (MLA): If your database is converted to enable Module Level Associations, the ALLOW_REVERSE_ASSOCIATION property is not supported. The **Reverse Association** flag in queries and reports is ignored, and only forward associations are allowed in queries and reports.

TRIRIGA stores all associations between records in the IBS_SPEC_ASSIGNMENTS table. Over time, this table can exceed 100 million rows. Processes like the Platform Maintenance Scheduler, which was named Cleanup Agent, take longer to finish because it queries this table to clean associated data. Analyze your IBS_SPEC_ASSIGNMENTS table to identify data in your system that you can delete to reduce the size of your table.

Performance tuning indexes

Tune those indexes that are the result of iterative performance tuning cycles. The indexes that are listed are not included in the TRIRIGA base product unless otherwise stated. For more information, see:

- Section *Application platform indexes* in “Tuning the IBM Db2 indexes” on page 23.
- Section *Reserve platform indexes* in “Tuning the IBM Db2 indexes” on page 23.
- Section *Application platform indexes* in Tune the Oracle Database.
- Section *Reserve platform indexes* in Tune the Oracle Database.
- Section *Application platform indexes* in Tune the Microsoft SQL Server indexes.
- Section *Reserve platform indexes* in Tune the Microsoft SQL Server indexes.

These indexes provide performance improvements when measured against a broad performance test workload. Add these indexes based on their respective database platform. However, performance gains might vary depending on factors such as application usage, load patterns, hardware sizing, application,

and database server configuration. Database administrators must monitor databases for efficient index usage to determine the overall impact that is produced by applying the indexes and to determine more indexes that improve performance based on situational and data composition needs.

Tuning queries

Review custom queries and reports for efficient SQL and use of indexes. In all of these cases, improving the efficiency of user queries also improves the efficiency of the workflows that use them.

Many long-running queries and reports are often improperly filtered. In addition, most of the filters in reports and queries are generated by users with run-time filter operators and by association filter operators that are defined in the Report Manager. While these filters are powerful features of TRIRIGA, they can produce inefficient SQL. For more information, see *Query tuning* in "Query and report performance" on page 87.

Maintaining the database

Maintain your database by, for example, reorganizing tables and indexes.

Database statistics

To ensure that the database is using the optimal path to access data, periodically generate and update statistics on all the tables and indexes.

Configure the Platform Maintenance Scheduler, which was formerly named Cleanup Agent, correctly to ensure that the statistics are updated daily. regularly analyze the server log on the server that runs the Platform Maintenance Scheduler. Analysis verifies that the agent is completing successfully and running the appropriate database statistics command. If you do not use the Platform Maintenance Scheduler for this task, then schedule the statistics update to run on the database daily or weekly, depending on the system that is used.

IBM Db2 provides the **runstats** tool to update statistics directly on the database because the Platform Maintenance Scheduler does not automatically call **runstats** on that database platform.

Oracle provides equivalent commands for manually updating statistics on the database:

- **dbms_stats.gather_schema_stats**
- **dbms_stats.gather_table_stats**
- **dbms_stats.gather_index_stats**

Reorganizing tables and indexes

If you do not expect further growth in a table, reorganize it to reclaim space. When many updates are made on a table, the space can become fragmented. However, if you expect further inserts, then the database should reuse the space within the table.

Reorganizations or rebuilds can also improve performance. When the data is not aligned on an index, performance can become degraded.

Consider these questions before you reorganize or rebuild tables an indexes:

- Does fragmented space cause a performance problem?
- Are the records in the table updated or are new records inserted within a short period?
- Does fragmented space cause inefficient storage use?
- Is the order of the table data aligned with a particular index?
- Has performance degraded over time or has it happened at a specific time?
- What changes might cause the incident?
- How does the performance correlate to system CPU and I/O use?

- If table reorganization is done, what is the expected benefit? Running reorganization affects data availability, uses system resources, and takes time to complete.

Reorganize and rebuild the tables based on the following guidance:

- Reorganize a table only when you carefully analyze to determine that a reorganization corrects the performance problem.
- Reorganize tables that have many deletes and updates to reclaim space only when you determine that no further inserts are done in the near term.
- Do not reorganize metadata tables.
- If you determine that a reorganization improves a performance issue with the table or index, then consider whether to reorganize online or offline while the system is down. System resources are needed for reorganizations. A reorganization can slow down the system. Also, consider that users must wait for locks during an online reorganization.
- Maintain the size of the database by archiving and trimming tables where appropriate to reduce the time that is needed for reorganization.
- Index rebuilds help performance greatly but can slow the database system down greatly. Therefore, rebuild offline if possible and only when needed.

The Platform Maintenance Scheduler has commands for IBM Db2 that automatically reorganizes tables and indexes. In addition, IBM Db2 provides the **REORGCHK** and **REORG TABLE** commands to perform these activities directly on the database. However, use the Platform Maintenance Scheduler for this task.

For more information on reorganizing tables and indexes on an Oracle database, see the Oracle Database Documentation Library. For more information on Microsoft SQL Server databases, see the Reorganize and Rebuild Indexes.

Multibyte character sets

Using multibyte character sets (MBCS) on the database has performance implications. Single-byte character sets are better for performance than multibyte character sets, and they are also the most efficient in terms of storage.

Archiving and deleting historical data

Deleting unnecessary data from the database helps to optimize its performance. If you are archiving to an external system or database, use integrations or reports to query the archived data. Archiving data can also facilitate upgrading to a new version of TRIRIGA because the historical data is not critical to day-to-day operations and might not be needed in the new system.

Some databases have tools to assess data usage and aid in information lifecycle management, such as heat maps. Use these tools to identify data that might be purged or archived.

Table spaces

TRIRIGA uses two table spaces, `TRIDATA_DATA` and `TRIDATA_INDX`. These names are not mandatory, and you can use your own names. These tables separate the data tables from the indexes. `TRIDATA_DATA` contains all the `IBM_TRIRIGA` tables, and the `TRIDATA_INDX` table space stores all the indexes.

Over time, custom indexes might not be created in the proper table space. Check that these indexes after the system changes.

Use a third table space if you have large amounts of data and heavy usage of the TRIRIGA Document Manager. Moving the tables that store the binary large objects (BLOBs) into their own table space optimizes performance. However, TRIRIGA does not currently manage multiple table spaces, so track and record any changes that you make.

Database server virtualization

Virtualization always adds to processor usage, and while virtualization is supported in all tiers of the stack, do not use it for the database tier because of its resource use. Testing shows a near linear 30% CPU resource overhead when a database is run on a virtual machine (VM), with transactions per second also decreasing by approximately 5%.

TRIRIGA is not aware if it is running SQL statements against a virtualized or physical database. TRIRIGA makes requests and awaits responses. However, with virtualization in place, these responses might be delayed due to overhead, which is outside of TRIRIGA. Usually, this delay is fractions of a second, and while negligible, this delay can accumulate during heavy system usage and potentially become a performance degradation. Given this behavior, run the database on physical hardware.

If you are using virtualization for the database, it becomes another layer to consider when you diagnose performance issues. Use dedicated resources, including dedicated fast storage that is attached to the virtual environment.

Virtualized Environments:

Deployments in virtualized environments such as VMware might not see the same performance benefits from these practices. Using shared resources in virtualized environments does not give optimal performance. Instead, use dedicated CPU and memory resources.

Internal testing shows that putting the TRIRIGA database in a virtualized environment such as VMware has significant performance impacts. These impacts increase rapidly with larger workloads. While some customers run their TRIRIGA database successfully on a virtual machine, some use dedicated hardware for optimal performance.

If you deploy TRIRIGA on VMware, see the following tuning guide that is available from VMware: Performance Best Practices for VMware vSphere 5.5

Database server throughput

Ensure that the throughput of the database server meets expectations for enterprise-class fast data storage, which is 8 Gbps Fibre Channel or better. Follow the instructions by using the spreadsheet for your database server.

The spreadsheet shows whether a performance issue is at the database I/O level. This test removes any TRIRIGA platform code and runs a full table scan on one of the largest tables in the system. The number of blocks that are processed is divided by the number of seconds it takes to run the full table scan. Then, compare the result to a reference system that is configured with an 8 Gbps Fibre Channel storage area network (SAN).

To test the throughput of the database server, take the following steps:

1. Download and open the spreadsheet file that corresponds to your database server. Several statements are provided in each spreadsheet:
   - DB2® Database Perf IO Throughput.xlsx
   - Oracle Database Perf IO Throughput.xlsx
   - MSSQL Database Perf IO Throughput.xlsx
2. In a database tool that corresponds to your database server, such as Oracle SQL Developer, run the spreadsheet statements as your TRIDATA database user.

Database disk-caching policies

Your disk-caching policies might affect database performance. For example, if the hardware of your staging and production environments is identical, your staging disks might be configured with a write-back caching policy. At the same time, your production disks might be configured with a write-through caching policy. In this example, write-back might be 4 times faster than write-through.

To verify that differences in disk-caching policies are affecting performance, run a test on staging by switching one disk, or group of disks, from one caching policy to another policy. Then conduct a speed test, and switch back to the original policy for another speed test.

After you verify any differences in performance, implement the disk-caching policy change to your production environment.

Take the following steps:

1. Stop your production site and database.
2. Ensure that no processes or services are either reading or writing to any of the production disks. Ensure that there are no side effects due to cached data.
3. Change your production disks to the write-back caching policy.
4. Run a quick speed test to verify that performance is improved.
5. Restart your production database and site. Depending on your environment, the outage takes approximately 30-45 minutes.

Creating batches for business object (BO) data cleanup

If you are doing a large BO data cleanup exercise and have hundreds of thousands or millions of records to delete, create batches to spread the cleanup work over a time. Use the **mod** function to do the cleanup. The **mod** function returns the remainder as an integer when dividing one number by another. Use this function to group a large set of numbers into a nearly-equivalent number of groups or batches.

For example, run the following query to divide the set of `spec_id` records into 7 roughly equal batches.

- Oracle Database or IBM Db2:

```
select count(*), mod(spec_id ,7) from ibs_spec
group by mod(spec_id ,7);
```

- Microsoft SQL Server:

```
select count(*), spec_id % 7 from ibs_spec
group by spec_id % 7;
```

Using this information, split the cleanup work into a more reasonable number of records. Typically, cleaning 10,000-20,000 records takes about 30-60 minutes. You can modify the query by adding a `where` clause to filter only the records to be deleted.

- Find the total number of records to be deleted. Then run the following query:

```
select count(*)/20000 from ibs_spec
where object_id < 0
and system_flag = 0;
```

For example, assume that the result of this query is 7 batches. Then run the following query to verify the number of records to be cleaned up in the following days:

- Oracle Database or IBM Db2:

```
select count(*), mod(spec_id ,7) from ibs_spec
where object_id < 0
and system_flag = 0
group by mod(spec_id ,7);
```

- Microsoft SQL Server:

```
select count(*), spec_id % 7 from ibs_spec
where object_id < 0
and system_flag = 0
group by spec_id % 7;
```

To implement the query, update the `ibs_spec` table and set the `updated_date` from today, adding a day into the future:

- Oracle Database or IBM Db2:

```
update ibs_spec
set updated_date = sysdate + mod(spec_id ,7)
where object_id < 0 and system_flag = 0;
```

- Microsoft SQL Server:

```
update ibs_spec
set updated_date = getdate() + spec_id % 7
where object_id < 0 and and system_flag = 0;
```

IBM Db2

Tune Db2 components such as the server and indexes.

Tuning the IBM Db2 server

Tune the registry and database manager settings and the automatic buffer pool size and auto extends.

Db2 is configured automatically if you use the TRIRIGA scripts to configure your instance and database. For more information, see the Installation and Implementation Guide. However, you can also tune the database to maximize your specific Db2 implementation.

Registry and database manager settings

Use the following Db2 registry settings so that your database performs efficiently:

```
db2set DB2_COMPATIBILITY_VECTOR=ORA
db2set DB2_DEFERRED_PREPARE_SEMANTICS=YES
db2set DB2_ATS_ENABLE=YES
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
```

Use the following Db2 database manager settings:

```
db2 update dbm cfg using RQRIOBLK 65535
```

Create the database with PAGESIZE=32K.

The following Db2 database configuration settings are required:

```
db2 update db cfg for <dbname> using STMT_CONC OFF
db2 update db cfg for <dbname> using LOCKTIMEOUT 30
```

The following optional database configuration settings can improve the performance of the log file:

```
db2 update db cfg for <dbname> using LOGPRIMARY 23
db2 update db cfg for <dbname> using LOGFILSIZ 32768
db2 update db cfg for <dbname> using LOGSECOND 12
db2 update db cfg for <dbname> using LOGBUFSZ 8192
db2 update db cfg for <dbname> using CATALOGCACHE_SZ 2048
```

If you are supporting multibyte characters, configure the database with the UTF-8 code set and the CODEUNITS32 string unit:

```
db2 update db cfg for <dbname> using string_units CODEUNITS32
```

Do not turn on automatic maintenance settings, such as AUTO_MAINT, AUTO_TBL_MAINT, AUTO_RUNSTATS, and AUTO_STMT_STATS. Instead, schedule database maintenance activities during regular maintenance windows to avoid affecting user performance.

To help the query optimizer select an efficient access plan, specify the REOPT(ONCE) bind option when you run queries. When you use the REOPT(ONCE) bind option, the query optimizer selects the access

plan the first time that the query is run. Each subsequent time that the query is run, the access plan is reused. To specify the REOPT(ONCE) bind option, run the following command:

```
db2 bind '<db2home>/bnd/db2clipk.bnd' collection NULLIDR1
```

The user ID that is created for TRIRIGA to access the database must have DBADM, SECADM, ACCESSCTRL, and DATAACCESS privileges for the database. For information about Db2 administration tasks, see the DB2 product documentation.

Db2 automatic buffer pool size and auto extends

To avoid a performance issue, ensure that the database is set to use an AUTOMATIC buffer pool instead of a static size.

To manually change the database settings, complete the following steps:

1. In IBM Data Studio, log in as the TRIRIGA user to the database or instance that is used by TRIRIGA.
2. Run the following SQL:

   ```
   > select AUTOMATIC FROM TABLE(MON_GET_BUFFERPOOL('',-2)) where upper(bp_name) =
   'TRIRIGABUFFERPOOL'
   ```

   - If the return value is 0 (zero), then continue to step 3.
   - If the return value is 1 (one), then the buffer pool is already set to AUTOMATIC.
3. Stop the application servers that run Db2.
4. Run the following SQL in IBM Data Studio:

   ```
   alter bufferpool TRIRIGABUFFERPOOL immediate size AUTOMATIC
   ```
5. Run the original SQL **select** to make sure that the return value is now 1 (one).
6. Start and stop the Db2 instance that is used by TRIRIGA:
   - For Windows, from the Db2 Command Window - Administrator, log in as the Db2 administrator user, which is typically db2admin, and run the following commands, where *XXXXX* is the name of the instance that is used by TRIRIGA:

     ```
     set db2instance=XXXXX
     ```

     ```
     db2stop force
     ```

     ```
     db2start
     ```
   - For Linux, from a shell window, log in as the TRIRIGA instance user and run the following command:

     ```
     db2stop force
     ```

     ```
     db2start
     ```
7. Restart the application servers that run Db2.

Db2 diagnostic log

Ensure that the diagnostic log for errors does not become large enough to cause storage space issues. An oversize diagnostic log might cause performance problems. For information about configuring Db2 to use rotating Db2 diagnostic log files, see DB2 diagnostic (db2diag) log files.

Tuning the IBM Db2 indexes

Tuning the application platform, reserve, and lease indexes can improve performance.

Overview

These indexes provide significant performance improvements when measured against a broad performance test workload. However, database platform performance gains might vary depending on factors such as application usage, load patterns, hardware sizing, application, and database server configuration. Db2 database administrators must monitor databases for efficient index usage to determine the impact that is produced by applying the recommended indexes. They must also determine more indexes that improve performance based on situational and data composition needs.

The indexes that are listed are not included in the TRIRIGA base product unless otherwise stated.

Application platform indexes

Add the following performance tuning indexes to Db2 databases.

```
CREATE INDEX PERF01_T_TRISPACEALLOCATIONFACT ON T_TRISPACEALLOCATIONFACT
(TRICAPTUREPERIODTXOBJID ASC, TRIFACTALLOCMOVESNU ASC, TRIFACTALLOCWORKERSNU ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRISPACEALLOCATIONFACT');
CALL SYSPROC.admin_cmd('runstats on table T_TRISPACEALLOCATIONFACT WITH distribution and
sampled detailed index PERF01_T_TRISPACEALLOCATIONFACT');

CREATE INDEX PERF02_T_TRISPACEALLOCATIONFACT ON T_TRISPACEALLOCATIONFACT
(TRICAPTUREPERIODTXOBJID ASC, TRIDIMWORKPOINTFLAGLI ASC, TRIFACTALLOCAREAIMPNU ASC,
TRIFACTALLOCWORKPOINTS ASC, TRIDIMSPACECLASSTXOBJID ASC, TRIDIMLOCATIONTXOBJID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRISPACEALLOCATIONFACT');
CALL SYSPROC.admin_cmd('runstats on table T_TRISPACEALLOCATIONFACT WITH distribution and
sampled detailed index PERF02_T_TRISPACEALLOCATIONFACT');

CREATE INDEX PERF01_T_TRIPEOPLE ON T_TRIPEOPLE
(TRIRECORDNAMESY ASC, SPEC_ID DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIPEOPLE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIPEOPLE WITH distribution and sampled detailed
index PERF01_T_TRIPEOPLE');

CREATE INDEX PERF01_T_TRICOSTCODE ON T_TRICOSTCODE
(SYS_PROJECTID ASC, TRISTATUSCL ASC, SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRICOSTCODE');
CALL SYSPROC.admin_cmd('runstats on table T_TRICOSTCODE WITH distribution and sampled detailed
index PERF01_T_TRICOSTCODE');

CREATE INDEX PERF01_T_TRISPACE ON T_TRISPACE
(TRIPARENTPROPERTYTXOBJID ASC, TRIIDTX ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRISPACE');
CALL SYSPROC.admin_cmd('runstats on table T_TRISPACE WITH distribution and sampled detailed
index PERF01_T_TRISPACE');

CREATE UNIQUE INDEX PERF01_T_TRISPACECLASSCURRENT ON T_TRISPACECLASSCURRENT
(SPEC_ID ASC) INCLUDE (TRINAMETX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRISPACECLASSCURRENT');
CALL SYSPROC.admin_cmd('runstats on table T_TRISPACECLASSCURRENT WITH distribution and sampled
detailed index PERF01_T_TRISPACECLASSCURRENT');

CREATE INDEX PERF02_T_TRIPEOPLE ON T_TRIPEOPLE
(SYS_GUIID ASC, UPPER(triLastNameTX) ASC, SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIPEOPLE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIPEOPLE WITH distribution and sampled detailed
index PERF02_T_TRIPEOPLE');

CREATE INDEX PERF01_T_ORGANIZATION ON T_ORGANIZATION
(SYS_GUIID ASC, UPPER(triNameTX) ASC, SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_ORGANIZATION');
CALL SYSPROC.admin_cmd('runstats on table T_ORGANIZATION WITH distribution and sampled detailed
index PERF01_T_ORGANIZATION');

CREATE INDEX PERF03_T_TRIWORKTASK ON T_TRIWORKTASK
```

```
(SYS_GUIID ASC, UPPER(triNameTX) ASC, SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIWORKTASK');
CALL SYSPROC.admin_cmd('runstats on table T_TRIWORKTASK WITH distribution and sampled detailed
index PERF03_T_TRIWORKTASK');
```

Reserve indexes

In internal performance testing, the following indexes increased performance on Db2. Review, add, and tune the indexes for your implementation of the Reserve index:

```
CREATE INDEX "PERF01_TRIRESERVATIONINSTANCE" ON "T_TRIRESERVATIONINSTANCE"
(triPlannedStartDT,SYS_OBJECTID,SYS_GUIID,SYS_PROJECTID,triStatusCL) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIRESERVATIONINSTANCE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIRESERVATIONINSTANCE WITH distribution and
sampled detailed index PERF01_TRIRESERVATIONINSTANCE');

CREATE INDEX "PERF01_TRIRESERVATIONRESOURCE" ON "T_TRIRESERVATIONRESOURCE"
(SPEC_ID,SYS_OBJECTID,triResourceTypeLI,SYS_PROJECTID) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIRESERVATIONRESOURCE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIRESERVATIONRESOURCE WITH distribution and
sampled detailed index PERF01_TRIRESERVATIONRESOURCE');

CREATE INDEX "PERF03_TRIPEOPLE" ON "T_TRIPEOPLE" (SPEC_ID,SYS_OBJECTID) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIPEOPLE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIPEOPLE WITH distribution and sampled detailed
index PERF03_TRIPEOPLE');

CREATE INDEX "PERF01_MYPROFILE" ON "T_MYPROFILE" (SPEC_ID,SYS_OBJECTID,triRecordIdSY) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_MYPROFILE');
CALL SYSPROC.admin_cmd('runstats on table T_MYPROFILE WITH distribution and sampled detailed
index PERF01_MYPROFILE');

CREATE INDEX "PERF01_TRIRESERVATIONDEF" ON "T_TRIRESERVATIONDEFINITION"
(SPEC_ID,SYS_OBJECTID,SYS_GUIID,SYS_PROJECTID) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIRESERVATIONDEFINITION');
CALL SYSPROC.admin_cmd('runstats on table T_TRIRESERVATIONDEFINITION WITH distribution and
sampled detailed index PERF01_TRIRESERVATIONDEF');

CREATE INDEX "PERF01_TRIROLE" ON "T_TRIROLE" (SPEC_ID,triNameTX) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRIROLE');
CALL SYSPROC.admin_cmd('runstats on table T_TRIROLE WITH distribution and sampled detailed
index PERF01_TRIROLE');

CREATE INDEX "PERF01_TRICONTACTROLE" ON "T_TRICONTACTROLE"
(SPEC_ID,SYS_OBJECTID,ClassifiedByRoleSysKey) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
CALL SYSPROC.admin_cmd('reorg table T_TRICONTACTROLE');
CALL SYSPROC.admin_cmd('runstats on table T_TRICONTACTROLE WITH distribution and sampled
detailed index PERF01_TRICONTACTROLE');
```

Lease indexes

Tune any implementation of the Lease Management and Lease Accounting applications to include indexes for performance improvement. The following indexes can increase performance for lease queries. However, review and tune these indexes for your specific implementation.

```
-- LIST OF RECOMMENDED INDEXES
-- ===========================
CREATE INDEX TRIDATA.PERF1804101535060 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804101535230 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID ASC)
INCLUDE (TRIACCOUNTINGTYPELI, SYS_GUIID, TRIDUEDA, TRIEXPECTEDEXPENSENU, TRIEXPECTEDAMOUNTNU,
TRIEXPECTEDACCRUALNU, TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL, TRISTATUSCLOBJID, TRISTATUSCL,
TRICURRENCYUO, SYS_TYPE1, TRIUSERMESSAGEFLAGTX, SYS_OBJECTID) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

```
CREATE INDEX TRIDATA.PERF1804101544190 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC, SPEC_ID
DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804101551040 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804201552470 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC,
ENDDATETIME ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804201553110 ON TRIDATA.T_SCHEDULEDEVENTS (EVENTSTATUS ASC,
SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804201629100 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC,
STARTDATETIME ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221912280 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221916070 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221918070 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, SYS_GUIID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221919560 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC, SYS_GUIID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221921350 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221923250 ON TRIDATA.T_TRIBUILDING (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221925140 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804221928490 ON TRIDATA.T_TRIPROPERTY (SYS_GUIID ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222031340 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX
ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222035120 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX
ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222037150 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, TRISTATUSCL ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222038500 ON TRIDATA.T_TRIFLOOR (SYS_PROJECTID ASC, SYS_OBJECTID
```

```
ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX
ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222040390 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC,
TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222042280 ON TRIDATA.T_TRIBUILDING (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC,
TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222044170 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC,
TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1804222047510 ON TRIDATA.T_TRIPROPERTY (SYS_GUIID ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC,
TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804222208420 ON TRIDATA.T_TRILEASECLAUSE (SPEC_ID ASC) INCLUDE
(SYS_GUIID, TRICLAUSESECTIONCATEGO, TRICLAUSETYPECL, SYS_OBJECTID, SYS_TYPE1, TRIDESCRIPTIONTX,
TRIPAGETX, TRISECTIONTX, TRIGRANTEDINLEASECLOBJID, TRIGRANTEDINLEASECL, TRICLAUSETYPECLOBJID,
TRICLAUSESECTIONCATEGOOBJID, TRIUSERMESSAGEFLAGTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804222214470 ON TRIDATA.T_TRILEASECLAUSE (SPEC_ID ASC)
INCLUDE (SYS_GUIID, SYS_OBJECTID, SYS_TYPE1, TRIDESCRIPTIONTX, TRIPAGETX, TRISECTIONTX,
TRIGRANTEDINLEASECLOBJID, TRIGRANTEDINLEASECL, TRICLAUSETYPECLOBJID, TRICLAUSETYPECL,
TRICLAUSESECTIONCATEGOOBJID, TRIUSERMESSAGEFLAGTX, TRICLAUSESECTIONCATEGO) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804232123170 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID ASC)
INCLUDE (SYS_PROJECTID, ISPAYMENTFORSYSKEY, SYS_OBJECTID, TRICURRENCYUO, TRIPAYMENTTYPECLOBJID,
TRIPAYMENTTYPECL, TRIEXPECTEDAMOUNTNU, TRIACTUALAMOUNTNU, TRISTATUSCLOBJID, TRISTATUSCL,
SYS_TYPE1, SYS_GUIID) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804232356320 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID ASC)
INCLUDE (SYS_PROJECTID, ISPAYMENTFORSYSKEY, SYS_OBJECTID, TRICURRENCYUO, TRIPAYMENTTYPECLOBJID,
TRIPAYMENTTYPECL, TRIEXPECTEDAMOUNTNU, TRIACTUALAMOUNTNU, TRISTATUSCLOBJID, TRISTATUSCL,
SYS_TYPE1, SYS_GUIID) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1804232356370 ON TRIDATA.T_TRIREALESTATECONTRACT (SPEC_ID ASC)
INCLUDE (TRINAMETX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX PERF35.PERF_GETPAYMENTS ON PERF35.T_TRIPAYMENTLINEITEM (TRIPROCESSPERIODCL
ASC, TRISTATUSCL ASC, TRIACCOUNTINGTYPELI ASC, TRIPAYPROCESSEDMODELI ASC, SYS_GUIID ASC,
SYS_OBJECTID ASC, TRIDUEDA ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072214360 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072216250 ON TRIDATA.T_TRILAND (SYS_PROJECTID ASC, SYS_OBJECTID
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

```
CREATE INDEX TRIDATA.PERF1806072218140 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072221430 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRIPARENTPROPERTYTX ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072223320 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC, SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRIPARENTPROPERTYTX ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072223410 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072227100 ON TRIDATA.T_TRIBUILDING (SYS_TYPE1 ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRIPARENTPROPERTYTX
ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX
ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072227190 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072229080 ON TRIDATA.T_TRIPROPERTY (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072230570 ON TRIDATA.T_LOCATION (SYS_PROJECTID ASC, SYS_OBJECTID
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072232460 ON TRIDATA.T_TRIPROPOSEDSITE (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072238300 ON TRIDATA.T_TRILAND (SYS_PROJECTID ASC, SYS_OBJECTID
ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRISTATEPROVTX ASC,
TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072243510 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072245400 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

```
CREATE INDEX TRIDATA.PERF1806072245490 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX
ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072249210 ON TRIDATA.T_TRIBUILDING (SYS_TYPE1 ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072251210 ON TRIDATA.T_TRIPROPERTY (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX
ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072253120 ON TRIDATA.T_LOCATION (SYS_PROJECTID ASC, SYS_OBJECTID
ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, TRISTATEPROVTX ASC,
TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL
ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072255030 ON TRIDATA.T_TRIPROPOSEDSITE (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX
ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072327310 ON TRIDATA.T_TRIPEOPLE (SYS_GUIID ASC, SYS_OBJECTID ASC,
PRIMARYORGANIZATIONSYSKEY ASC, TRITITLETX ASC, TRISTATUSCL ASC, TRIEMAILTX ASC, TRIWORKPHONETX
ASC, TRIWORKFAXTX ASC, TRINAMETX ASC, TRIUSERMESSAGEFLAGTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806072328180 ON TRIDATA.T_ORGANIZATION (SPEC_ID ASC) INCLUDE
(TRIPATHTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080059310 ON TRIDATA.T_TRILEASEABSTRACT (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GUIID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080101330 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID ASC,
SYS_PROJECTID ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC,
TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080101410 ON TRIDATA.T_TRIPURCHASEREQUISITION (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080103340 ON TRIDATA.T_TRISTANDARDCONTRACT (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080105280 ON TRIDATA.T_TRIASSETLEASE (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080107220 ON TRIDATA.T_TRIWARRANTY (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

```
CREATE INDEX TRIDATA.PERF1806080111080 ON TRIDATA.T_TRIPURCHASEORDER (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080113010 ON TRIDATA.T_TRISPACEUSEAGREEMENT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080116470 ON TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080120340 ON TRIDATA.T_TRIPRIMECONTRACT (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080122280 ON TRIDATA.T_TR_STAND_CONTR_CHAN_ORD (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080124220 ON TRIDATA.T_TRICONTRACT (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080126160 ON TRIDATA.T_CSTDEFAULTRETENTION (SYS_PROJECTID ASC,
TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080128380 ON TRIDATA.REP_TEMPLATE_HDR (REP_NAME ASC,
OBJECT_TEMPLATE_ID ASC, CLASS_TYPE_ID ASC, REP_TEMPLATE_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806080129090 ON TRIDATA.MODULE_ASSOCIATION (ASS_BO_ID ASC,
DEPENDENT_FLAG ASC, CASCADE_READ_ONLY ASC, REV_ASSOCIATION_VERB ASC, PROJ_CONTAINMENT_DISABLED
ASC, ASSOCIATION_VERB ASC, SOURCE_BO_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806071659590 ON TRIDATA.T_TRIAPPROVAL (TRIRECORDIDSY ASC, SYS_GUIID
ASC, TRILINKEDBUSINESSOBJ ASC, TRILINKEDRECORDTX ASC, TRISUBMITTEDBYTX ASC, TRISTATUSCL ASC,
SYS_OBJECTID ASC, SPEC_ID ASC, SYS_TYPE1 ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806071724280 ON TRIDATA.T_ORGANIZATION (SYS_GUIID ASC, SYS_OBJECTID
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC,
TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806071721520 ON TRIDATA.T_TRICITY (SYS_OBJECTID ASC, TRIPATHSY ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806071724500 ON TRIDATA.T_ORGANIZATION (SPEC_ID ASC) INCLUDE
(SYS_OBJECTID, TRIPATHSY) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806071945040 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID
ASC) INCLUDE (TRISUMMARYTYPECL, TRISTATUSCL, TRIACCOUNTINGTYPELI, TRIDUEDA, SYS_OBJECTID,
TRICURRENCYUO, SYS_GUIID, SYS_TYPE1, TRIEXPECTEDTOTALNU, TRISTATUSCLOBJID, TRIEXPECTEDAMOUNTNU,
TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806071948000 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID ASC)
INCLUDE (TRISUMMARYTYPECL, TRISTATUSCL, TRIACCOUNTINGTYPELI, SYS_OBJECTID, TRICURRENCYUO,
```

```
SYS_GUIID, SYS_TYPE1, TRISTATUSCLOBJID, TRIACTUALAMOUNTNU, TRIPAYMENTTYPECLOBJID,
TRIPAYMENTTYPECL) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806071623390 ON TRIDATA.UI_TARGET (PRIMARY_OBJECT_ID ASC,
UI_TARGET_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806071623470 ON TRIDATA.NAV_ITEM (UI_TARGET_ID ASC, NAV_ITEM_ID DESC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072038510 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRICONTRACTSTATUSCLOBJID ASC, TRICONTRACTSTATUSCL
ASC, TRIAREAUO ASC, TRICOUNTRYTX ASC, TRISTATEPROVTX ASC, TRICITYTX ASC, TRIIMAGEIM
ASC, TRICONTRACTRENTABLENU ASC, TRIEXPIRATIONDA ASC, TRINAMETX ASC, TRISTATUSCL ASC,
TRIUSERMESSAGEFLAGTX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072042370 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_OBJECTSTATE
ASC, SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRICONTRACTSTATUSCLOBJID ASC,
TRICONTRACTSTATUSCL ASC, TRIAREAUO ASC, TRICOUNTRYTX ASC, TRISTATEPROVTX ASC, TRICITYTX ASC,
TRIIMAGEIM ASC, TRICONTRACTRENTABLENU ASC, TRIEXPIRATIONDA ASC, TRINAMETX ASC, TRIIDTX ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072102530 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_OBJECTSTATE
ASC, SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072106180 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072109460 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_OBJECTSTATE ASC,
SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRINAMETX ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072121060 ON TRIDATA.T_TRIPEOPLE (TRIRECORDNAMETX ASC, SPEC_ID
DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072126500 ON TRIDATA.T_TRIINVOICE (SYS_PROJECTID ASC, SYS_GUIID
ASC, SYS_OBJECTID ASC, TRIDATEDA ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072131470 ON TRIDATA.T_TRILEASEABSTRACT (SYS_TYPE1 ASC,
SYS_OBJECTID ASC, TRIBUSINESSOBJECTLABEL2 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072132150 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_TYPE1 ASC,
SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072132430 ON TRIDATA.T_TRIPURCHASEREQUISITION (SYS_TYPE1 ASC,
SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072132450 ON TRIDATA.T_TRISTANDARDCONTRACT (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072133130 ON TRIDATA.T_TRIASSETLEASE (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072134090 ON TRIDATA.T_TRIBLANKETPURCHASEORDER (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;
```

```
CREATE INDEX TRIDATA.PERF1806072135310 ON TRIDATA.T_TRISPACEUSEAGREEMENT (SYS_TYPE1 ASC,
SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072136010 ON TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072136570 ON TRIDATA.T_TRIPRIMECONTRACT (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072137530 ON TRIDATA.T_TRICONTRACT (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072138210 ON TRIDATA.T_CSTDEFAULTRETENTION (SYS_OBJECTID ASC,
TRIBUSINESSOBJECTLABEL1 ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072130330 ON TRIDATA.T_TRIREINVOICE (SYS_GUIID ASC, SYS_OBJECTID
ASC, HASCONTRACTSYSKEY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC,
TRIIDTX ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806072138490 ON TRIDATA.T_TRIREALESTATECONTRACT (SPEC_ID ASC)
INCLUDE (TRINAMETX, TRIIDTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806072141080 ON TRIDATA.T_TRICONTACTROLE (SPEC_ID ASC) INCLUDE
(SYS_OBJECTID, TRISTATUSCL) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE UNIQUE INDEX TRIDATA.PERF1806072141250 ON TRIDATA.T_TRISTANDARDCONTRACT
(SPEC_ID ASC) INCLUDE (SYS_GUIID, TRIIDTX, TRINAMETX, TRITOORGANIZATIONTXOBJID,
TRISTATUSCL, TRISTATUSCLOBJID, TRIPENDINGCHANGESROL, TRICOMMITMENTORIGINA, TRICURRENCYUO,
TRITOORGANIZATIONTX, SYS_TYPE1, TRICOMMITMENTCHANGES, SYS_OBJECTID) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806072144040 ON TRIDATA.T_TRIPURCHASEORDER (SYS_GUIID ASC,
SYS_OBJECTID ASC, TRICURRENCYUO ASC, SYS_TYPE1 ASC, TRISTATUSCLOBJID ASC, TRILINEITEMTOTALNU
ASC, TRIVENDORCOMPANYLOOKOBJID ASC, TRIVENDORCOMPANYLOOK ASC, TRINAMETX ASC, TRIIDTX ASC,
SPEC_ID ASC, TRISTATUSCL ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1808180010570 ON TRIDATA.T_TRIJOURNALENTRY (TRISTATUSCL ASC,
SYS_OBJECTID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1808180017030 ON TRIDATA.WF_EVENT (AGENT_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1808180017130 ON TRIDATA.WF_EVENT (AGENT_ID ASC, BO_ID DESC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1808180022250 ON TRIDATA.T_TRIJOURNALENTRY (SYS_OBJECTID ASC, SPEC_ID
DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

CREATE INDEX TRIDATA.PERF1806112022490 ON TRIDATA.APP_OBJECT_PERMISSION (GROUP_ID ASC,
COMPANY_ID ASC, SERVICE_ID ASC, APPLICATION_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_LOCATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_LOCATIONGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBUILDING FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIBUILDINGGUIID1;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINSTAGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILAND FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRILANDGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPLANNGUIID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPLANNGUIID5;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPOSEDSITE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPROPOGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIVERTIGUIID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILEASECLAUSE FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRILEASECLAUSE;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPAYMENTLINEITEM;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPAYMENTLINEITEM;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_LOCATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_LOCATIONOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILAND FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRILANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPOSEDSITE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPROPOOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRISPACEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIVERTICALSHAFT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIINSTALLATION;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIPLANNEDFLOOR;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIPLANNEDSPACE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIINSTAGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPLANNGUIID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPLANNGUIID5;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIVERTIGUIID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPEOPLE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPEOPLEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPEOPLE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPEOPLEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX TRIDATA.PK_ORGANIZATION;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICOUNTRY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRICOUNTRYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSURANCE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINSUROBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILEASEABSTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRILEASEOBJECTID3;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROCUREMENTCARD FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPROCUOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEREQUISITION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID3;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACEUSEAGREEMENT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISPACEOBJECTID26;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTANDARDCONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISTANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTATE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISTATEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIWARRANTY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIWARRANTYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TR_STAND_CONTR_CHAN_ORD FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TR_STANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIASSETLEASE FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIASSETLEASE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBLANKETPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIBLANKETPURCHASEORDER;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTRACT FOR SAMPLED DETAILED INDEX TRIDATA.PK_TRICONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPRIMECONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPRIMECONTRACTCHANGEORDER;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_CSTDEFAULTRETENTION FOR SAMPLED DETAILED INDEX
TRIDATA.P_CSTDEFAULTRETENTION;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_TRIAPPROVAL FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIAPPROVALGUIID1;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.T_TRIAPPROVAL FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIAPPROVALOBJECTID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_ORGANIZAOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX TRIDATA.PK_ORGANIZATION;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINVOICE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINVOICEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINVOICE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINVOICEOBJECTID1;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.IDX03_IBS_SPEC_ASSIGN;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_CSTDEFAULTRETENTION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_CSTDEFAUGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIASSETLEASE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIASSETLEASEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBLANKETPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIBLANKOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRICONTRACTGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPRIMEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPRIMEGUIID2;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREINVOICE FOR SAMPLED DETAILED INDEX TRIDATA.PK_TRIREINVOICE;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.T_TRICONTACTROLE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_TRICONTACTROLE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTANDARDCONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRISTANDARDCONTRACT;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.ASS_CTYP_TMPL_SPID;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTACTROLE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_TRICONTACTROLE;
-- COMMIT WORK;

-- RUNSTATS ON TABLE TRIDATA.APP_OBJECT_PERMISSION FOR SAMPLED DETAILED INDEX
TRIDATA.PK_APP_OBJECT_PERMISSION;
-- COMMIT WORK;
```

Internal lease benchmarks

Lease Management and Lease Accounting performance testing used a database that was loaded with approximately 17,000 lease abstracts and 24,000 leases with other large amounts of portfolio data to represent a large customer deployment of the lease application. The performance test indexes were applied to the data-load and benchmark-test environments to improve overall throughput and response times as a result of this project.

Performance environment

The performance environment consists of three hardware components based on the Advanced system configuration example in Chapter 3, “Tuning system architecture and hardware,” on page 7. The performance benchmark test was conducted by using a specific configuration of this environment:

Application servers
: The application servers run most of the business logic. Application server processes are CPU-intensive and require sufficient RAM. The application tier consists of JavaServer Pages (JSP) and Java classes. The Java Platform, Enterprise Edition application servers provide a JSP container, a database connection pool, and transaction management services. Application servers are the physical manifestation of the application, or middleware, tier.

Process servers
: The process servers are configured similarly to the application servers, but users do not log on to these servers. Process servers handle the asynchronous workflow requests that are queued from users or by the IBM TRIRIGA software. Process servers are the physical manifestation of the application tier.

Database server
: The database server runs the database processes. The application tier communicates with the database tier by using JDBC connection pools. Database servers are the physical manifestation of the database tier.

Data load hardware environment

The following values describe the data load hardware environment. With this configuration, the resource use bottlenecks were eliminated to load lease batches of at least 1000 leases at a time.

- Database server:
  - Number of vCPUs: 64 virtual CPUs
  - CPU: Intel® Xeon® CPU E5-2686 v4 @ 2.3 GHz
  - Storage: Storage Area network (SAN) attached
  - Memory: 488 GB
  - Network Interface Card (NIC): 1 GBit/sec
  - Operating System (OS): CentOS Linux 7.4, 64-bit
  - Database: Db2 Enterprise Database Server 11.1.2.2, 64-bit
- Application Server virtual machine 1 of 1:
  - Number of vCPUs: 4 virtual CPUs
  - CPU: Intel® Xeon® CPU E5-2686 v4 @ 2.3 GHz
  - Memory: 32 GB
  - OS: CentOS Linux 7.4, 64-bit
  - Application Server: WebSphere Liberty Profile (TRIRIGA embedded)
  - Java Version: Oracle Java 1.8.0_162
  - Java virtual machine (JVM) Heap Size: 4096 MB
  - Number of JVMs: 1
  - Usage: Front-end access only
- Process Server virtual machine 1 of 3:
  - Number of vCPUs: 36 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 72 GB
  - OS: CentOS Linux 7.4, 64-bit
  - Application Server: WebSphere Liberty Profile (TRIRIGA embedded)
  - Java Version: Oracle Java 1.8.0_162
  - JVM Heap Size: 6144 MB
  - Number of JVMs: 1
  - Usage: Non-PM data loads restricted to single user
- Process Server virtual machine 2 of 3:
  - Number of vCPUs: 72 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 144 GB
  - OS: CentOS Linux 7.4, 64-bit
  - Application Server: WebSphere Liberty Profile (TRIRIGA embedded)
  - Java Version: Oracle Java 1.8.0_162
  - JVM Heap Size: 6144 MB
  - Number of JVMs: 1
  - Usage: Lease data loader restricted to data-load user and SCHEVENT user
- Process Server virtual machine 3 of 3:
  - Number of vCPUs: 72 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 144 GB
  - OS: CentOS Linux 7.4, 64-bit

  - Application Server: WebSphere Liberty Profile (TRIRIGA embedded)
  - Java Version: Oracle Java 1.8.0_162
  - JVM Heap Size: 6144 MB
  - Number of JVMs: 1
  - Usage: System workflow processing restricted to system user

Multi-user benchmark test environment

The following values describe the multi-user benchmark-test hardware environment.

- Database Server:
  - Number of vCPUs: 64 virtual CPUs
  - CPU: Intel® Xeon® CPU E5-2686 v4 @ 2.3 GHz
  - Storage: SAN attached
  - Memory: 488 GB
  - NIC: 1 GBit/sec
  - OS: CentOS Linux 7.4, 64-bit
  - Database: Db2 Enterprise Database Server 11.1.2.2, 64-bit
- Application Servers (VMs):
  - Number of vCPUs: 16 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 32 GB
  - OS: CentOS Linux 7.4, 64-bit
  - Application Server: WebSphere Application Server 8.5.5.12
  - Java Version: IBM Java 1.8.0 SR4 FP5, 64-bit
  - JVM Heap Size: 4096 MB
  - Number of VMs: 2
  - JVMs per VM: 1
- Process and BIRT/Integration Servers (VMs):
  - Number of vCPUs: 36 virtual CPUs
  - CPU: Intel® Xeon® CPU E5-2686 v4 @ 2.3 GHz
  - Memory: 72 GB
  - OS: CentOS Linux 7.4, 64-bit
  - Application Server: WebSphere Application Server 8.5.5.12
  - Java Version: IBM Java 1.8.0 SR4 FP5, 64-bit
  - JVM Heap Size: 6144 MB
  - Number of VMs: 2
  - JVMs per VM: 1

Note:

Although other values apply to older software versions, the following values describe the multi-user benchmark-test environment for IBM TRIRIGA 11.6 and IBM TRIRIGA Application Platform 5.0.

- Database Server:
  - Number of vCPUs: 64 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8259CL CPU @ 2.5 GHz
  - Storage: SAN attached

  - Memory: 498 GB
  - NIC: 1 GBit/sec
  - OS: Red Hat® Enterprise Linux 9.4
  - Database: Db2 Enterprise Database Server 11.5.8.0
- Application Server (VM 1 of 1):
  - Number of vCPUs: 16 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 32 GB
  - OS: Red Hat Enterprise Linux 9.3
  - Application Server: WebSphere Application Server Liberty (IBM TRIRIGA embedded)
  - Java Version: Oracle Java 17
  - JVM Heap Size: 4096 MB
  - Number of VMs: 1
  - JVMs per VM: 1
- Process Server (VM 1 of 1):
  - Number of vCPUs: 16 virtual CPUs
  - CPU: Intel® Xeon® Platinum 8124M CPU @ 3.0 GHz
  - Memory: 32 GB
  - OS: Red Hat Enterprise Linux 9.3
  - Application Server: WebSphere Application Server Liberty (IBM TRIRIGA embedded)
  - Java version: Oracle Java 17
  - JVM Heap Size: 6144 MB
  - Number of VMs: 1
  - JVMs per VM: 1

Software environment

The following values show the software versions that were used for both the data-load and multi-user benchmark-test environments:

- IBM TRIRIGA 10.5.3.2
- IBM TRIRIGA Application Platform 3.5.3.4

Key configurations

TRIRIGA uses standard settings and settings from the performance best practices.

The following settings and values identify the key tuning and configuration for the performance test environments.

The settings with asterisks (*) were modified from best practices to achieve a balance between response times and resource utilization.

Database server

For more information, see section *IBM Db2 Database Server Tuning*.

- Registry settings:
  - `DB2_COMPATIBILITY_VECTOR: ORA`
  - `DB2_DEFERRED_PREPARE_SEMANTICS: YES`
  - `DB2_ATS_ENABLE: YES`

  - DB2_USE_ALTERNATE_PAGE_CLEANING: ON
- Database Manager (DBM) configuration:
  - RQRIOBLK: 65535
  - AGENT_STACK_SZ: 1024
- Database (DB) configuration:
  - STMT_CONC: OFF
  - LOCKTIMEOUT: 30
  - LOGPRIMARY: 64*
  - LOGFILSIZ: 65535*
  - LOGSECOND: 48*
  - LOGBUFSZ: 8192
  - CATALOGCACHE_SZ: 2048
  - STRING_UNITS: CODEUNITS32
- Other settings:
  - db2 bind '<db2home>/bnd/db2clipk.bnd' collection NULLIDR1: REOPT(ONCE)
  - Bufferpool Size: AUTOMATIC
- DB connection pool:
  - Minimum connections: 10
  - Maximum connections: 100*
- Default thread pool (For Single-User Manual Test Environment Only)**:**
  - Minimum size: 20
  - Maximum size: 50
  - Thread inactivity timeout: 30000
- WebContainer thread pool - for single-user manual test environment only:
  - Minimum size: 120
  - Maximum size: 120
  - Thread inactivity timeout: 60000
- Data source custom properties:
  - webSphereDefaultIsolationLevel: 2
  - jdbcCollection: NULLIDR1

Operating system

For more information, see Chapter 4, “Tuning the operating system,” on page 13.

The following changes were applied to the application and process servers involved in the test:

- net.ipv4.ip_local_port_range = 32768 61000

The following `ulimit` changes were applied to all servers involved in the test:

- Max user processes 8192
- Open files 131072

TRIRIGA platform

For more information on tuning and configuring the `TRIRIGAWEB.properties` file, see “System properties” on page 77.

Data-load environment:

- Application servers:
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- Lease load process servers:
  - `WFAgentMaxThreads: 64*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 64*`
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- Data process server:
  - `WFAgentMaxThreads: 32*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 32*`
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`

Multi-user benchmark test environment:

- Application servers:
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- Workflow process servers:
  - `WFAgentMaxThreads: 64*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 16*`
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- BIRT and additional-agents process server:
  - `WFAgentMaxThreads: 64*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 16*`
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`

Note: Versions 11.6 and 5.0

Although other values apply to older software versions, the following values describe the multi-user benchmark-test environment for IBM TRIRIGA 11.6 and IBM TRIRIGA Application Platform 5.0.

For multi-user benchmark test environment:

- Application servers:
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- Workflow process servers:
  - `WFAgentMaxThreads: 80*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 70*`
  - `WF_INSTANCE_SAVE: ERRORS_ONLY`
  - `BIRT_MEMORY_USAGE_LIMIT: 35`
- BIRT and additional-agents process server:
  - `WFAgentMaxThreads: 80*`
  - `WF_AGENT_MAX_ACTIVE_PER_USER: 70*`

- WF_INSTANCE_SAVE: ERRORS_ONLY
- BIRT_MEMORY_USAGE_LIMIT: 35

Performance test indexes

Testing identified long-running SQL statements for data load, single user, and multi user test indexes. These statements were then analyzed with the IBM Db2 SQL Advisor and appropriate indexes were created.

Data load indexes

During the initial phases of the Lease data loading, Db2 snapshots were taken to identify long-running SQL statements. These snapshots were then analyzed with the Db2 SQL Advisor and appropriate indexes were created:

Non-data load environments:

Because the internal Lease data loader also uses the default workflows for the Generate Schedules, Accounting Review, and Activation applications, the following indexes might also apply to the individual test cases in the single-user manual test. However, because these indexes were applied first for data loading, they were not discovered during single-user manual testing. Consider using these indexes even for non-data-load environments.

Between adding the indexes and increasing the resources on the data-load environment as outlined, resource use issues were eliminated for data-load batches up to 1000 leases. The following Lease data is now loading:

- 5-year simple leases: 200 per hour
- 10-year simple leases: 100 per hour
- 5-year complex leases: 100 per hour
- 10-year complex leases: 50 per hour

```
Query 1:

SELECT T1.triUserMessageFlagTX AS T1_1017, T1.triAccountingTypeLI AS T1_1059,
T1.triPaymentTypeCL AS T1_1091, T1.triPaymentTypeCLObjId AS T1_1091_OBJID, T1.triDueDA AS
T1_1089, T1.triExpectedAccrualNU AS T1_1167, T1.triExpectedAmountNU AS T1_1088,
T1.triExpectedExpenseNU AS T1_1168, T1.triStatusCL AS T1_1023, T1.triStatusCLObjId AS
T1_1023_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS
T1_SPEC_ID, T1.triCurrencyUO AS T1_1143 FROM T_TRIPAYMENTLINEITEM T1 WHERE T1.SYS_GUIID = ? AND
( T1.triAccountingTypeLI = ? AND UPPER(T1.triPaymentTypeCL) LIKE ? AND (T1.triStatusCL != ? OR
T1.triStatusCL IS NULL) ) AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE
SPEC_ID IN (SELECT T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIREALESTATECONTRACT T1 WHERE
T1.SYS_PROJECTID = ? AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE
SPEC_ID = ?) AND T1.SYS_OBJECTID > ?)) AND T1.SYS_OBJECTID > ?

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1]
CREATE INDEX TRIDATA.PERF1804101535060 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2]
CREATE UNIQUE INDEX TRIDATA.PERF1804101535230 ON TRIDATA.T_TRIPAYMENTLINEITEM (SPEC_ID
ASC) INCLUDE (TRIACCOUNTINGTYPELI, SYS_GUIID, TRIDUEDA,
TRIEXPECTEDEXPENSENU, TRIEXPECTEDAMOUNTNU, TRIEXPECTEDACCRUALNU, TRIPAYMENTTYPECLOBJID,
TRIPAYMENTTYPECL, TRISTATUSCLOBJID, TRISTATUSCL, TRICURRENCYUO, SYS_TYPE1,
TRIUSERMESSAGEFLAGTX, SYS_OBJECTID) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

Query 2:

SELECT T1.StartDatetime AS T1_1006, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID,
T1.SPEC_ID AS T1_SPEC_ID FROM T_SCHEDULEDEVENTS T1 WHERE T1.SPEC_ID IN
(?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) AND T1.SYS_OBJECTID > ? ORDER BY CASE WHEN
T1.StartDatetime is NULL THEN 1 ELSE 0 END, T1.StartDatetime DESC;

-- LIST OF RECOMMENDED INDEXES
```

```
-- ============================
-- index[1]
CREATE INDEX TRIDATA.PERF1804101544190 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC,
SPEC_ID DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2]
CREATE INDEX TRIDATA.PERF1804101551040 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

Queries 3 & 4:

SELECT T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID
FROM T_SCHEDULEDEVENTS T1 WHERE T1.EventStatus = ? AND T1.EndDatetime <= ? AND T1.SYS_OBJECTID
> ? ORDER BY T1.StartDatetime

SELECT T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID
FROM T_SCHEDULEDEVENTS T1 WHERE T1.EventStatus = ? AND T1.StartDatetime <= ? AND
T1.SYS_OBJECTID > ? ORDER BY T1.StartDatetime;

-- LIST OF RECOMMENDED INDEXES
-- ============================
-- index[1]
CREATE INDEX TRIDATA.PERF1804201552470 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID
ASC, ENDDATETIME ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2]
CREATE INDEX TRIDATA.PERF1804201553110 ON TRIDATA.T_SCHEDULEDEVENTS (EVENTSTATUS
ASC, SYS_OBJECTID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3]
CREATE INDEX TRIDATA.PERF1804201629100 ON TRIDATA.T_SCHEDULEDEVENTS (SYS_OBJECTID
ASC, STARTDATETIME ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

Single-user manual test indexes

Long-running SQL: Long-running SQL queries are those that run longer than one second.

Steps were performed manually with the Performance Analyzer, which was activated for SQL logging. Long-running SQL queries were identified and analyzed by using the Db2 SQL Advisor. Then, recommended indexes were applied. If the Analyzer recommended indexes, but the improvement was less than 10%, those indexes were not applied and were not included.

Also, any recommended indexes against the IBS_SPEC and IBS_SPEC_ASSIGNMENTS tables were not applied because experience shows that indexes against these tables can be detrimental to overall system performance, especially under load.

Customizations:

In customized environments, you might need to modify the following recommended indexes to account for custom fields and specific database statistics in the database.

Also, identify the corresponding SQL queries in their environment and use the Db2 SQL Analyzer to verify that the recommended indexes are appropriate for their specific environment.

Lease abstract

```
Query 1:

SELECT T1.triUserMessageFlagTX AS T1_1049, T1.triNameTX AS T1_1045, T1.triFormLabelSY
AS T1_1058, T1.triCityTX AS T1_1136, T1.triStateProvTX AS T1_1134, T1.triCountryTX AS
T1_1133, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS
T1_SPEC_ID FROM M_LOCATION T1 WHERE T1.SYS_TYPE1 IN (10002586,10002100) AND T1.SYS_GUIID
IN (10002954,10003156,10018036,10019061) AND T1.SYS_PROJECTID = 1 AND ( (T1.triStatusCL !
= 'Retired' OR T1.triStatusCL IS NULL) AND (T1.triStatusCL != 'Upload Error' OR
T1.triStatusCL IS NULL) ) AND T1.SYS_OBJECTID > 0 ORDER BY T1.triNameTX, T1.triFormLabelSY,
T1.triCityTX, T1.triStateProvTX;

-- LIST OF RECOMMENDED INDEXES
-- ============================
-- index[1]
CREATE INDEX TRIDATA.PERF1804221912280 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
```

```
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2]
CREATE INDEX TRIDATA.PERF1804221916070 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3]
CREATE INDEX TRIDATA.PERF1804221918070 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, SYS_GUIID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[4]
CREATE INDEX TRIDATA.PERF1804221919560 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC, SYS_GUIID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[5]
CREATE INDEX TRIDATA.PERF1804221921350 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[6]
CREATE INDEX TRIDATA.PERF1804221923250 ON TRIDATA.T_TRIBUILDING (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC,
SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[7]
CREATE INDEX TRIDATA.PERF1804221925140 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[8]
CREATE INDEX TRIDATA.PERF1804221928490 ON TRIDATA.T_TRIPROPERTY (SYS_GUIID ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRICITYTX ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC,
SYS_TYPE1 ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX
ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

Query 2:

SELECT T1.triUserMessageFlagTX AS T1_1049, T1.triNameTX AS T1_1045, T1.triIdTX
AS T1_1044, T1.triParentPropertyTX AS T1_1081, T1.triParentBuildingTX AS T1_1075,
T1.triParentFloorTX AS T1_1080, T1.triFormLabelSY AS T1_1058, T1.triStatusCL
AS T1_1046, T1.triStatusCLObjId AS T1_1046_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM M_LOCATION T1 WHERE
T1.SYS_TYPE1 IN (10002585,10002873,10002586,10002100,10002582) AND T1.SYS_GUIID IN
(10002954,10003153,10003156,10018036,10019061,10003142) AND T1.SYS_PROJECTID = 1 AND
( (T1.triStatusCL != 'Retired' OR T1.triStatusCL IS NULL) AND(T1.triStatusCL != 'Upload
Error' OR T1.triStatusCL IS NULL) ) AND T1.SYS_OBJECTID > 0 ORDER BY T1.triNameTX,
T1.triIdTX;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.009MB
CREATE INDEX TRIDATA.PERF1804222031340 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL
ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.009MB
CREATE INDEX TRIDATA.PERF1804222035120 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL
ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3], 0.212MB
CREATE INDEX TRIDATA.PERF1804222037150 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC, TRISTATUSCL
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[4], 3.806MB
CREATE INDEX TRIDATA.PERF1804222038500 ON TRIDATA.T_TRIFLOOR (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC,
TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID
ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[5], 0.009MB
```

```
CREATE INDEX TRIDATA.PERF1804222040390 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX
ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[6], 3.837MB
CREATE INDEX TRIDATA.PERF1804222042280 ON TRIDATA.T_TRIBUILDING (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX
ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[7], 0.009MB
CREATE INDEX TRIDATA.PERF1804222044170 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX
ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC,
TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[8], 0.259MB
CREATE INDEX TRIDATA.PERF1804222047510 ON TRIDATA.T_TRIPROPERTY (SYS_GUIID ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_TYPE1 ASC, TRIPARENTPROPERTYTX ASC,
TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX
ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_LOCATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_LOCATIONGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBUILDING FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIBUILDINGGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINSTAGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILAND FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRILANDGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPLANNGUIID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPLANNGUIID5;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPOSEDSITE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPROPOGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIVERTIGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804221918070;
-- COMMIT WORK;
```

Generate schedules

```
Query 1:

SELECT T1.triUserMessageFlagTX AS T1_1005, T1.triClauseSectionCatego AS T1_1296,
T1.triClauseSectionCategoObjId AS T1_1296_OBJID, T1.triClauseTypeCL AS T1_1095,
T1.triClauseTypeCLObjId AS T1_1095_OBJID, T1.triGrantedInLeaseCL AS T1_1304,
T1.triGrantedInLeaseCLObjId AS T1_1304_OBJID, T1.triSectionTX AS T1_1067, T1.triPageTX
AS T1_1301, T1.triDescriptionTX AS T1_1026, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID
AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRILEASECLAUSE T1 WHERE T1.SYS_GUIID
= 10011868 AND ( (T1.triClauseSectionCatego NOT IN ('Rights','Landlord Rights','Tenant
Rights') OR T1.triClauseSectionCatego IS NULL) ) AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM
IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID = 69786165 AND ASS_TYPE = 'Has Covenant Clause') AND
T1.SYS_OBJECTID > 0 ORDER BY T1.triClauseSectionCatego, T1.triClauseTypeCL;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 6.036MB
CREATE UNIQUE INDEX TRIDATA.PERF1804222208420 ON TRIDATA.T_TRILEASECLAUSE (SPEC_ID
ASC) INCLUDE (SYS_GUIID, TRICLAUSESECTIONCATEGO, TRICLAUSETYPECL, SYS_OBJECTID,
SYS_TYPE1, TRIDESCRIPTIONTX, TRIPAGETX, TRISECTIONTX, TRIGRANTEDINLEASECLOBJID,
TRIGRANTEDINLEASECL, TRICLAUSETYPECLOBJID, TRICLAUSESECTIONCATEGOOBJID,
TRIUSERMESSAGEFLAGTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
```

```
COMMIT WORK;
-- index[2], 6.036MB
CREATE UNIQUE INDEX TRIDATA.PERF1804222214470 ON TRIDATA.T_TRILEASECLAUSE (SPEC_ID
ASC) INCLUDE (SYS_GUIID, SYS_OBJECTID, SYS_TYPE1, TRIDESCRIPTIONTX, TRIPAGETX,
TRISECTIONTX, TRIGRANTEDINLEASECLOBJID, TRIGRANTEDINLEASECL, TRICLAUSETYPECLOBJID,
TRICLAUSETYPECL, TRICLAUSESECTIONCATEGOOBJID, TRIUSERMESSAGEFLAGTX, TRICLAUSESECTIONCATEGO)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILEASECLAUSE FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRILEASECLAUSE;
-- COMMIT WORK;
```

Activate lease

No long-running SQL queries.

Data Loading: Indexes were applied for data loading that might improve lease schedule creation, accounting review, and lease activation.

Amend and extend lease

No long-running SQL queries.

Revise lease contract data

```
Query 1:

SELECT T1.triPaymentTypeCL AS T1_1091, T1.triPaymentTypeCLObjId AS T1_1091_OBJID,
T1.triStatusCL AS T1_1023, T1.triStatusCLObjId AS T1_1023_OBJID, T1.triExpectedAmountNU AS
T1_1088, T1.triActualAmountNU AS T1_1061, T2.triNameTX AS T2_1154_SID_1_7, T1.SYS_TYPE1
AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triCurrencyUO
AS T1_1143 FROM T_TRIPAYMENTLINEITEM T1 LEFT OUTER JOIN T_TRIREALESTATECONTRACT T2 ON
T1.IsPaymentForSysKey = T2.SPEC_ID WHERE T1.SYS_PROJECTID = 1 AND T1.SPEC_ID IN (SELECT
ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID = 66152947 AND ASS_TYPE = 'Has
Payment') AND T1.SYS_OBJECTID > 0;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 49.942MB
CREATE UNIQUE INDEX TRIDATA.PERF1804232123170 ON TRIDATA.T_TRIPAYMENTLINEITEM
(SPEC_ID ASC) INCLUDE (SYS_PROJECTID, ISPAYMENTFORSYSKEY, SYS_OBJECTID,
TRICURRENCYUO, TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL, TRIEXPECTEDAMOUNTNU,
TRIACTUALAMOUNTNU, TRISTATUSCLOBJID, TRISTATUSCL, SYS_TYPE1, SYS_GUIID) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPAYMENTLINEITEM;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;
```

Revise lease accounting data

```
Query 1:

SELECT T1.triPaymentTypeCL AS T1_1091, T1.triPaymentTypeCLObjId AS T1_1091_OBJID,
T1.triStatusCL AS T1_1023, T1.triStatusCLObjId AS T1_1023_OBJID, T1.triExpectedAmountNU AS
T1_1088, T1.triActualAmountNU AS T1_1061, T2.triNameTX AS T2_1154_SID_1_7, T1.SYS_TYPE1
AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triCurrencyUO
AS T1_1143 FROM T_TRIPAYMENTLINEITEM T1 LEFT OUTER JOIN T_TRIREALESTATECONTRACT T2 ON
T1.IsPaymentForSysKey = T2.SPEC_ID WHERE T1.SYS_PROJECTID = 1 AND T1.SPEC_ID IN (SELECT
ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID = 66152951 AND ASS_TYPE = 'Has
Payment') AND T1.SYS_OBJECTID > 0;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 49.942MB
```

```
CREATE UNIQUE INDEX TRIDATA.PERF1804232356320 ON TRIDATA.T_TRIPAYMENTLINEITEM
(SPEC_ID ASC) INCLUDE (SYS_PROJECTID, ISPAYMENTFORSYSKEY, SYS_OBJECTID,
TRICURRENCYUO, TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL, TRIEXPECTEDAMOUNTNU,
TRIACTUALAMOUNTNU, TRISTATUSCLOBJID, TRISTATUSCL, SYS_TYPE1, SYS_GUIID) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.079MB
CREATE UNIQUE INDEX TRIDATA.PERF1804232356370 ON TRIDATA.T_TRIREALESTATECONTRACT (SPEC_ID
ASC) INCLUDE (TRINAMETX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPAYMENTLINEITEM;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;
```

Expire lease

No long-running SQL queries.

Get payment

```
Query 1:

SELECT T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIPAYMENTLINEITEM
T1 LEFT OUTER JOIN T_TRIREALESTATECONTRACT T2 ON T1.IsPaymentForSysKey = T2.SPEC_ID
WHERE T1.SYS_GUIID = ? AND ( T1.triStatusCL = ? AND T1.triAccountingTypeLI = ? AND
T1.triPayProcessedModeLI = ? ) AND T1.triProcessPeriodCL = ? AND T1.triDueDA < ?
AND T1.SYS_OBJECTID > ? ORDER BY T2.triNameTX, T1.triDueDA, T1.triPaymentTypeCL,
T1.triExpectedTotalNU;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 230.341MB
CREATE INDEX PERF35.PERF_GETPAYMENTS ON PERF35.T_TRIPAYMENTLINEITEM (TRIPROCESSPERIODCL
ASC, TRISTATUSCL ASC, TRIACCOUNTINGTYPELI ASC, TRIPAYPROCESSEDMODELI ASC, SYS_GUIID
ASC, SYS_OBJECTID ASC, TRIDUEDA ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_SCHEDULEDEVENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804201552470;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_SCHEDULEDEVENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804201629100;
-- COMMIT WORK;
```

Multi-user benchmark test indexes

Db2 snapshots were taken to identify long-running and expensive, in terms of CPU use, SQL queries. Those were analyzed by using the Db2 SQL Advisor and recommended indexes were applied. If the Analyzer recommended indexes, but the improvement was less than 10%, those indexes were not applied and were not included.

In addition, any recommended indexes against the IBS_SPEC and IBS_SPEC_ASSIGNMENTS tables were not applied because experience shows that indexes against these tables can be detrimental to overall system performance, especially under load.

Customizations:

In customized environments, you might need to modify the following recommended indexes to account for custom fields and specific database statistics in the database.

Also, identify the corresponding SQL queries in their environment and use the Db2 SQL Analyzer to verify that the recommended indexes are appropriate for their specific environment.

Lease abstract draft

```
Queries 1, 2, & 3:

SELECT T1.triUserMessageFlagTX AS T1_1049, T1.triNameTX AS T1_1045, T1.triIdTX AS T1_1044,
T1.triParentPropertyTX AS T1_1081, T1.triParentBuildingTX AS T1_1075, T1.triParentFloorTX
AS T1_1080, T1.triFormLabelSY AS T1_1058, T1.triStatusCL AS T1_1046, T1.triStatusCLObjId
AS T1_1046_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID
AS T1_SPEC_ID FROM M_LOCATION T1 WHERE T1.SYS_TYPE1 IN (?,?,?,?,?) AND T1.SYS_GUIID IN
(?,?,?,?,?,?) AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL != ? OR T1.triStatusCL IS
NULL) AND (T1.triStatusCL != ? OR T1.triStatusCL IS NULL) ) AND T1.SYS_OBJECTID > ? AND
(T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY like '\Organizations\MyCompany9\%'))
OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\External Companies' or TRIPATHSY like '\Organizations\External
Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where
SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus Corporation' or TRIPATHSY
like '\Organizations\Focus Corporation\%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND
(T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Geography\North America\United States' or TRIPATHSY like '\Geography\North
America\United States\%')) OR T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX,
T1.triIdTX;

SELECT T1.triUserMessageFlagTX AS T1_1049, T1.triNameTX AS T1_1045, T1.triFormLabelSY
AS T1_1058, T1.triCityTX AS T1_1136, T1.triStateProvTX AS T1_1134, T1.triCountryTX
AS T1_1133, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID
AS T1_SPEC_ID FROM M_LOCATION T1 WHERE T1.SYS_TYPE1 IN (?,?) AND T1.SYS_GUIID IN
(?,?,?,?) AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL != ? OR T1.triStatusCL IS NULL)
AND (T1.triStatusCL != ? OR T1.triStatusCL IS NULL) ) AND T1.SYS_OBJECTID > ? AND
(T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY like '\Organizations\MyCompany9\%'))
OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\External Companies' or TRIPATHSY like '\Organizations\External
Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where
SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus Corporation' or TRIPATHSY
like '\Organizations\Focus Corporation\%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND
(T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Geography\North America\United States' or TRIPATHSY like '\Geography\North
America\United States\%')) OR T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX,
T1.triFormLabelSY, T1.triCityTX, T1.triStateProvTX;

SELECT T1.triUserMessageFlagTX AS T1_1049, T1.triNameTX AS T1_1045, T1.triIdTX
AS T1_1044, T1.triParentPropertyTX AS T1_1081, T1.triParentBuildingTX AS T1_1075,
T1.triParentFloorTX AS T1_1080, T1.triFormLabelSY AS T1_1058, T1.triStatusCL AS T1_1046,
T1.triStatusCLObjId AS T1_1046_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS
T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM M_LOCATION T1 WHERE T1.SYS_TYPE1 IN (?,?,?,?,?)
AND T1.SYS_GUIID IN (?,?,?,?,?,?) AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL != ?
OR T1.triStatusCL IS NULL) AND (T1.triStatusCL != ? OR T1.triStatusCL IS NULL) ) AND
UPPER(T1.triParentPropertyTX) LIKE ? AND T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY
= '\Organizations\MyCompany9' or TRIPATHSY like '\Organizations\MyCompany9\%')) OR
T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\External Companies' or TRIPATHSY like '\Organizations\External
Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where
SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus Corporation' or TRIPATHSY
like '\Organizations\Focus Corporation\%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND
(T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Geography\North America\United States' or TRIPATHSY like '\Geography\North
America\United States\%')) OR T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX,
T1.triIdTX;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.009MB
CREATE INDEX TRIDATA.PERF1806072214360 ON TRIDATA.T_TRIPLANNEDSPACE (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX
ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

COMMIT WORK;
-- index[2], 0.013MB
CREATE INDEX TRIDATA.PERF1806072216250 ON TRIDATA.T_TRILAND (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[3], 0.009MB
CREATE INDEX TRIDATA.PERF1806072218140 ON TRIDATA.T_TRIPLANNEDFLOOR (SYS_PROJECTID
```

```
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
SYS_TYPE1 ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX
ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

COMMIT WORK;
-- index[4], 0.224MB
CREATE INDEX TRIDATA.PERF1806072221430 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[5], 0.114MB
CREATE INDEX TRIDATA.PERF1806072223320 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[6], 0.009MB
CREATE INDEX TRIDATA.PERF1806072223410 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[7], 0.106MB
CREATE INDEX TRIDATA.PERF1806072227100 ON TRIDATA.T_TRIBUILDING (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[8], 0.009MB
CREATE INDEX TRIDATA.PERF1806072227190 ON TRIDATA.T_TRIVERTICALSHAFT (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[9], 0.298MB
CREATE INDEX TRIDATA.PERF1806072229080 ON TRIDATA.T_TRIPROPERTY (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[10], 0.013MB
CREATE INDEX TRIDATA.PERF1806072230570 ON TRIDATA.T_LOCATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[11], 0.013MB
CREATE INDEX TRIDATA.PERF1806072232460 ON TRIDATA.T_TRIPROPOSEDSITE (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1
ASC, TRIPARENTPROPERTYTX ASC, TRIPARENTFLOORTX ASC, TRIPARENTBUILDINGTX ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC, TRINAMETX ASC,
TRIIDTX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[12], 0.013MB
CREATE INDEX TRIDATA.PERF1806072238300 ON TRIDATA.T_TRILAND (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[13], 0.216MB
CREATE INDEX TRIDATA.PERF1806072243510 ON TRIDATA.T_TRISPACE (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID
ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
```

```
COMMIT WORK;
-- index[14], 0.110MB
CREATE INDEX TRIDATA.PERF1806072245400 ON TRIDATA.T_TRIFLOOR (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC,
TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID
ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[15], 0.009MB
CREATE INDEX TRIDATA.PERF1806072245490 ON TRIDATA.T_TRIINSTALLATION (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[16], 0.106MB
CREATE INDEX TRIDATA.PERF1806072249210 ON TRIDATA.T_TRIBUILDING (SYS_TYPE1 ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[17], 0.239MB
CREATE INDEX TRIDATA.PERF1806072251210 ON TRIDATA.T_TRIPROPERTY (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[18], 0.013MB
CREATE INDEX TRIDATA.PERF1806072253120 ON TRIDATA.T_LOCATION (SYS_PROJECTID ASC,
SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY ASC,
TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[19], 0.013MB
CREATE INDEX TRIDATA.PERF1806072255030 ON TRIDATA.T_TRIPROPOSEDSITE (SYS_PROJECTID
ASC, SYS_OBJECTID ASC, TRICITYTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, TRISTATEPROVTX ASC, TRICOUNTRYTX ASC, SYS_TYPE1 ASC, TRIFORMLABELSY
ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC, TRINAMETX ASC, SYS_GUIID ASC, SPEC_ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804221918070;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBUILDING FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804221923250;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPERTY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804221928490;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804222037150;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804222038500;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_LOCATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_LOCATIONOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILAND FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRILANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPOSEDSITE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPROPOOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRISPACEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
```

```
TRIDATA.PK_TRIVERTICALSHAFT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIINSTALLATION;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIPLANNEDFLOOR;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.P_TRIPLANNEDSPACE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSTALLATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIINSTAGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDFLOOR FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPLANNGUIID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPLANNEDSPACE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIPLANNGUIID5;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIVERTICALSHAFT FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_SYS_T_TRIVERTIGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBUILDING FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804222042280;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROPERTY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804222047510;
-- COMMIT WORK;

Query 4:

SELECT T1.triUserMessageFlagTX AS T1_1145, T1.triNameTX AS T1_1156, T1.triTitleTX AS
T1_1314, T2.triPathTX AS T2_1354_SID_1_71, T1.triWorkPhoneTX AS T1_1214, T1.triWorkFaxTX
AS T1_1213, T1.triEmailTX AS T1_1216, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS
T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIPEOPLE T1 LEFT OUTER JOIN T_ORGANIZATION
T2 ON T1.PrimaryOrganizationSysKey = T2.SPEC_ID WHERE T1.SYS_GUIID IN (?,?,?) AND
( (T1.triStatusCL != ? OR T1.triStatusCL IS NULL) ) AND UPPER(T1.triNameTX) LIKE ? AND
T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY
like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX, T1.triTitleTX, T2.triPathTX,
T1.triWorkPhoneTX;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.189MB
CREATE INDEX TRIDATA.PERF1806072327310 ON TRIDATA.T_TRIPEOPLE (SYS_GUIID ASC,
SYS_OBJECTID ASC, PRIMARYORGANIZATIONSYSKEY ASC, TRITITLETX ASC, TRISTATUSCL ASC,
TRIEMAILTX ASC, TRIWORKPHONETX ASC, TRIWORKFAXTX ASC, TRINAMETX ASC, TRIUSERMESSAGEFLAGTX
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 6.634MB
CREATE UNIQUE INDEX TRIDATA.PERF1806072328180 ON TRIDATA.T_ORGANIZATION (SPEC_ID ASC)
INCLUDE (TRIPATHTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPEOPLE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPEOPLEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPEOPLE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPEOPLEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PK_ORGANIZATION;
-- COMMIT WORK;
```

Generate rent schedule

```
Query 1:

SELECT T1.triExpirationYearQuart AS T1_1369, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID
AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM M_TRICONTRACT T1 WHERE T1.SYS_TYPE1 IN
(?,?) AND T1.SYS_GUIID IN (?,?) AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL NOT
IN ('Expired','History','Draft','Terminated','Retired','Upload Error') OR T1.triStatusCL
IS NULL) AND T1.triExpirationDA != ? and T1.triExpirationDA IS NOT NULL AND
T1.triExpirationDA != ? AND T1.triExpirationDA <= ? ) AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID
FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID IN (?,?)) AND T1.SYS_OBJECTID > ? AND
(T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY like '\Organizations\MyCompany9\%'))
OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\External Companies' or TRIPATHSY like '\Organizations\External
Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where
SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus Corporation' or TRIPATHSY
like '\Organizations\Focus Corporation\%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND
(T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID
> 0 and (TRIPATHSY = '\Geography\North America\United States' or TRIPATHSY like
'\Geography\North America\United States\%')) OR T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY
T1.triExpirationYearQuart;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.017MB
CREATE INDEX TRIDATA.PERF1806080059310 ON TRIDATA.T_TRILEASEABSTRACT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GUIID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.032MB
CREATE INDEX TRIDATA.PERF1806080101330 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID
ASC, SYS_PROJECTID ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART
ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3], 0.181MB
CREATE INDEX TRIDATA.PERF1806080101410 ON TRIDATA.T_TRIPURCHASEREQUISITION (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[4], 0.017MB
CREATE INDEX TRIDATA.PERF1806080103340 ON TRIDATA.T_TRISTANDARDCONTRACT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[5], 0.009MB
CREATE INDEX TRIDATA.PERF1806080105280 ON TRIDATA.T_TRIASSETLEASE (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[6], 0.013MB
CREATE INDEX TRIDATA.PERF1806080107220 ON TRIDATA.T_TRIWARRANTY (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[7], 0.013MB
CREATE INDEX TRIDATA.PERF1806080111080 ON TRIDATA.T_TRIPURCHASEORDER (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[8], 0.032MB
CREATE INDEX TRIDATA.PERF1806080113010 ON TRIDATA.T_TRISPACEUSEAGREEMENT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[9], 0.009MB
CREATE INDEX TRIDATA.PERF1806080116470 ON TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER
(SYS_PROJECTID ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
```

```
COMMIT WORK;
-- index[10], 0.009MB
CREATE INDEX TRIDATA.PERF1806080120340 ON TRIDATA.T_TRIPRIMECONTRACT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[11], 0.013MB
CREATE INDEX TRIDATA.PERF1806080122280 ON TRIDATA.T_TR_STAND_CONTR_CHAN_ORD
(SYS_PROJECTID ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, SPEC_ID
ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[12], 0.009MB
CREATE INDEX TRIDATA.PERF1806080124220 ON TRIDATA.T_TRICONTRACT (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[13], 0.009MB
CREATE INDEX TRIDATA.PERF1806080126160 ON TRIDATA.T_CSTDEFAULTRETENTION (SYS_PROJECTID
ASC, TRIEXPIRATIONDA ASC, SYS_OBJECTID ASC, TRIEXPIRATIONYEARQUART ASC, TRISTATUSCL
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SYS_GUIID ASC,
SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806072106180;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICOUNTRY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRICOUNTRYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINSURANCE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINSUROBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRILEASEABSTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRILEASEOBJECTID3;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPROCUREMENTCARD FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPROCUOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEREQUISITION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID3;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISPACEUSEAGREEMENT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISPACEOBJECTID26;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTANDARDCONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISTANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTATE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRISTATEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIWARRANTY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIWARRANTYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TR_STAND_CONTR_CHAN_ORD FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TR_STANDOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
```

```
-- RUNSTATS ON TABLE TRIDATA.T_TRIASSETLEASE FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIASSETLEASE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBLANKETPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIBLANKETPURCHASEORDER;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRICONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPRIMECONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIPRIMECONTRACTCHANGEORDER;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_CSTDEFAULTRETENTION FOR SAMPLED DETAILED INDEX
TRIDATA.P_CSTDEFAULTRETENTION;
-- COMMIT WORK;

Query 2:

SELECT REP_TEMPLATE_ID from REP_TEMPLATE_HDR where CLASS_TYPE_ID=? and OBJECT_TEMPLATE_ID=?
and REP_NAME=?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.204MB
CREATE INDEX TRIDATA.PERF1806080128380 ON TRIDATA.REP_TEMPLATE_HDR (REP_NAME
ASC, OBJECT_TEMPLATE_ID ASC, CLASS_TYPE_ID ASC, REP_TEMPLATE_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
None

Query 3:

SELECT SOURCE_BO_ID, ASSOCIATION_VERB, PROJ_CONTAINMENT_DISABLED, REV_ASSOCIATION_VERB FROM
MODULE_ASSOCIATION WHERE ASS_BO_ID=? AND DEPENDENT_FLAG=? AND CASCADE_READ_ONLY=?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.325MB
CREATE INDEX TRIDATA.PERF1806080129090 ON TRIDATA.MODULE_ASSOCIATION (ASS_BO_ID
ASC, DEPENDENT_FLAG ASC, CASCADE_READ_ONLY ASC, REV_ASSOCIATION_VERB
ASC, PROJ_CONTAINMENT_DISABLED ASC, ASSOCIATION_VERB ASC, SOURCE_BO_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
None
```

Activate lease

```
Query 1:

SELECT T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIAPPROVAL T1
WHERE T1.SYS_GUIID = ? AND T1.triRecordIdSY = ? AND T1.SYS_OBJECTID > ? ORDER BY
T1.triLinkedBusinessObj, T1.triLinkedRecordTX, T1.triSubmittedByTX, T1.triStatusCL;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 25.528MB
CREATE INDEX TRIDATA.PERF1806071659590 ON TRIDATA.T_TRIAPPROVAL (TRIRECORDIDSY ASC,
SYS_GUIID ASC, TRILINKEDBUSINESSOBJ ASC, TRILINKEDRECORDTX ASC, TRISUBMITTEDBYTX ASC,
TRISTATUSCL ASC, SYS_OBJECTID ASC, SPEC_ID ASC, SYS_TYPE1 ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRIAPPROVAL FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIAPPROVALGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIAPPROVAL FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIAPPROVALOBJECTID1;
-- COMMIT WORK;
```

Complete lease abstract

No long-running or CPU-expensive SQL queries.

Amend lease

```
Query 1:

SELECT T1.triUserMessageFlagTX AS T1_1102, T1.triNameTX AS T1_1098, T1.triIdTX AS
T1_1097, T1.triFormLabelSY AS T1_1111, T1.triStatusCL AS T1_1099, T1.triStatusCLObjId
AS T1_1099_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID
AS T1_SPEC_ID FROM T_ORGANIZATION T1 WHERE T1.SYS_GUIID IN (?,?,?,?,?,?,?,?,?) AND
T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY
like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX, T1.triIdTX, T1.triStatusCL;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.392MB
CREATE INDEX TRIDATA.PERF1806071724280 ON TRIDATA.T_ORGANIZATION (SYS_GUIID ASC,
SYS_OBJECTID ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.013MB
CREATE INDEX TRIDATA.PERF1806071721520 ON TRIDATA.T_TRICITY (SYS_OBJECTID ASC, TRIPATHSY
ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3], 0.384MB
CREATE UNIQUE INDEX TRIDATA.PERF1806071724500 ON TRIDATA.T_ORGANIZATION (SPEC_ID ASC)
INCLUDE (SYS_OBJECTID, TRIPATHSY) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_ORGANIZAOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PK_ORGANIZATION;
-- COMMIT WORK;
```

Lease accounting review

No long-running or CPU-expensive SQL queries.

Revise lease contract data

```
Query 1:

SELECT T1.triPaymentTypeCL AS T1_1091, T1.triPaymentTypeCLObjId AS T1_1091_OBJID,
T1.triAccountingTypeLI AS T1_1059, T1.triExpectedAmountNU AS T1_1088, T1.triStatusCL
AS T1_1023, T1.triStatusCLObjId AS T1_1023_OBJID, T1.triExpectedTotalNU AS T1_1169,
T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID,
T1.triCurrencyUO AS T1_1143 FROM T_TRIPAYMENTLINEITEM T1 WHERE ( T1.triAccountingTypeLI = ?
AND T1.triStatusCL = ? AND T1.triSummaryTypeCL = ? AND T1.triDueDA <= ? ) AND T1.SPEC_ID
IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID = ? AND ASS_TYPE = ?) AND
T1.SYS_OBJECTID > ?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 8.517MB
CREATE UNIQUE INDEX TRIDATA.PERF1806071945040 ON TRIDATA.T_TRIPAYMENTLINEITEM
(SPEC_ID ASC) INCLUDE (TRISUMMARYTYPECL, TRISTATUSCL, TRIACCOUNTINGTYPELI,
TRIDUEDA, SYS_OBJECTID, TRICURRENCYUO, SYS_GUIID, SYS_TYPE1, TRIEXPECTEDTOTALNU,
TRISTATUSCLOBJID, TRIEXPECTEDAMOUNTNU, TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
```

```
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804232123170;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;

Query 2:

SELECT T1.triPaymentTypeCL AS T1_1091, T1.triPaymentTypeCLObjId AS T1_1091_OBJID,
T1.triAccountingTypeLI AS T1_1059, T1.triActualAmountNU AS T1_1061, T1.triStatusCL
AS T1_1023, T1.triStatusCLObjId AS T1_1023_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triCurrencyUO AS T1_1143 FROM
T_TRIPAYMENTLINEITEM T1 WHERE ( T1.triAccountingTypeLI = ? AND T1.triStatusCL = ? AND
T1.triSummaryTypeCL = ? ) AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS
WHERE SPEC_ID = ? AND ASS_TYPE = ?) AND T1.SYS_OBJECTID > ?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 8.517MB
CREATE UNIQUE INDEX TRIDATA.PERF1806071948000 ON TRIDATA.T_TRIPAYMENTLINEITEM
(SPEC_ID ASC) INCLUDE (TRISUMMARYTYPECL, TRISTATUSCL, TRIACCOUNTINGTYPELI,
SYS_OBJECTID, TRICURRENCYUO, SYS_GUIID, SYS_TYPE1, TRISTATUSCLOBJID, TRIACTUALAMOUNTNU,
TRIPAYMENTTYPECLOBJID, TRIPAYMENTTYPECL) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRIPAYMENTLINEITEM FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1804232123170;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
```

Revise lease accounting data

Indexes: Revise Lease Accounting Data

```
Query 1:

SELECT NAV_ITEM_ID FROM NAV_ITEM WHERE UI_TARGET_ID IN (SELECT UI_TARGET_ID FROM UI_TARGET
WHERE PRIMARY_OBJECT_ID = ?);

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.021MB
CREATE INDEX TRIDATA.PERF1806071623390 ON TRIDATA.UI_TARGET (PRIMARY_OBJECT_ID
ASC, UI_TARGET_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.017MB
CREATE INDEX TRIDATA.PERF1806071623470 ON TRIDATA.NAV_ITEM (UI_TARGET_ID ASC,
NAV_ITEM_ID DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

Expire lease

```
Query 1:

SELECT T1.triUserMessageFlagTX AS T1_1161, T1.triImageIM AS T1_1293, T1.triIdTX AS
T1_1153, T1.triNameTX AS T1_1163, T1.triCityTX AS T1_1795, T1.triStateProvTX AS T1_1798,
T1.triCountryTX AS T1_1884, T1.triExpirationDA AS T1_1172, T1.triContractRentableNU
AS T1_1251, T1.triContractStatusCL AS T1_2052, T1.triContractStatusCLObjId AS
T1_2052_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS
T1_SPEC_ID, T1.triAreaUO AS T1_1891 FROM T_TRIREALESTATECONTRACT T1 WHERE T1.SYS_GUIID
= ? AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL NOT IN ('Retired','Upload
Error','History','Deleted','Terminated','Expired') OR T1.triStatusCL IS NULL) ) AND
T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY
like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX, T1.triCityTX, T1.triStateProvTX;
```

```
-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.892MB
CREATE INDEX TRIDATA.PERF1806072038510 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID
ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRICONTRACTSTATUSCLOBJID ASC, TRICONTRACTSTATUSCL
ASC, TRIAREAUO ASC, TRICOUNTRYTX ASC, TRISTATEPROVTX ASC, TRICITYTX ASC, TRIIMAGEIM
ASC, TRICONTRACTRENTABLENU ASC, TRIEXPIRATIONDA ASC, TRINAMETX ASC, TRISTATUSCL
ASC, TRIUSERMESSAGEFLAGTX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID
ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;

Query 2:

SELECT T1.triImageIM AS T1_1293, T1.triIdTX AS T1_1153, T1.triNameTX AS T1_1163,
T1.triCityTX AS T1_1795, T1.triStateProvTX AS T1_1798, T1.triCountryTX AS T1_1884,
T1.triExpirationDA AS T1_1172, T1.triContractRentableNU AS T1_1251, T1.triContractStatusCL
AS T1_2052, T1.triContractStatusCLObjId AS T1_2052_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1,
T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triAreaUO AS T1_1891 FROM
T_TRIREALESTATECONTRACT T1 WHERE T1.SYS_OBJECTSTATE = ? AND T1.SYS_GUIID = ? AND
T1.SYS_PROJECTID = ? AND T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID
from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or
TRIPATHSY like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID
from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triNameTX, T1.triCityTX, T1.triStateProvTX;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.892MB
CREATE INDEX TRIDATA.PERF1806072042370 ON TRIDATA.T_TRIREALESTATECONTRACT
(SYS_OBJECTSTATE ASC, SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC,
TRICONTRACTSTATUSCLOBJID ASC, TRICONTRACTSTATUSCL ASC, TRIAREAUO ASC, TRICOUNTRYTX
ASC, TRISTATEPROVTX ASC, TRICITYTX ASC, TRIIMAGEIM ASC, TRICONTRACTRENTABLENU
ASC, TRIEXPIRATIONDA ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;

Queries 3, 4, & 5:
```

```
SELECT COUNT(1) FROM T_TRIREALESTATECONTRACT T1 WHERE T1.SYS_OBJECTSTATE = ? AND
T1.SYS_GUIID = ? AND T1.SYS_PROJECTID = ? AND T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY
= '\Organizations\MyCompany9' or TRIPATHSY like '\Organizations\MyCompany9\%')) OR
T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Organizations\External Companies' or TRIPATHSY like '\Organizations\External
Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION where
SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus Corporation' or TRIPATHSY
like '\Organizations\Focus Corporation\%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND
(T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and
(TRIPATHSY = '\Geography\North America\United States' or TRIPATHSY like '\Geography\North
America\United States\%')) OR T1.SYS_GEOGRAPHYNAMEOBJID IS NULL);

SELECT COUNT(1) FROM T_TRIREALESTATECONTRACT T1 WHERE T1.SYS_GUIID = ?
AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL NOT IN ('Retired','Upload
Error','History','Deleted','Terminated', 'Expired') OR T1.triStatusCL IS NULL) ) AND
T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY
like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL);

SELECT COUNT(1) FROM T_TRIREALESTATECONTRACT T1 WHERE T1.SYS_OBJECTSTATE = ? AND
T1.SYS_GUIID = ? AND T1.SYS_PROJECTID = ? AND UPPER(T1.triNameTX) LIKE ? AND
T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY
like '\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External
Companies' or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID
IN (select SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY =
'\Organizations\Focus Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\
%')) OR T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select
SPEC_ID from M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North
America\United States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL);

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.560MB
CREATE INDEX TRIDATA.PERF1806072102530 ON TRIDATA.T_TRIREALESTATECONTRACT
(SYS_OBJECTSTATE ASC, SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC,
SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.521MB
CREATE INDEX TRIDATA.PERF1806072106180 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_GUIID
ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, SYS_GEOGRAPHYNAMEOBJID
ASC, SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3], 1.036MB
CREATE INDEX TRIDATA.PERF1806072109460 ON TRIDATA.T_TRIREALESTATECONTRACT
(SYS_OBJECTSTATE ASC, SYS_GUIID ASC, SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRINAMETX
ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIREALEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;
```

Period close (including journal entries)

```
Query 1:

SELECT SPEC_ID FROM M_TRIPEOPLE WHERE triRecordNameTX = ?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.985MB
CREATE INDEX TRIDATA.PERF1806072121060 ON TRIDATA.T_TRIPEOPLE (TRIRECORDNAMETX ASC,
SPEC_ID DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
None

Query 2:

SELECT T1.triUserMessageFlagTX AS T1_1111, T1.triIdTX AS T1_1094, T1.triDateDA AS T1_1168,
T1.triNameTX AS T1_1096, T1.triStatusCL AS T1_1109, T1.triStatusCLObjId AS T1_1109_OBJID,
T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM
T_TRIINVOICE T1 WHERE T1.SYS_GUIID = ? AND T1.SYS_PROJECTID = ? AND ( (T1.triStatusCL NOT
IN ('Retired','Upload Error','Template','History','Deleted') OR T1.triStatusCL IS NULL) )
AND T1.SYS_OBJECTID > ? AND (T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from T_ORGANIZATION
where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\MyCompany9' or TRIPATHSY like
'\Organizations\MyCompany9\%')) OR T1.SYS_ORGNAMEOBJID IN (select SPEC_ID from
T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\External Companies'
or TRIPATHSY like '\Organizations\External Companies\%')) OR T1.SYS_ORGNAMEOBJID IN (select
SPEC_ID from T_ORGANIZATION where SYS_OBJECTID > 0 and (TRIPATHSY = '\Organizations\Focus
Corporation' or TRIPATHSY like '\Organizations\Focus Corporation\%')) OR
T1.SYS_ORGNAMEOBJID IS NULL) AND (T1.SYS_GEOGRAPHYNAMEOBJID IN (select SPEC_ID from
M_GEOGRAPHY where SYS_OBJECTID > 0 and (TRIPATHSY = '\Geography\North America\United
States' or TRIPATHSY like '\Geography\North America\United States\%')) OR
T1.SYS_GEOGRAPHYNAMEOBJID IS NULL) ORDER BY T1.triIdTX, T1.triDateDA, T1.triNameTX,
T1.triStatusCL;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.825MB
CREATE INDEX TRIDATA.PERF1806072126500 ON TRIDATA.T_TRIINVOICE (SYS_PROJECTID ASC,
SYS_GUIID ASC, SYS_OBJECTID ASC, TRIDATEDA ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID
ASC, TRISTATUSCL ASC, TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC,
SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED
DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_GEOGRAPHY FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_GEOGRAPHYOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINVOICE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINVOICEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIINVOICE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIINVOICEOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICITY FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071721520;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_ORGANIZATION FOR SAMPLED DETAILED INDEX
TRIDATA.PERF1806071724500;
-- COMMIT WORK;

Query 3:

SELECT T1.triUserMessageFlagTX AS T1_1111, T1.triIdTX AS T1_1094, T3.triIdTX AS
T3_1359_SID_1_31, T3.triNameTX AS T3_1357_SID_1_31, T2.triBusinessObjectLab AS T2_1150,
T1.triStatusCL AS T1_1109, T1.triStatusCLObjId AS T1_1109_OBJID, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T2.SPEC_ID AS T2_SPEC_ID, T2.SYS_TYPE1 AS
T2_SYS_TYPE1, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIREINVOICE T1 LEFT OUTER JOIN
IBS_SPEC_ASSIGNMENTS T4 ON T1.SPEC_ID = T4.SPEC_ID AND T4.ASS_SPEC_CLASS_TYPE = ? LEFT
OUTER JOIN M_TRICONTRACT T2 ON T4.ASS_SPEC_ID = T2.SPEC_ID AND T2.SYS_OBJECTID > 0 AND
T2.SYS_TYPE1 IN (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) LEFT OUTER JOIN T_TRIREALESTATECONTRACT
T3 ON T1.HasContractSysKey = T3.SPEC_ID WHERE T1.SYS_GUIID IN (?,?) AND ( (T1.triStatusCL
NOT IN ('Retired','Upload Error','Template','History','Deleted') OR T1.triStatusCL IS
NULL) ) AND T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID = ?)
AND T1.SYS_OBJECTID > ? ORDER BY T1.triIdTX;

-- LIST OF RECOMMENDED INDEXES
```

```
-- ============================
-- index[1], 0.091MB
CREATE INDEX TRIDATA.PERF1806072131470 ON TRIDATA.T_TRILEASEABSTRACT (SYS_TYPE1
ASC, SYS_OBJECTID ASC, TRIBUSINESSOBJECTLABEL2 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.231MB
CREATE INDEX TRIDATA.PERF1806072132150 ON TRIDATA.T_TRIREALESTATECONTRACT (SYS_TYPE1
ASC, SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[3], 0.056MB
CREATE INDEX TRIDATA.PERF1806072132430 ON TRIDATA.T_TRIPURCHASEREQUISITION (SYS_TYPE1
ASC, SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[4], 0.013MB
CREATE INDEX TRIDATA.PERF1806072132450 ON TRIDATA.T_TRISTANDARDCONTRACT (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[5], 0.009MB
CREATE INDEX TRIDATA.PERF1806072133130 ON TRIDATA.T_TRIASSETLEASE (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[6], 0.013MB
CREATE INDEX TRIDATA.PERF1806072134090 ON TRIDATA.T_TRIBLANKETPURCHASEORDER (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[7], 0.017MB
CREATE INDEX TRIDATA.PERF1806072135310 ON TRIDATA.T_TRISPACEUSEAGREEMENT (SYS_TYPE1
ASC, SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[8], 0.009MB
CREATE INDEX TRIDATA.PERF1806072136010 ON TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER
(SYS_OBJECTID ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[9], 0.009MB
CREATE INDEX TRIDATA.PERF1806072136570 ON TRIDATA.T_TRIPRIMECONTRACT (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[10], 0.009MB
CREATE INDEX TRIDATA.PERF1806072137530 ON TRIDATA.T_TRICONTRACT (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLAB ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[11], 0.009MB
CREATE INDEX TRIDATA.PERF1806072138210 ON TRIDATA.T_CSTDEFAULTRETENTION (SYS_OBJECTID
ASC, TRIBUSINESSOBJECTLABEL1 ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS
COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[12], 0.044MB
CREATE INDEX TRIDATA.PERF1806072130330 ON TRIDATA.T_TRIREINVOICE (SYS_GUIID ASC,
SYS_OBJECTID ASC, HASCONTRACTSYSKEY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC,
TRISTATUSCL ASC, TRIIDTX ASC, SYS_TYPE1 ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[13], 0.669MB
CREATE UNIQUE INDEX TRIDATA.PERF1806072138490 ON TRIDATA.T_TRIREALESTATECONTRACT (SPEC_ID
ASC) INCLUDE (TRINAMETX, TRIIDTX) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.IDX03_IBS_SPEC_ASSIGN;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_CSTDEFAULTRETENTION FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_CSTDEFAUGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIASSETLEASE FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIASSETLEASEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIBLANKETPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIBLANKOBJECTID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTRACT FOR SAMPLED DETAILED INDEX
```

```
TRIDATA.IDX_SYS_T_TRICONTRACTGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPRIMEGUIID1;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPRIMECONTRACTCHANGEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPRIMEGUIID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PERF_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREALESTATECONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREALESTATECONTRACT;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIREINVOICE FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRIREINVOICE;
-- COMMIT WORK;

Query 4:

SELECT T1.triIdTX AS T1_1153, T1.triNameTX AS T1_1163, T1.triToOrganizationTX AS T1_1326,
T1.triToOrganizationTXObjId AS T1_1326_OBJID, T1.triCommitmentOrigina AS T1_1165,
T1.triPendingChangesRol AS T1_1176, T1.triCommitmentChanges AS T1_1164, T1.triStatusCL AS
T1_1162, T1.triStatusCLObjId AS T1_1162_OBJID, T1.SYS_TYPE1 AS T1_SYS_TYPE1, T1.SYS_GUIID
AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triCurrencyUO AS T1_1309 FROM
T_TRISTANDARDCONTRACT T1 WHERE T1.SYS_GUIID = ? AND ( (T1.triStatusCL NOT IN
('Retired','Upload Error','Template', 'History','Deleted') OR T1.triStatusCL IS NULL) ) AND
T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID IN (SELECT
T1.SPEC_ID AS T1_SPEC_ID FROM T_TRICONTACTROLE T1 WHERE ( (T1.triStatusCL NOT IN
('Template','History','Retired','Deleted','Upload Error') OR T1.triStatusCL IS NULL) ) AND
T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID IN (?,?) AND
ASS_TYPE = ?) AND T1.SYS_OBJECTID > ?) AND ASS_TYPE = ?) AND T1.SYS_OBJECTID > ? ORDER BY
T1.triIdTX, T1.triNameTX, T1.triToOrganizationTX;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.200MB
CREATE UNIQUE INDEX TRIDATA.PERF1806072141080 ON TRIDATA.T_TRICONTACTROLE (SPEC_ID
ASC) INCLUDE (SYS_OBJECTID, TRISTATUSCL) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;
-- index[2], 0.013MB
CREATE UNIQUE INDEX TRIDATA.PERF1806072141250 ON TRIDATA.T_TRISTANDARDCONTRACT (SPEC_ID
ASC) INCLUDE (SYS_GUIID, TRIIDTX, TRINAMETX, TRITOORGANIZATIONTXOBJID,
TRISTATUSCL, TRISTATUSCLOBJID, TRIPENDINGCHANGESROL, TRICOMMITMENTORIGINA,
TRICURRENCYUO, TRITOORGANIZATIONTX, SYS_TYPE1, TRICOMMITMENTCHANGES, SYS_OBJECTID) ALLOW
REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTACTROLE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_TRICONTACTROLE;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.PK_IBS_SPEC_ASSIGNMENTS;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRISTANDARDCONTRACT FOR SAMPLED DETAILED INDEX
TRIDATA.PK_TRISTANDARDCONTRACT;
-- COMMIT WORK;

Query 5:

SELECT T1.triIdTX AS T1_1153, T1.triNameTX AS T1_1163, T1.triVendorCompanyLook AS T1_1408,
T1.triVendorCompanyLookObjId AS T1_1408_OBJID, T1.triLineItemTotalNU AS T1_1463,
T1.triStatusCL AS T1_1162, T1.triStatusCLObjId AS T1_1162_OBJID, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID, T1.triCurrencyUO AS
T1_1367 FROM T_TRIPURCHASEORDER T1 WHERE T1.SYS_GUIID = ? AND ( (T1.triStatusCL NOT IN
('Retired','Upload Error','Template', 'History','Deleted') OR T1.triStatusCL IS NULL) ) AND
T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID IN (SELECT
T1.SPEC_ID AS T1_SPEC_ID FROM T_TRICONTACTROLE T1 WHERE ( (T1.triStatusCL NOT IN
('Template','History','Retired','Deleted','Upload Error') OR T1.triStatusCL IS NULL) ) AND
T1.SPEC_ID IN (SELECT ASS_SPEC_ID FROM IBS_SPEC_ASSIGNMENTS WHERE SPEC_ID IN (?,?) AND
ASS_TYPE = ?) AND T1.SYS_OBJECTID > ?) AND ASS_TYPE = ?) AND T1.SYS_OBJECTID > ? ORDER BY
T1.triIdTX, T1.triNameTX, T1.triVendorCompanyLook;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.013MB
CREATE INDEX TRIDATA.PERF1806072144040 ON TRIDATA.T_TRIPURCHASEORDER (SYS_GUIID
ASC, SYS_OBJECTID ASC, TRICURRENCYUO ASC, SYS_TYPE1 ASC, TRISTATUSCLOBJID
ASC, TRILINEITEMTOTALNU ASC, TRIVENDORCOMPANYLOOKOBJID ASC, TRIVENDORCOMPANYLOOK ASC,
```

```
TRINAMETX ASC, TRIIDTX ASC, SPEC_ID ASC, TRISTATUSCL ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.IBS_SPEC_ASSIGNMENTS FOR SAMPLED DETAILED INDEX
TRIDATA.ASS_CTYP_TMPL_SPID;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRIPURCHASEORDER FOR SAMPLED DETAILED INDEX
TRIDATA.IDX_SYS_T_TRIPURCHOBJECTID2;
-- COMMIT WORK;
-- RUNSTATS ON TABLE TRIDATA.T_TRICONTACTROLE FOR SAMPLED DETAILED INDEX
TRIDATA.PERF01_TRICONTACTROLE;
-- COMMIT WORK;

Query 6:

SELECT T1.triStatusCL AS T1_1054, T1.triStatusCLObjId AS T1_1054_OBJID, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIJOURNALENTRY
T1 WHERE T1.SYS_OBJECTID >= ? AND T1.SPEC_ID IN
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) AND
T1.triStatusCL = ?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.118MB
CREATE INDEX TRIDATA.PERF1808180010570 ON TRIDATA.T_TRIJOURNALENTRY (TRISTATUSCL
ASC, SYS_OBJECTID ASC, SPEC_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED
STATISTICS;
COMMIT WORK;

Query 7:

SELECT MIN(INSTANCE_ID) AS INSTANCE_ID, WFE.BO_ID, WFE.USER_ID FROM WF_EVENT WFE WHERE
WFE.AGENT_ID = -1 AND NOT EXISTS (SELECT 'X' FROM WF_EVENT WFE2 WHERE WFE2.AGENT_ID > -1
AND WFE2.BO_ID = WFE.BO_ID) AND WFE.USER_ID NOT IN (?,?) GROUP BY WFE.BO_ID, WFE.USER_ID
ORDER BY 1;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 0.118MB
CREATE INDEX TRIDATA.PERF1808180017030 ON TRIDATA.WF_EVENT (AGENT_ID ASC) ALLOW REVERSE
SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
-- index[2], 0.649MB
CREATE INDEX TRIDATA.PERF1808180017130 ON TRIDATA.WF_EVENT (AGENT_ID ASC, BO_ID DESC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;

Query 8:

SELECT T1.triStatusCL AS T1_1054, T1.triStatusCLObjId AS T1_1054_OBJID, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM T_TRIJOURNALENTRY
T1 WHERE T1.SYS_OBJECTID >= ? AND T1.SPEC_ID IN
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, ?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) AND
T1.triStatusCL = ?;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 3.259MB
CREATE INDEX TRIDATA.PERF1808180022250 ON TRIDATA.T_TRIJOURNALENTRY (SYS_OBJECTID ASC,
SPEC_ID DESC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
COMMIT WORK;
```

Finance lease cash-flow disclosure report

```
Query 1:

SELECT distinct APPLICATION_ID from APP_OBJECT_PERMISSION where company_id = ?
and APPLICATION_ID in (-1,22021,6,7,8,9,10,11,13,15,16,23825,17,18,19,20,21,25,26,27,28,
23325,29,23326,30,23327,31,23328,32,23329,35,20004,37,20006,39,20007,20519,20008,40,22825,
```

```
20010,20011,44,45,20013,46,20015,48,49,50,22323,22324,53,22325,54,57,58,60,61,24125,23625,
23626,20819,20820,23125,22623,20319,22624,22625,22626,23925,21119,23425,20619,20620,22925,
20621,22926,22423,20119,20120,20121,20124,20125,24225,24226,23725,23225,22723,20419,22724,
22725,22730,22731,22223,22224,24025,21219,23527,20719,23025,23026,22523) and GROUP_ID in
(-1,0,89666588,89669132,89665760,89667300) and service_id > 1;

-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1], 1.415MB
CREATE INDEX TRIDATA.PERF1806112022490 ON TRIDATA.APP_OBJECT_PERMISSION (GROUP_ID
ASC, COMPANY_ID ASC, SERVICE_ID ASC, APPLICATION_ID ASC) ALLOW REVERSE SCANS COLLECT
SAMPLED DETAILED STATISTICS;
COMMIT WORK;

-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE TRIDATA.APP_OBJECT_PERMISSION FOR SAMPLED DETAILED INDEX
TRIDATA.PK_APP_OBJECT_PERMISSION;
-- COMMIT WORK;
```

Oracle database

Tune the Oracle database server, and then add the platform, reserve, lease, and hierarchy indexes.

Tune the database server

Specify the following initialization parameters:

CURSOR_SHARING
Set **CURSOR_SHARING** to the default value EXACT to optimize performance. EXACT ensures that TRIRIGA uses bind variables extensively.

NLS_LENGTH_SEMANTICS
If you require multibyte support, set **NLS_LENGTH_SEMANTICS** to CHAR and set the database character set to AL32UTF8 or AL16UTF16.

WORKAREA_SIZE_POLICY
Use the Program Global Area (PGA) setting **WORKAREA_SIZE_POLICY=AUTO** to automatically size work areas.

OPTIMIZER_FEATURES_ENABLE
Set **OPTIMIZER_FEATURES_ENABLE** to your current Oracle version.

PARALLEL_DEGREE_POLICY
Specifies whether automatic degree of parallelism, statement queuing, and in-memory parallel execution are enabled. Change the default of MANUAL to ADAPTIVE.

PROCESSES=
Increase the process number to handle more concurrent users by setting the **PROCESSES=** initialization parameter to the maximum number of users who can access Oracle concurrently. A guideline is to add up all maximum Java Database Connectivity (JDBC) connection pool values from each Java virtual machine (JVM) and add another 100 for Oracle. For example, adding 200 for an application server, plus 200 for a process server, plus 100 for Oracle requires a **PROCESSES=** value of at least 500.

Automatic memory management

In Oracle 11g and 12c, automatic memory management automatically readjusts the following main pool sizes based on existing workloads:

- db_cache_size
- shared_pool_size
- large_pool_size
- java_pool_size

The **SGA_TARGET** and **SGA_MAX_SIZE** initialization parameters enable automatic memory management.

The **memory_target** parameter is dynamic. You can change **memory_target** by using the **alter system** command, which deallocates RAM from an instance's System Global Area (SGA) or Program Global Area (PGA) and reallocates that RAM to another instance. The **memory_target** and **MEMORY_MAX_TARGET** initialization parameters enable this feature. Consult with your database administrator to size these parameters

For more information about automatic memory management, see Enabling Automatic Memory Management.

For more information about tuning the Oracle database, see the Oracle Database Documentation Library.

Oracle RAC considerations

Many smaller nodes in a Real Application Clusters (RAC) configuration might yield better performance and might be suitable for high availability. However, small system-based RAC impairs performance. To scale Oracle for performance, use vertical scalability by adding CPU, RAM, and disk to the individual nodes.

RAC requires more resources because it maintains the cluster state across all the nodes. RAC also serves the data to the applications. Because resources are used by the work to maintain the cluster, each node is less powerful than if it were in a single instance. RAC is highly available but at a cost of maintainability and individual system resources. Scale the hardware solution vertically to handle the load of the requests from the application tier. Adding nodes horizontally slows down the environment.

Platform indexes

Indexes provide significant performance improvements when measured against a broad performance test workload. However, database platform improvements might vary depending on factors such as application usage, load patterns, hardware sizing, application, and database server configuration. Database administrators must monitor databases for efficient index usage to determine the impact that is produced by applying the indexes. Administrators must also identify indexes that improve performance based on situational and data composition needs.

The indexes that are listed are not included in the TRIRIGA base product unless otherwise stated.

Application platform indexes

Add the following application platform tuning indexes to Oracle databases:

```
CREATE INDEX "PERF01_BUDGET_CODES" ON "BUDGET_CODES" ("STATUS","CODE_REF_ID","TRANSACTION_ID")
TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF01_BUDGET_CURRENCIES" ON "BUDGET_CURRENCIES"
("TRANSACTION_ID","CURRENCY_CODE","AMOUNT") TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF01_BUDGET_TRANSACTION" ON "BUDGET_TRANSACTION"
("TRANSACTION_TYPE","REVERSE_FLAG","SYSTEM_DATE") TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF01_ORGANIZATION" ON "T_ORGANIZATION" (SYS_GUIID,SYS_OBJECTID) TABLESPACE
"TRIDATA_INDX";

CREATE INDEX "PERF02_ORGANIZATION" ON "T_ORGANIZATION" (UPPER(TRINAMETX)) TABLESPACE
"TRIDATA_INDX";

CREATE INDEX "PERF01_TRIBUILDING" ON "T_TRIBUILDING" (SYS_GUIID,SYS_OBJECTID) TABLESPACE
"TRIDATA_INDX";

CREATE INDEX "PERF01_TRIPEOPLE" ON "T_TRIPEOPLE" ("TRIRECORDNAMESY") TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF02_TRIPEOPLE" ON "T_TRIPEOPLE" (TRIIDTX) TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF01_TRIPROPERTY" ON "T_TRIPROPERTY" (SYS_GUIID,SYS_OBJECTID) TABLESPACE
"TRIDATA_INDX";

CREATE INDEX "PERF01_TRISPACE" ON "T_TRISPACE"
(SYS_OBJECTID,SYS_GUIID,UPPER(triNameTX),UPPER(triIdTX)) TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF01_TRISPACEALLOCATIONFACT" ON "T_TRISPACEALLOCATIONFACT"
(TRICAPTUREPERIODTXOBJID,triDimSpaceClassTXObjId,triDimLocationTXObjId,triDimWorkpointFlagLI,
TRIFACTALLOCWORKPOINTS,TRIFACTALLOCAREAIMPNU) TABLESPACE "TRIDATA_INDX";

CREATE INDEX "PERF02_TRISPACEALLOCATIONFACT" ON "T_TRISPACEALLOCATIONFACT"
(TRICAPTUREPERIODTXOBJID,TRIFACTALLOCMOVESNU,TRIFACTALLOCWORKERSNU) TABLESPACE "TRIDATA_INDX";
```

```
CREATE INDEX "PERF01_TRIWORKTASK" ON "T_TRIWORKTASK"
(SYS_GUIID,UPPER("TRINAMETX"),SYS_OBJECTID) TABLESPACE "TRIDATA_INDX";

CREATE INDEX TRIDATA.IDX2002171047150 ON TRIDATA.T_TRISPACECLASSCURRENT (TRIFLOORCOMMONBL ASC,
SPEC_ID ASC);

CREATE INDEX TRIDATA.IDX2002171049050 ON TRIDATA.T_TRISPACE (CLASSIFIEDBYSPACESYSKEY ASC,
SYS_PROJECTID ASC,SYS_OBJECTID ASC, TRISTATUSCL ASC, TRINOTCOMMONBL ASC, TRINOAREAROLLUPBL ASC,
SPEC_ID ASC, TRIUSERMESSAGEFLAGTX ASC, TRIPATHTX ASC, TRIAREANU ASC, SYS_TYPE1 ASC, SYS_GUIID
ASC, TRIAREAUO ASC);

CREATE INDEX TRIDATA.IDX2002171054180 ON TRIDATA.T_TRISPACECLASSCURRENT (TRIBUILDINGCOMMONBL
ASC, SPEC_ID ASC, TRINOTRENTABLEBL ASC);

CREATE INDEX TRIDATA.IDX2002171058490 ON TRIDATA.T_ORGANIZATION (SYS_GUIID ASC, SYS_OBJECTID
ASC, SPEC_ID ASC, SYS_TYPE1 ASC, TRIPATHTX ASC, TRISHORTNAMETX ASC, TRIORGTYPECLOBJID ASC,
TRIORGTYPECL ASC, TRIIDTX ASC, TRINAMETX ASC, TRISTATUSCL ASC);

CREATE INDEX TRIDATA.IDX2002171100140 ON TRIDATA.T_TRISPACE (SYS_OBJECTID ASC,
TRINOAREAROLLUPBL ASC);

CREATE INDEX TRIDATA.IDX2002171100080 ON TRIDATA.T_TRISPACE (SYS_OBJECTID ASC, TRISTATUSCL ASC);

CREATE INDEX TRIDATA.IDX2002171102140 ON TRIDATA.T_TRISPACECLASSCURRENT (TRINOTRENTABLEBL ASC,
SPEC_ID ASC, TRINAMETX ASC);

CREATE INDEX TRIDATA.IDX2002171104130 ON TRIDATA.T_TRISPACE (CLASSIFIEDBYSPACESYSKEY ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, TRINOAREAROLLUPBL ASC, SPEC_ID ASC,
TRINAMETX ASC, TRIOCCUPANCYSTATUSCL ASC, TRIOCCUPANCYSTATUSCLOBJID ASC, TRIAREANU ASC,
TRITOTALPRORATEDAREANU ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, TRIAREAUO ASC);

CREATE INDEX TRIDATA.IDX2002171104410 ON TRIDATA.T_TRISPACECLASSCURRENT (TRINOTRENTABLEBL ASC,
SPEC_ID ASC);

CREATE INDEX TRIDATA.IDX2002171106310 ON TRIDATA.T_TRISPACE (CLASSIFIEDBYSPACESYSKEY ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, TRINOAREAROLLUPBL ASC, SPEC_ID ASC,
TRIUSERMESSAGEFLAGTX ASC, TRIPATHTX ASC, TRIAREANU ASC, SYS_TYPE1 ASC, SYS_GUIID ASC, TRIAREAUO
ASC);

CREATE INDEX TRIDATA.IDX2002171108520 ON TRIDATA.T_TRISPACE (CLASSIFIEDBYSPACESYSKEY ASC,
SYS_PROJECTID ASC, SYS_OBJECTID ASC, TRISTATUSCL ASC, TRINOAREAROLLUPBL ASC, SPEC_ID ASC,
TRINAMETX ASC, TRIOCCUPANCYSTATUSCL ASC, TRIOCCUPANCYSTATUSCLOBJID ASC, TRIAREANU ASC,
SYS_TYPE1 ASC, SYS_GUIID ASC, TRIAREAUO ASC);

CREATE INDEX TRIDATA.IDX2002171109370 ON TRIDATA.T_TRISPACECLASSCURRENT (TRINOTUSABLEBL ASC,
SPEC_ID ASC, TRIPATHTX ASC, TRINAMETX ASC);

CREATE INDEX TRIDATA.IDX2002171112150 ON TRIDATA.T_TRIMOVEREQUEST (SYS_GUIID ASC, SYS_PROJECTID
ASC, SYS_OBJECTID ASC, SPEC_ID ASC, SYS_TYPE1 ASC, TRIREVISIONNU ASC, TRISTATUSCLOBJID
ASC, TRILOCATIONREQUESTEDOBJID ASC, TRILOCATIONREQUESTED ASC, TRIREQUESTEDFORTXOBJID ASC,
TRIREQUESTEDFORTX ASC, TRIDESCRIPTIONTX ASC, TRIREQUESTCLASSCLOBJID ASC, TRIREQUESTCLASSCL ASC,
TRICREATEDDATETIMESY ASC, TRIIDTX ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCL ASC);

CREATE UNIQUE INDEX TRIDATA.IDX2002171101580 ON TRIDATA.T_TRISPACECLASSCURRENT (SPEC_ID ASC,
TRIFLOORCOMMONBL, TRIBUILDINGCOMMONBL, TRIPROPERTYCOMMONBL, TRINAMETX);

CREATE INDEX TRIDATA.PERF1806071724280 ON TRIDATA.T_ORGANIZATION (SYS_GUIID ASC, SYS_OBJECTID
ASC, TRIFORMLABELSY ASC, TRIUSERMESSAGEFLAGTX ASC, TRISTATUSCLOBJID ASC, TRISTATUSCL ASC,
TRINAMETX ASC, TRIIDTX ASC, SYS_GEOGRAPHYNAMEOBJID ASC, SYS_ORGNAMEOBJID ASC, SYS_TYPE1 ASC,
SPEC_ID ASC);
```

Reserve indexes

Tune any Reserve indexes to include appropriate indexes for performance improvement. The following indexes increase performance for reserve queries on Oracle. However, review and tune these indexes for your implementation.

```
CREATE INDEX "PERF01_TRIRESERVATIONINSTANCE" ON "T_TRIRESERVATIONINSTANCE"
(triPlannedStartDT,SYS_OBJECTID,SYS_GUIID,SYS_PROJECTID,triStatusCL) TABLESPACE "PERF34_INDX";

CREATE INDEX "PERF01_TRIRESERVATIONRESOURCE" ON "T_TRIRESERVATIONRESOURCE"
(SPEC_ID,SYS_OBJECTID,triResourceTypeLI,SYS_PROJECTID) TABLESPACE "PERF34_INDX";

CREATE INDEX "PERF03_TRIPEOPLE" ON "T_TRIPEOPLE" (SPEC_ID,SYS_OBJECTID) TABLESPACE
"PERF34_INDX";

CREATE INDEX "PERF01_MYPROFILE" ON "T_MYPROFILE" (SPEC_ID,SYS_OBJECTID,triRecordIdSY)
TABLESPACE "PERF34_INDX";
```

```
CREATE INDEX "PERF01_TRIRESERVATIONDEF" ON "T_TRIRESERVATIONDEFINITION"
(SPEC_ID,SYS_OBJECTID,SYS_GUIID,SYS_PROJECTID) TABLESPACE "PERF34_INDX";

CREATE INDEX "PERF01_TRIROLE" ON "T_TRIROLE" (SPEC_ID,triNameTX) TABLESPACE "PERF34_INDX";

CREATE INDEX "PERF01_TRICONTACTROLE" ON "T_TRICONTACTROLE"
(SPEC_ID,SYS_OBJECTID,ClassifiedByRoleSysKey) TABLESPACE "PERF34_INDX";
```

Lease indexes

Performance benchmark testing for Lease indexes was performed on the DB2 database platform. However, the findings from that platform might also apply to Oracle Database. Your database administrator can take the identified queries from the Db2 results and use the index advisor to see which indexes are used on that platform.

Customizations:

The Db2 results are based on default queries and do not consider any additional columns that might be in your deployment. For more information, see *Lease indexes* in "Tuning the IBM Db2 indexes" on page 23.

Hierarchy indexes

If you have large hierarchies and see slow hierarchy cache (HCACHE) rebuild times of 5 minutes or more, add the following index to reduce rebuild times:

```
CREATE UNIQUE INDEX TRIDATA.UK2_IBS_SPEC_STRUCTURE ON TRIDATA.IBS_SPEC_STRUCTURE
(PARENT_SPEC_ID, CHILD_SPEC_ID,CHILD_SPEC_TEMP_ID)
LOGGING
TABLESPACE TRIDATA_DATA
PCTFREE 10
INITRANS 2
MAXTRANS 255
STORAGE (
INITIAL 10M
NEXT 10M
MINEXTENTS 1
MAXEXTENTS UNLIMITED
PCTINCREASE 0
BUFFER_POOL DEFAULT
);
```

Microsoft SQL Server database

Tune Microsoft SQL Server components such as the server and indexes.

Tuning Microsoft SQL Server

Tune Microsoft SQL Server for optimal performance.

Tuning the server and memory

Use a dedicated server for the TRIRIGA database when you use Microsoft SQL Server. Compared to other database platforms, Microsoft SQL Server requires up to twice the memory resources to achieve the same level of performance as other database platforms, so a large memory allocation is needed when you are implementing Microsoft SQL Server.

Isolating snapshots

Configure the database to allow `read committed isolation` to reduce blocking:

```
ALTER DATABASE <dbname>
SET ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE <dbname>
SET READ_COMMITTED_SNAPSHOT ON
```

For more information, see:

- Snapshot Isolation in SQL Server

- Row Versioning-based Isolation Levels in the Database Engine
- Using Row Versioning-based Isolation Levels

Implicit conversions

When SQL Server tries to join on or compare fields of different data types, it converts one to match the other. This is called *implicit conversion*. An implicit conversion is not desired and can lead to poor performance because SQL Server does not use indexes optimally. For more information, see decimal and numeric (Transact-SQL).

If there are implicit conversions, the plans that are generated might still be cached in SQL Server. For more information, see the following script that shows the plans with the implicit conversions Show Plans with Implicit Conversions.

When you run the script, the results are shown. If you click the XML link in the results, you can search for and observe the implicit conversion.

Sparse columns

Microsoft SQL Server has a limitation for row size of approximately 8060 bytes of data. This limitation applies to data that is stored in the row. Most *VARCHAR/NVARCHAR* data is kept off row in a stored pointer.

Some users have issues when storing data because the row data is too large for specific business object tables, for example, T tables such as `T_TRIREALESTATECONTRACT`, `T_TRICAPITALPROJECT`, and `T_TRIBUILDING`. Users see the `Cannot Create a row of size NNNN which is greater than the allowable maximum row size of 8060` in the `server.log`, and they cannot save their record. To resolve such a situation, analyze the business object and delete unused fields.

An alternative is to use is *sparse columns*, which is a column that is optimized for null value storage. Null value storage is optimized at the expense of value storage. A sparse column with a null value takes up no storage space. However, if the column has a value, 2-4 more bytes over the value size are needed to save the field value. There is a tradeoff, and the ratio of nonnull to null values needs to be significant for any benefit. Do not use sparse columns unless the space saved is at least 20-40%. There is also a cost when reading nonnull values from sparse columns. Table operations including this column might require more processing. However, depending on the data that is being stored, you might use this option to reduce row sizes.

Database Table Manager:

For environments that use a Microsoft SQL Server database, IBM TRIRIGA Application Platform supports the actions of viewing, creating, and removing sparse columns by using the Database Table Manager tool. For more information, see Database tables.

Alternatively, to make a sparse column, you must work outside of TRIRIGA directly in the database. Do not publish the business object because this action might make the field revert to nonsparse. Use sparse columns sparingly because they might have a performance impact depending on the data. For more information, see Use Sparse Columns.

Cleanup parameters for rebuilding the index

The `INDEX REBUILD` script contains the nightly Platform Maintenance stored procedure for Microsoft SQL Server. However, you must configure the following parameters for each of your environments. Consult with your SQL Server database administrator to tune these settings for optimal impact in your environment:

`reorg_frag_thresh`
The percentage threshold at which to reorganize indexes.

`rebuild_frag_thresh`
The percentage threshold at which to switch from reorganizing to rebuilding.

fill_factor
The percentage of space on each leaf-level page to fill with data.

page_count_thresh
If set, the number of pages that the current table uses.

rebuild_online
Rebuilds the indexes online so that other processes can access the table while the rebuild is occurring.

compute_statistics
If set, computes the statistics on the tables.

report_only
If set to true, this parameter does not build the index. Instead, it reports the status of the indexes, stats, and tables.

MinutesToRun
The number of minutes to run, where **0** (zero) means run until complete. **MinutesToRun** can be set to a limit. For example, set it to 60 minutes, so that the process stops after 1 hour. On the next run, the process picks up where it last stopped.

Tuning the Microsoft SQL Server indexes

Tune the application platform, reserve, and lease indexes to improve performance.

Overview

Indexes provide significant performance improvements when measured against a broad performance test workload. However, database platform performance gains might vary depending on factors such as application usage, load patterns, hardware sizing, application, and database server configuration. Database administrators must monitor databases for efficient index usage to identify the impact of applying the recommended indexes. They must also determine other indexes that improve performance based on situational and data composition needs.

The indexes that are listed are not included in the TRIRIGA base product unless otherwise stated.

Customizations:

These indexes are based on default SQL queries, which you might need to alter to account for custom columns or other customizations that alter the query to which the index pertains.

In addition, Microsoft SQL Server imposes different restrictions on the size of indexes depending on the version that you are using. If you try to apply these indexes and receive a warning about the length of the index, you might need to remove columns from the end of the recommended index to achieve an index size that works for your version of Microsoft SQL Server. Multibyte character sets (MBCS) are especially vulnerable to these restrictions.

Application platform indexes

Add the following application platform tuning indexes to Microsoft SQL Server databases:

```
CREATE INDEX [PERF_APP_OBJECT_PERMISSION1] ON [APP_OBJECT_PERMISSION] ([APPLICATION_ID],
[TEMPLATE_ID], [TAB_ID], [SECTION_ID], [FIELD_ID], [SERVICE_ID], [GROUP_ID])
GO

CREATE INDEX [PERF_APP_OBJECT_PERMISSION2] ON [APP_OBJECT_PERMISSION] ([TAB_ID], [SECTION_ID],
[FIELD_ID], [TEMPLATE_ID], [SERVICE_ID], [GROUP_ID])
GO

CREATE INDEX [PERF_BUDGET_CODES1] ON [BUDGET_CODES] ([STATUS], [CODE_REF_ID], [TRANSACTION_ID])
GO

CREATE INDEX [PERF_BUDGET_CURRENCIES1] ON [BUDGET_CURRENCIES] ([TRANSACTION_ID],
[CURRENCY_CODE], [AMOUNT])
GO

CREATE INDEX [PERF_BUDGET_TRANSACTION1] ON [BUDGET_TRANSACTION] ([TRANSACTION_TYPE],
```

```
[REVERSE_FLAG], [SYSTEM_DATE])
GO

CREATE INDEX [PERF_BUDGET_TRANSACTION2] ON [BUDGET_TRANSACTION] ([OBJECT_ID],
[REVERSE_FLAG]) INCLUDE ([TRANSACTION_ID], [TRANSACTION_TYPE], [DESCRIPTION],
[TRANSACTION_DATE], [SYSTEM_DATE], [COMPANY_ID], [PROGRAM_ID], [PROJECT_ID], [BO_ID],
[OBJECT_VERSION], [MODULE_ID], [ORGANIZATION_ID], [GEOGRAPHY_ID], [USER_ID], [REF_OBJECT_ID],
[REF_OBJECT_VERSION], [REF_MODULE_ID], [REF_BO_ID], [REVERSE_DATE], [LOCATION_ID])
GO

CREATE INDEX [PERF_GROUPMEMBER1] ON [T_GROUPMEMBER] ([PAR_SPEC_ID], [MEMBERTYPE],
[SYS_OBJECTID]) INCLUDE ([MEMBERID])
GO

CREATE INDEX [PERF_GUI_HEADER_PUBL1] ON [GUI_HEADER_PUBL] ([GUI_NAME])
GO

CREATE INDEX [PERF_GUI_HEADER_PUBL2] ON [GUI_HEADER_PUBL] ([SPEC_CLASS_TYPE], [ALT_PRINT_FORM])
GO

CREATE INDEX [PERF_IBS_SPEC1] ON [IBS_SPEC] ([ROOT_FLG]) INCLUDE ([SPEC_CLASS_TYPE], [SPEC_ID])
GO

CREATE INDEX [PERF_IBS_SPEC_TYPE1] ON [IBS_SPEC_TYPE] ([COMPANY_ID], [SPEC_CLASS_TYPE],
[DELETED_FLAG])
GO

CREATE INDEX [PERF_IBS_SPEC_TYPE2] ON [IBS_SPEC_TYPE] ([DELETED_FLAG], [EXT_MANAGED],
[PASS_THROUGH_FLAG], [SHOW_IN_MANAGER], [SPEC_CLASS_TYPE])
GO

CREATE INDEX [PERF_IBS_SPEC_TYPE3] ON [IBS_SPEC_TYPE] ([NAME], [DELETED_FLAG])
GO

CREATE INDEX [PERF_IBS_TEMP_SPEC_ASSIGNMENTS1] ON [IBS_TEMP_SPEC_ASSIGNMENTS] ([RAND_NO],
[SPEC_ID])
GO

CREATE INDEX [PERF_IBS_TEMP_SPEC_ASSIGNMENTS2] ON [IBS_TEMP_SPEC_ASSIGNMENTS] ([RAND_NO],
[SPEC_ID], [ASS_TYPE]) INCLUDE ([ASS_SPEC_ID], [ACTION])
GO

CREATE INDEX [PERF_LIST_VALUE1] ON [LIST_VALUE] ([LIST_ID], [COMPANY_ID], [LANGUAGE_ID])
GO

CREATE INDEX [PERF_ORGANIZATION1] ON [T_ORGANIZATION] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([TRISTATUSCL])
GO

CREATE INDEX [PERF_ORGANIZATION2] ON [T_ORGANIZATION] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([SPEC_ID], [SYS_TYPE1], [TRIIDTX], [TRINAMETX], [TRISTATUSCL], [TRIPATHTX], [TRISHORTNAMETX],
[TRIORGTYPECL], [TRIORGTYPECLOBJID])
GO

CREATE INDEX [PERF_ORGANIZATION3] ON [T_ORGANIZATION] ([SYS_PROJECTID], [SYS_OBJECTID])
INCLUDE ([SPEC_ID], [SYS_GUIID], [SYS_TYPE1], [SYS_ORGNAME], [SYS_ORGNAMEOBJID], [TRIIDTX],
[TRINAMETX], [TRISTATUSCL], [TRIFORMLABELSY])
GO

CREATE INDEX [PERF_RESOURCE_AVAILABILITY1] ON [RESOURCE_AVAILABILITY] ([SPEC_ID], [TASK_ID])
GO

CREATE INDEX [PERF_SCHEDULEDEVENTS1] ON [T_SCHEDULEDEVENTS] ([EVENTSTATUS], [SYS_OBJECTID],
[ENDDATETIME]) INCLUDE ([SPEC_ID], [SYS_GUIID], [SYS_TYPE1], [STARTDATETIME])
GO

CREATE INDEX [PERF_TRIBUILDING0] ON [T_TRIBUILDING] ([SYS_GUIID], [SYS_OBJECTID])
GO

CREATE INDEX [PERF_TRIBUILDING1] ON [T_TRIBUILDING] ([SYS_PROJECTID], [SYS_OBJECTID])
INCLUDE ([TRIGROSSMAREAMETNU], [TRIGROSSMAREAMETNU_UOM], [TRIGROSSMAREAIMPNU],
[TRIGROSSMAREAIMPNU_UOM], [TRIAREAUO], [TRIBUILDINGCOMMONAREAN], [SPEC_ID],
[SYS_GUIID], [SYS_TYPE1], [TRINAMETX], [TRIUSERMESSAGEFLAGTX], [TRIFORMLABELSY],
[TRIPATHTX], [TRIPARENTPROPERTYTX], [TRIPARENTPROPERTYTXOBJID], [TRIBUILDINGCLASSCL],
[TRIBUILDINGCLASSCLOBJID], [TRINUMBEROFFLOORSNU], [TRIGROSSAREAMETNU], [TRIGROSSAREAMETNU_UOM],
[TRIGROSSAREAIMPNU], [TRIGROSSAREAIMPNU_UOM], [TRILENGTHUO])
GO

CREATE INDEX [PERF_TRIBUILDING2] ON [T_TRIBUILDING] ([SYS_PROJECTID], [SYS_OBJECTID],
[SYS_GUIID]) INCLUDE ([TRINAMETX], [TRISTATUSCL])
GO
```

```
CREATE INDEX [PERF_TRIBUILDINGFACT1] ON [T_TRIBUILDINGFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIFACTCAPITALFIXEDASS], [TRIDIMBUILDINGTENURETXOBJID], [TRIDIMBUILDINGCLASSTXOBJID],
[TRIDIMLOCATIONTXOBJID], [TRIFACTREPLACEMENTVALU])
GO

CREATE INDEX [PERF_TRIBUILDINGFACT2] ON [T_TRIBUILDINGFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIFACTMAINTENANCECOST], [TRIDIMBUILDINGTENURETXOBJID], [TRIDIMBUILDINGCLASSTXOBJID],
[TRIDIMLOCATIONTXOBJID], [TRIFACTREPLACEMENTVALU])
GO

CREATE INDEX [PERF_TRIBUILDINGSYSTEMITEMFACT1] ON [T_TRIBUILDINGSYSTEMITEMFACT]
([TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMBUILDINGTENURETXOBJID],
[TRIDIMBUILDINGCLASSTXOBJID], [TRIFACTREPLACEMENTVALU], [TRIDIMLOCATIONTXOBJID],
[TRIFACTESTIMATEDREPAIR], [TRIDIMBUILDINGSYSTEMCLOBJID])
GO

CREATE INDEX [PERF_TRICAPITALPROJECT1] ON [T_TRICAPITALPROJECT] ([SYS_PROJECTID], [SYS_GUIID],
[SYS_OBJECTID]) INCLUDE ([TRIDATEDA], [SPEC_ID], [SYS_TYPE1], [TRIIDTX], [TRINAMETX],
[TRISTATUSCL], [TRISTATUSCLOBJID])
GO

CREATE INDEX [PERF_TRICAPITALPROJECT2] ON [T_TRICAPITALPROJECT] ([SYS_PROJECTID], [SYS_GUIID],
[SYS_OBJECTID]) INCLUDE ([TRINAMETX], [TRISTATUSCL])
GO

CREATE INDEX [PERF_TRICAPITALPROJECTFACT1] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTBUDGETCURRENTAM], [TRIFACTCOMMITMENTCHANG], [TRIDIMPROGRAMTX])
GO

CREATE INDEX [PERF_TRICAPITALPROJECTFACT2] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTCURRENTBUDGETTO], [TRIFACTBUDGETCURRENTAM], [TRIDIMPROGRAMTX])
GO

CREATE INDEX [PERF_TRICAPITALPROJECTFACT3] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTORIGINALBUDGETT], [TRIFACTBUDGETORIGINALA], [TRIDIMPROGRAMTX])
GO

CREATE INDEX [PERF_TRICAPITALPROJECTFACT4] ON [T_TRICAPITALPROJECTFACT] ([TRIDIMUSERIDTXOBJID],
[TRIDIMSTATUSTX], [TRICAPTUREPERIODTXOBJID], [TRIDIMPROGRAMTXOBJID]) INCLUDE
([TRIFACTSCHEDULEVARIANC], [TRIFACTCOUNTTOTALNUMBE2], [TRIDIMPROGRAMTX])
GO

CREATE INDEX [PERF_TRICLAUSETYPE1] ON [T_TRICLAUSETYPE] ([TRINAMETX])
GO

CREATE INDEX [PERF_TRIPEOPLE1] ON [T_TRIPEOPLE] ([TRIIDTX])
GO

CREATE INDEX [PERF_TRIPEOPLE2] ON [T_TRIPEOPLE] ([TRIRECORDNAMESY])
GO

CREATE INDEX [PERF_TRIPEOPLE3] ON [T_TRIPEOPLE] ([SYS_GUIID], [SYS_OBJECTID])
INCLUDE ([TRIWORKFAXTX], [TRIWORKPHONETX], [TRIEMAILTX], [TRISTATUSCL], [TRITITLETX],
[PRIMARYORGANIZATIONSYSKEY], [SPEC_ID], [SYS_TYPE1], [TRIUSERMESSAGEFLAGTX], [TRINAMETX])
GO

CREATE INDEX [PERF_TRIPEOPLE4] ON [T_TRIPEOPLE] ([SYS_GUIID], [SYS_OBJECTID], [TRIIDTX])
GO

CREATE INDEX [PERF_TRIPEOPLE5] ON [T_TRIPEOPLE] ([SYS_OBJECTID], [SYS_GUIID]) INCLUDE
([TRIFIRSTNAMETX], [TRILASTNAMETX], [TRISTATUSCL], [PRIMARYORGANIZATIONSYSKEY], [SPEC_ID],
[SYS_TYPE1], [TRIUSERMESSAGEFLAGTX], [TRINAMETX], [TRIIDTX], [TRIFORMLABELSY])
GO

CREATE INDEX [PERF_TRIPEOPLE6] ON [T_TRIPEOPLE] ([TRINAMETX]) INCLUDE ([SPEC_ID])
GO

CREATE INDEX [PERF_TRIPROJECTBUDGETCHANGE1] ON [T_TRIPROJECTBUDGETCHANGE] ([SYS_PROJECTID],
[SYS_GUIID], [SYS_OBJECTID]) INCLUDE ([SPEC_ID], [SYS_TYPE1], [TRIDATEDA], [TRINAMETX],
[TRIIDTX], [TRISTATUSCL], [TRISTATUSCLOBJID], [TRIUSERMESSAGEFLAGTX], [TRIREVISIONNU])
GO

CREATE INDEX [PERF_TRIPROJECTBUDGETCHANGE2] ON [T_TRIPROJECTBUDGETCHANGE] ([SYS_PROJECTID],
[SYS_GUIID], [SYS_OBJECTID]) INCLUDE ([TRISTATUSCL])
GO

CREATE INDEX [PERF_TRIPROJECTORIGINALBUDGET1] ON [T_TRIPROJECTORIGINALBUDGET] ([SYS_PROJECTID],
[SYS_GUIID], [SYS_OBJECTID])
GO
```

```
CREATE INDEX [PERF_TRIRECONTRACTFACT1] ON [T_TRIRECONTRACTFACT] ([TRIFACTACCOUNTINGTYPET])
INCLUDE ([TRIDIMORGANIZATIONTXOBJID], [TRIDIMPRIMARYUSETXOBJID], [TRIFACTTOTALCONTRACTRE],
[TRIFACTTOTALCOSTNU])
GO

CREATE INDEX [PERF_TRIREPAYMENTFACT1] ON [T_TRIREPAYMENTFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMCONTRACTTYPETXOBJID], [TRIDIMPAYMENTTYPETXOBJID], [TRIFACTOUTSTANDINGRECE],
[TRIFACTOUTSTANDINGDAYS])
GO

CREATE INDEX [PERF_TRIREPAYMENTFACT2] ON [T_TRIREPAYMENTFACT] ([TRIDIMCONTRACTADMINISTOBJID],
[TRIDIMISPAIDTXOBJID], [TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMCONTRACTTYPETXOBJID],
[TRIDIMPAYMENTTYPETXOBJID], [TRIFACTTOTALPAYMENTSNU], [TRISCOREONTIMENU])
GO

CREATE INDEX [PERF_TRISPACE1] ON [T_TRISPACE] ([SYS_OBJECTID], [SYS_GUIID], [TRINAMETX],
[TRIIDTX])
GO

CREATE INDEX [PERF_TRISPACEALLOCATIONFACT1] ON [T_TRISPACEALLOCATIONFACT]
([TRICAPTUREPERIODTXOBJID], [TRIDIMSPACECLASSTXOBJID], [TRIDIMLOCATIONTXOBJID],
[TRIDIMWORKPOINTFLAGLI], [TRIFACTALLOCWORKPOINTS], [TRIFACTALLOCAREAIMPNU])
GO

CREATE INDEX [PERF_TRISPACEALLOCATIONFACT2] ON [T_TRISPACEALLOCATIONFACT]
([TRICAPTUREPERIODTXOBJID], [TRIFACTALLOCMOVESNU], [TRIFACTALLOCWORKERSNU])
GO

CREATE INDEX [PERF_TRISPACEFACT1] ON [T_TRISPACEFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMGEOGRAPHYTXOBJID], [TRIDIMSPACECLASSTXOBJID], [TRIFACTSPACEAREAIMPNU],
[TRIFACTSPACEALLOCATEDA])
GO

CREATE INDEX [PERF_TRISPACEFACT2] ON [T_TRISPACEFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMSPACECLASSTXOBJID], [TRIDIMLOCATIONTXOBJID], [TRIFACTSPACEAREAIMPNU],
[TRIFACTSPACEALLOCATEDA])
GO

CREATE INDEX [PERF_TRISPACEPEOPLEFACT1] ON [T_TRISPACEPEOPLEFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMSPACECLASSTXOBJID], [TRIFACTALLOCWORKERSNU], [TRIFACTALLOCAREAIMPNU],
[TRIDIMWORKERTYPETXOBJID], [TRIDIMLOCATIONTXOBJID])
GO

CREATE INDEX [PERF_TRISURVEYFACT1] ON [T_TRISURVEYFACT] ([TRIDIMSURVEYTYPETX])
INCLUDE ([TRIDIMREQUESTCLASSTXOBJID], [TRIFACTRESPONSESCORENU], [TRIFACTMAXIMUMSCORENU],
[TRICAPTUREPERIODTXOBJID])
GO

CREATE INDEX [PERF_TRISURVEYFACT2] ON [T_TRISURVEYFACT] ([TRIDIMSURVEYTYPETX],
[TRICAPTUREPERIODTXOBJID]) INCLUDE ([TRIDIMLOCATIONTXOBJID], [TRIDIMREQUESTCLASSTXOBJID],
[TRIFACTRESPONSESCORENU], [TRIFACTMAXIMUMSCORENU])
GO

CREATE INDEX [PERF_TRITASKDETAILFACT1] ON [T_TRITASKDETAILFACT] ([TRICAPTUREPERIODTXOBJID])
INCLUDE ([TRIDIMTASKTYPETXOBJID], [TRIFACTPREVENTIVETASKC], [TRIFACTPREVENTIVETASKS],
[TRIDIMLOCATIONTXOBJID])
GO

CREATE INDEX [PERF_TRIWORKTASK1] ON [T_TRIWORKTASK] ([SYS_PROJECTID],
[SYS_GUIID], [SYS_OBJECTID]) INCLUDE ([TRIMATRIXSERVICECLAS], [TRIMATRIXSERVICECLASOBJID],
[TRIWORKINGLOCATIONTX], [TRIWORKINGLOCATIONTXOBJID], [SPEC_ID], [SYS_TYPE1], [TRINAMETX],
[TRIUSERMESSAGEFLAGTX], [TRIIDTX], [TRISTATUSCL], [TRISTATUSCLOBJID], [TRIACTUALPERCENTCOMP],
[TRIACTUALPERCENTCOMP_UOM], [TRIACTUALENDDT], [TRIACTUALSTARTDT], [TRIPLANNEDSTARTDT],
[TRIPLANNEDENDDT])
GO

CREATE INDEX [PERF_WEB_LABEL1] ON [WEB_LABEL] ([APPLICATION_ID], [BO_ID]) INCLUDE
([LANGUAGE_ID], [LABEL_NAME], [LABEL_VALUE], [UPDATED_BY], [UPDATED_DATE])
GO

CREATE INDEX [PERF_WEB_MESSAGE1] ON [WEB_MESSAGE] ([LANGUAGE_ID], [USE_NAME])
GO

CREATE INDEX [PERF_WF_EVENT_HISTORY1] ON [WF_EVENT_HISTORY] ([COMPLETED_DATE])
GO

CREATE INDEX [PERF_WF_TEMPLATE1] ON [WF_TEMPLATE] ([STATUS_ID], [TEMPLATE_FLAG],
[UPDATED_DATE]) INCLUDE ([WF_TEMPLATE_ID], [WF_TEMPLATE_VERSION])
GO
```

Reserve indexes

Tune Reserve indexes to include appropriate indexes for performance improvement. The following indexes were identified to help increase performance dramatically for reserve queries by the TRIRIGA performance team on SQL Server, but you must review and tune for your specific implementation.

```
CREATE INDEX [PERF01_TRIRESERVATIONINSTANCE] ON [T_TRIRESERVATIONINSTANCE]
([triPlannedStartDT], [SYS_OBJECTID], [SYS_GUIID], [SYS_PROJECTID])
GO

CREATE INDEX [PERF01_TRIRESERVATIONRESOURCE] ON [T_TRIRESERVATIONRESOURCE] ([SPEC_ID],
[SYS_OBJECTID])
GO

CREATE INDEX [PERF03_TRIPEOPLE] ON [T_TRIPEOPLE] ([SPEC_ID], [SYS_OBJECTID])
GO

CREATE INDEX [PERF01_MYPROFILE] ON [T_MYPROFILE] ([SPEC_ID], [SYS_OBJECTID])
GO

CREATE INDEX [PERF01_TRIRESERVATIONDEF] ON [T_TRIRESERVATIONDEFINITION] ([SPEC_ID],
[SYS_OBJECTID], [SYS_GUIID], [SYS_PROJECTID])
GO

CREATE INDEX [PERF01_TRIROLE] ON [T_TRIROLE] ([SPEC_ID], [triNameTX])
GO

CREATE INDEX [PERF01_TRICONTACTROLE] ON [T_TRICONTACTROLE] ([SPEC_ID], [SYS_OBJECTID],
[ClassifiedByRoleSysKey])
GO
```

Lease indexes

Performance benchmark testing for Lease indexes was performed on the DB2 database platform. However, the findings from that platform might also apply to Microsoft SQL Server. Your database administrator can take the identified queries from the Db2 results and use the index advisor for your database platform to see which indexes are recommended on that platform.

Customizations:

The Db2 results are based on default queries and do not consider any additional columns that might be in your deployment. For more information, see *Lease indexes* in “Tuning the IBM Db2 indexes” on page 23.

Chapter 6. Tuning the application server

Tune your application server by configuring Java virtual machine (JVM) heap size, WebSphere Application Server Liberty profile, and HTTP compression settings.

Overview

Application server settings provide optimal performance in IBM test environments. You can use these settings as a starting point, and then customize the settings to your environment requirements. For the latest supported application server versions and fix packs, see IBM TRIRIGA Compatibility Matrix.

General tuning

JVM heap size

To begin tuning, use the following minimum heap size values:

- Initial Heap Size: 6144M
- Maximum Heap Size: 6144M

Overall system memory and heap sizes:

When you are planning for overall system memory consumption, include more memory to be used by the JVM outside of the heap size. You also need to plan for available RAM for the operating system. Include an extra 30-40% of memory to account for this overhead. Insufficient available RAM can cause severe performance issues.

If you are not using the Lease Accounting or Business Intelligence and Reporting Tool (BIRT) reporting features of TRIRIGA, you might start tuning your heap size with a value less than 6144M. However, do not use values less than 4096M, unless you have a development or test deployment of TRIRIGA.

In a production environment, set the initial heap size to be the same value as the maximum heap size. This setting avoids the overhead of Java having to increase the heap size during processing. However, in modern JVMs, only a small amount of time is spent allocating new heap space. So, in a nonproduction environment, or one where speed is not a concern yet memory might be constrained, keep the initial heap size smaller.

JVM heap size parameters directly influence garbage collection (GC) behavior. Increasing the heap size allows more objects to be created before garbage collection is triggered. However, a larger heap size means that it takes longer to find and process objects that need to be garbage collected. Therefore, tuning the JVM heap size often involves a balance between the interval between garbage collections and the pause time that is needed to perform the garbage collection.

To tune the JVM heap size, enable verbose GC. When you enable verbose GC, the JVM prints useful information at each garbage collection. Such information includes the amount of free and used bytes in the heap, the interval between garbage collections, and the pause time. Use this information to analyze the heap usage.

When you have finished tuning, disable verbose GC and delete the log file because the log file can grow large while verbose GC is enabled. To enable verbose GC for WebSphere Application Server, see Troubleshooting and Monitoring.

In addition, the TRIRIGA Application Platform can log heap memory usage in `systemmetrics.log`. You can analyze the tab-delimited `systemmetrics.log` to ensure optimal heap memory use. To set up Performance Monitor logging (Key Metrics section) in the Administrator Console to obtain this data, see Administrator Console User Guide.

Use the guidelines to analyze heap memory usage and garbage collection data. Then determine the appropriate value for your deployment that minimizes garbage collection times while providing enough memory to the application. Your deployment might require different settings to produce optimal performance or avoid out-of-memory issues in the JVM.

Lease accounting and BIRT reporting

Lease accounting and BIRT reporting can require large amounts of heap space, depending on the data that is being processed. Lease accounting requires large heap space for any process server that runs the asynchronous workflows.

For BIRT, heap requirements depend on the type of report that is being run and the amount of data that a report returns. Some BIRT reports might require heap sizes larger than `6144M`. Sometimes, you might need heap sizes as large as `12288M` to process Lease Disclosure reports with a large amount of data.

Even if you offload BIRT reporting to a separate BIRT process server, BIRT reports can run on the application server. For example, when reports are rerun from the **BIRT report** icon bar. In those cases, heap memory on the application server might also require heap sizes as large as those that are allocated on the BIRT process server.

Connection pool

Configure the JDBC data source connection pool properties to allow for enough connections needed by your system. These values depend on the number of concurrent users, in addition to the number of transactions expected. The following options and values can be configured for connection pools:

- Minimum Connections: Specifies the minimum number of connections to maintain in the pool.
  - Typical Default `1`
  - Recommended `10`
- Maximum Connections: Specifies the maximum number of connections to maintain in the pool.
  - Typical Default: `10`
  - Recommended: `200`

A setting of `200` maximum connections is common for a JVM that is not expected to exceed 2500 concurrent users. For example, if you have two JVMs, both with `200` maximum connections, then in theory, you can support up to 5000 concurrent users.

Temporary files

Over time, on the application or process servers, the `temp` (temporary) directory might fill with temporary files. Clean up these files regularly. Generally, it is safe to delete temporary files that are at least two days old.

You can set up a `cronjob` for Linux , or a scheduled task for Windows. The **`cronjob`** deletes `temp` files that are more than 2 days old, and that are owned by the user who is running the TRIRIGA application.

For example, to set a `cronjob` in a Linux environment, thake the following steps:

1. Add the following **`find`** command in a shell script file `/path/to/script/cleanTririgaTemp.sh`:

   ```
   find -mtime +2 -user tririga -exec rm {} \;
   ```

2. Use **`chmod`** to make it executable:

   ```
   chmod u+x /path/to/script/cleanTririgaTemp.sh
   ```

3. Run **`crontab -e:`**

   ```
   $ crontab -e
   ```

4. Set the script to run daily at 1 AM:

   ```
   0 1 * * * /path/to/script/cleanTririgaTemp.sh
   ```

IBM WebSphere Application Server Liberty profile

IBM WebSphere Application Server Liberty profile is the supported application server for TRIRIGA Application Platform.

- Liberty is easy to manage and has only the following configuration files: `server.xml`, `server.env`, and `jvm.options`.
- Liberty is secure and efficient. It installs, enables, and configures only what is needed. The Liberty paradigm enables the application server to be nimble, dynamic, and fast.
- Liberty installation and configuration takes seconds with the TRIRIGA installer. With Liberty bundled with the TRIRIGA installer, without separate downloads, installations, or configuration steps.

For more information on how to tune Liberty profile, see Tuning WebSphere Liberty Profile

HTTP compression

HTTP compression is a capability that is built into web servers, such as IBM HTTP Server (IHS) and Apache HTTP Server (AHS) 2.x and later, and web browsers. HTTP compression makes better use of available bandwidth, provides faster transmission speeds. For more information, see *Using compression techniques to improve performance* in Chapter 2, “Tuning the network,” on page 5

Load balancing

Load balancing is the distribution of the task load across multiple instances of an application. User load comes from users who are logged in to the system and use the interface to perform tasks. Non-user load comes from items such as scheduled jobs, agents, and integrations. User and non-user load can be distributed across different application servers. The external application load balancer can be utilized with IBM WebSphere Application Server Liberty to function as an agent from the web server to the application server using the Application Load Balancer protocol.

The default load balance option, **Round Robin**, provides an even distribution of work across cluster members. However, the **Random** option gives a more even distribution of work across the cluster. Test the options to determine which option works better for your deployment of TRIRIGA. You can analyze the security logs for the TRIRIGA Application Platform to identify the number of users per JVM and then determine which option provides the best load balancing.

Chapter 7. Tuning IBM TRIRIGA components

Tune IBM TRIRIGA components such as system properties and the integration framework.

System properties

Use the system properties in the TRIRIGAWEB.properties file to tune performance in TRIRIGA.

TRIRIGAWEB.properties

Most system properties are maintained in the TRIRIGAWEB.properties file and are available dynamically in the Administrator Console. For changes in the TRIRIGAWEB.properties file to take effect, restart the application server.

The variables and settings in TRIRIGA properties files might change between versions. After an upgrade installation, review each newly installed properties file and adjust values for your implementation.

Configure the following key properties and values for your environment:

CLEAN_HOUR
Specifies the hour, from 0 to 23, at which the Platform Maintenance Scheduler Cleanup Agent starts on the server.

- Recommended value: An hour when the system has the least number of users.
- Default value: 0

CLEAN_TIMEOUT
Specifies the number of minutes that the Platform Maintenance Scheduler can run.

- Default value: 120

WF_INSTANCE_SAVE
Configures when workflow instances are saved. Use **WF_INSTANCE_SAVE** to debug in a nonproduction environment. In a production environment, always set to ERRORS_ONLY because of the significant load on the entire system if turned on. ERRORS_ONLY is the default and saves the tracing information only for those workflows that fail with an ERROR condition. The value of NEVER was renamed to ERRORS_ONLY, but the system accepts both values, where NEVER is equal to ERRORS_ONLY. Change this property without a system restart by using the Workflow Agent Manager in the Administrator Console. For more information, see *Workflow Agent* and *Workflow Performance*.

- Recommended value: ERRORS_ONLY
- Default value: ERRORS_ONLY
- Value for mass or batch loading records: DATA_LOAD This value bypasses and never saves any instances, which increases performance and decreases database size. Use this value in production environments, especially if the production environment has many workflows that end in error or have Stop tasks, but no application developers are available to correct the problems in the workflows.

WF_INSTANCE_SAVE_24HR_THRESHOLD
Configures when workflow instances stop being saved. When excessive workflow instances are saved, the platform stops saving if the number of saved instances exceeds a defined threshold. Administrators can set an upper limit for how many workflow instances are saved in a 24-hour period. This setting prevents excessive workflow instance data from affecting the Platform Maintenance Scheduler. The default value is 1000. To override the default value, set **WF_INSTANCE_SAVE_24HR_THRESHOLD=*####*** where *####* is a positive integer, and restart the server. Set **WF_INSTANCE_SAVE_24HR_THRESHOLD** on each server in the environment. Do not set this value to a number larger than 10000 or the Platform Maintenance Scheduler takes a long time to remove the debugging records.

- Default value: 1000
- Maximum Value: 10000

USE_WF_BINARY_LOAD
If set to Y, the system uses the binary workflow load process, which provides a faster load time. If the workflow templates cannot be found in the workflow template cache, they are loaded with their stored binary version.

- Recommended value: Y

WF_HISTORY_RETENTION_DAYS
The Platform Maintenance Scheduler deletes workflow instances that are not waiting on user or approval tasks that are older than this number of days. This value does not have a large impact if the **WF_INSTANCE_SAVE** property is set to ERRORS_ONLY. But on a system that is saving many workflow instances, this value can significantly impact system performance.

- Recommended value: 5
- Default value: 10

DC_HISTORY_RETENTION_DAYS
The Platform Maintenance Scheduler deletes completed or obsolete DataConnect jobs that are older than this number of days. If you do not use DataConnect, then changing this value does not affect your system.

- Recommended value: 5
- Default value: 10

REPORT_MEMORY_USAGE_LIMIT
Specifies the maximum percentage of available server memory that can be used while running a user report. If a user sees a `There are not enough resources available to run the report` error for a query, the query might be the cause of the error. However, other concurrent processes might consume memory while the query was assembling its results. Valid values are between 0 and 100. Values of 0 and 100 disable enforced limits and allow a single query that a user initiates to run the server out of memory. Tune this value so that no report uses more than 1-2 GB of heap size. However, temporarily increase the value to handle especially large reports.

- Default value: 90
-

BIRT_MEMORY_USAGE_LIMIT
Specifies the maximum percentage of available server memory that can be used to assemble the BIRT report query results. If the memory requirement for a query exceeds **BIRT_MEMORY_USAGE_LIMIT**, the query gives an error. However, other concurrent processes might use memory while the query was assembling its results. Valid values are between 0 and 100. Values of 0 and 100 disable any enforced limits and allow a single query that a single user initiates to run the server out of memory.

- Recommended value: 35

TREE_PAGING_SIZE
Specifies the maximum number of child records to show in the hierarchy tree for Location, Organization, Geography, Classification, Cost Code, and newly-created hierarchical managers. The system includes the child records of the root node in the count. Increasing this value can affect performance.

- Recommended value: 1000

SESSION_HISTORY_TRACKING
Indicates which user sessions are logged to the SESSION_HISTORY table. If set to WEB_USER, user sessions from IBM TRIRIGA Connector for Business Applications (CBA) are not logged to the SESSION_HISTORY table. If your system has many integration transactions, this table can grow quickly and use system resources.

- Recommended value: `WEB_USER`
- Default value: `ALL`

`BIRT_PROCESS_SERVER`

The following properties enable BIRT report offloading. By sending the report processing to another JVM, BIRT report offloading can eliminate system resource usage by BIRT reports on a server with user sessions. For more information, see *Advanced Reporting (BIRT)*.

- BIRT_PROCESS_SERVER_HOST_NAME
- BIRT_PROCESS_SERVER_PORT
- BIRT_PROCESS_SERVER_LISTENING_PORT

STATIC_CLIENT_RESOURCE_CACHE_CONTROL_VALUE

To improve performance, you can allow static resources that do not contain sensitive customer data to be cached rather than downloaded each time the resource is requested. For more information, see TRIRIGAWEB.properties.

For more information about system properties, see the Installation and Implementation Guide.

Agents

The IBM TRIRIGA Application Platform uses business process agents to perform automated work for its applications. When the system identifies an event that requires a business process agent to fulfill, it places the event into a queue where the agent can perform work on it. These agents are critical to sustaining a healthy system and managing system performance.

Disable or stop any unused agents. Also, do not configure these agents to start on system restart. These agents consume system resources, and some are multi-threaded, which can allow for multiple instances, and, if not tuned properly, can either restrict or overrun system resources. On a system that has a minimum of 2 servers (JVMs), most agents must run on a designated process server that does not have users logging in to it. Agents that require user connectivity, such as the Data Import Agent, are an exception to the rule for running an agent on the process server.

The following sections describe properties that are related to agents and are managed by editing the `TRIRIGAWEB.properties` file or by using the Administrator Console managers. For more information about agents, see Business process agents.

The following table shows the suggested startup locations and descriptions of the agents.

Table 1. Agent startup locations and descriptions

Agent	Location	Description
Data Import Agent	Application Server	• Looks for all tab-delimited files that are uploaded and imports the data into the platform. • On all application servers, if there are multiple application servers. • Must be turned on only if you are using Data Integrator (DI). For more information, see *Multi-Threaded Agents*.
DataConnect Agent	Application Server	• Looks for DataConnect (DC) jobs in the Job Control table that are ready to run. When the agent finds a job, it creates an appropriate smart object for the job. Then the agent posts an asynchronous workflow event to initiate the workflow that pulls the external data into the IBM TRIRIGA database tables. • This agent is on an application server because it needs access to the `userfiles` directory. • Turn on this agent only if you are using DataConnect.

Table 1. Agent startup locations and descriptions (continued)		
Agent	**Location**	**Description**
Object Migration Agent	Application Server	• Migrates TRIRIGA objects from one environment to another environment. • Regardless of how many application servers exist, only one instance of this agent can run. • Turn on this agent only if you are performing an object migration (OM) task. If left on constantly, the system might make unscheduled application changes.
Object Publish Agent	Application Server	• Publishes TRIRIGA objects in the platform. • Regardless of how many application servers exist, only one instance of this agent can run. • Turn on this agent when you are changing application objects in the Data Modeler.
Platform Maintenance Scheduler. Formerly Cleanup Agent.	Process Server	• Conducts data cleanup and runs an analysis on the database. Removes all data in the state of null and removes DataConnect (DC) jobs and staging table entries that are obsolete or completed. Cleans up completed workflow instances that do not have any user-operable tasks, such as user tasks and approval tasks, within the workflow. • This significant agent must always be turned on. Review the server log regularly to confirm its proper execution. For the location of log files, see Identifying performance problems. • The cleanup commands in the Administrator Console are different per database type. These commands can be modified and expanded based on system requirements.
Extended Formula Agent	Process Server	• Looks for and processes queued extended formulas. • Significant agent that must always be turned on.
Formula Recalc Agent	Process Server	• Recalculates paths. • Must always be turned on.
Incoming Mail Agent	Process Server	• Downloads mail from a Post Office Protocol 3 (POP3) server or Internet Message Access Protocol (IMAP) server and converts them into email message records. • Turn on this agent only if you have an IBM TRIRIGA Connector for Offline Forms license and are using the Offlining component.
Report Queue Agent	Process Server	• Retrieves queued report requests, processes the report, and notifies the user. • On the process server where BIRT reports are processed, if a separate BIRT process server is used. • Turn on this agent only if you are using queued reports. No reports are delivered this way with the as-delivered TRIRIGA. For more information, see the *Multi-Threaded Agents* section.

Table 1. Agent startup locations and descriptions (continued)

Agent	Location	Description
Reserve SMTP Agent	Process Server	• Receives and processes reservation emails that Microsoft Exchange sends. This SMTP receiver service allows Exchange to communicate with TRIRIGA and allows it to manage resources in Reserve. A Send connector in Microsoft Exchange is configured to forward any email address with the reservation-specific subdomain to the TRIRIGA application server that runs this SMTP agent. • Turn on this agent only if you have an IBM TRIRIGA Workplace Reservation Manager (Reserve) license and are using the Reserve product.
Scheduler Agent	Process Server	• Looks for and processes all scheduled and recurring events in the platform. • Significant agent that must always be turned on.
SNMP Agent	Process Server	• Receives and processes Simple Network Management Protocol (SNMP) traps. An analytic event from a building equipment asset is created for an SNMP trap. When the SNMP trap is received, a configured workflow is triggered. This workflow is given information on the SNMP trap by using triSnmpPdu and triSnmpPduVariable system business objects. SNMP traps include traps that are used for the IBM TRIRIGA Real Estate Environmental Sustainability Impact Manager. • Turn on this agent only if you are using SNMP traps. For more information, see the Prerequisite setup for using TRIRIGA Real Estate Environmental Sustainability Impact Manager.
Workflow Agent	Process Server	• Processes queued workflow events and the asynchronous workflows that are registered for those events. • Significant agent that must always be turned on.
Workflow Future Agent	Process Server	• Processes actions from a workflow that are set up to trigger at a future date. • Lightweight agent that must always be turned on. • Used by many applications including, but not limited to, BIM, Job Scheduling, Reserve, and Lease.
Workflow Notification Agent	Process Server	• Looks for and processes workflow notifications in the platform, including those notifications to be sent at a scheduled time. • Significant agent that must always be turned on.

Disabling Agents

Disabling agents that are not needed limits the use of system resources and keeps them from being started:

- **`AGENTS_NOT_ALLOWED`**: This property limits the agents that administrators can use.
  - The value is a comma-delimited list of agents that are not allowed to run on this server. For an example, or for the names of the agents to use in this list, see the comments in the `TRIRIGAWEB.properties` file.
  - A blank value allows any agent to be started on a server but does not start any agent automatically.

Multiple JVMs Per Server

If you plan to have multiple JVMs on the same machine, set the following properties to allow for unique names per server instead of the default hostname and ID.

INSTANCE_ID
Overrides the default machine ID.

- When two or more TRIRIGA servers are running on the same physical machine, **INSTANCE_ID** must be unique for independent agent management.
- Leave a blank value if you are running a single instance per physical machine.

INSTANCE_NAME
Overrides the default machine name.

- When two or more TRIRIGA servers are running on the same physical machine, **INSTANCE_NAME** must be unique for independent agent management.
- Leave a blank value if you are running a single instance per physical machine.

Multi-threaded agents

Multi-threaded agents allow you to set a limit on how many threads each agent uses. Increasing the following threads can increase the JVM heap usage, but also increase the JDBC connections and the load on the database server.

These agent threads can also be managed dynamically without a restart by the Threads Manager in the Administrator Console.

The following list shows which agents have their own unique maximum thread count per server.

Data Import Agent
This multi-threaded agent controls data loads by using the Data Integrator tool. The thread count is also used by IBM TRIRIGA Connector for Business Applications (CBA) to allow for throttling integration requests. The agent does not have to be enabled to use the Connector for Business Applications (CBA).

- Property: **DataImportAgentMaxThreads**
- Recommended Value is 2 to 8 depending on the need.

Report Queue Agent
This agent retrieves queued report requests then runs the report and notifies users. For more information on queued reports, see *Queued Reports (Asynchronous)*.

- Property: **ReportQueueAgentMaxThreads**
- Recommended Value is 1 to 4 depending on the need. If unused, set this agent to 1.

Scheduler Agent
This agent looks for and processes all scheduled and recurring events in the system. This agent must always be on, but the threads that are needed are low.

- Property: **SchedulerAgentMaxThreads**
- Recommended Value is 1 to 4 depending on the need.

Workflow (WF) Agent
This multi-server, multi-threaded agent processes queued workflow events and runs the asynchronous workflows that are registered for them. Setting the Workflow Agent maximum threads to a large value can slow the system. Typically, the maximum threads value for the Workflow Agent does not exceed more than 2 to 4 times the CPU core count on the database server but can be adjusted higher if tested properly. For more information, see *Workflow Agent* and *Workflow Performance*.

- Property: **WFAgentMaxThreads**
- Related Property: **WF_AGENT_MAX_ACTIVE_PER_USER**
- Recommended Value is 4 to 32 depending on load.

CAD Integrator

This setting controls server-side work when syncing attached CAD drawings from CAD Integrator/Publisher. The number of CAD Integrator threads that the platform allocates to CAD Integrator/Publisher is managed by the Threads Manager in the Administrator Console.

- Property: **CadIntegratorMaxThreads**
- Recommended Value is 5.

Workflow agent

Multiple workflow agents:

Do not implement multiple Workflow Agents unless the additional agents are configured for dedicated users or groups. For more information, see *Limiting the Number of Workflow Agents*.

Take the following actions in an environment with multiple workflow agents:

- Add and change the **WF_AGENT_SLEEPTIME** property value at the direction of IBM TRIRIGA Support or if you see contention on the WF_EVENT table in the database.
- If you change the **WF_AGENT_SLEEPTIME** property, make sure that each process server running the Workflow Agent has a unique value that is not duplicated on any other server that is connected to the same database. This setup prevents Workflow Agents from waking up at the same time and eases contention on the WF_EVENT table by giving more control for when the Workflow Agent wakes up. On each server, set a different value. But if the property is not set, the default value is a random number between 4500-5500 milliseconds.
- Add the **WF_AGENT_SLEEPTIME** property to the TRIRIGAWEB.properties file to change the time when the Workflow Agent wakes up and checks for new events. If events are already in the queue, the **WF_AGENT_WAITTIME** property waits before picking up new workflows that are already in the queue.

The Workflow Agent must always have at least one instance running to process asynchronous workflows. Almost every application process in TRIRIGA uses synchronous and asynchronous workflows, including all types of integrations, especially CAD Integrator/Publisher. System performance is affected by the Workflow Agent if it is not processing asynchronous processes quickly enough. If the Workflow Agent properties are not tuned properly, then the workflow queue can get backed up and grow quickly.

Use the Administrator Console to configure all the workflow tuning properties.

Use the following methods to manage Workflow Agent performance:

- Multiple Workflow Agents: If multiple agents are set up, you can designate specific users for a Workflow Agent exclusively or to give priority. Do not implement multiple Workflow Agents unless the additional agents are configured for dedicated users or groups. For more information, see *Limiting the Number of Workflow Agents*.
- Tune the threads and threads per user: With multiple agents, configure threads independently. A typical scenario for having multiple is when you have one process server that is configured as a general process server and another process server for integrations or data loads with a dedicated user to process these items.
  - Property: **WFAgentMaxThreads**
  - Property: **WF_AGENT_MAX_ACTIVE_PER_USER**. If you have a Workflow Agent that is assigned to a single user, for example, a unique user for integration purposes, set this value equal to the value of **WFAgentMaxThreads**. For general-purpose process servers, the workflow max active per-user property must be set to 70% of the agent maximum threads, which prevents a single user from maxing out the use of workflow threads.
- Workflow Instance Saving: Set the workflow instance recording property **WF_INSTANCE_SAVE** to ERRORS_ONLY to achieve the best workflow performance. For more information, see *Workflow Performance*.

Platform Maintenance Scheduler

The Platform Maintenance Scheduler performs multiple system cleanup activities at a specified time each day. The Platform Maintenance Scheduler is a critical agent in sustaining your TRIRIGA system. This agent

must always be enabled to help maintain system health. By default, the Platform Maintenance Scheduler performs the following activities:

- Clean up BO instance records that are stale, such as records marked for deletion and temporary objects.
- Clean up workflow instance data.
- Clean up scheduled events.
- Clean up Document Manager data.
- Run cleanup commands, for example **`database analyze`**.

The following cleanup commands are maintained in `TRIRIGAWEB.properties` file. You can configure the cleanup commands by using the Platform Maintenance Scheduler Manager in the Administrator Console. The commands are the only components that can be turned off if you choose to run them externally from the agent.

`CLEAN_HOUR`
Specifies the hour, from 0 to 23, at which the Platform Maintenance Scheduler Cleanup Agent starts on the server.

- Recommended value: A time when the system has the least number of users.
- Default value: `0`

`CLEAN_TIMEOUT`
Specifies the number of minutes that the Platform Maintenance Scheduler can run.

- Default value: `0`

`WF_HISTORY_RETENTION_DAYS`
The Platform Maintenance Scheduler deletes workflow instances that are not waiting on a user or on approval tasks that are older than this number of days. This value does not have a large impact if the **`WF_INSTANCE_SAVE`** property is set to `ERRORS_ONLY`. But on a system that is saving many workflow instances, this value can significantly impact system performance.

- Recommended value: 5
- Default value: `10`

`DC_HISTORY_RETENTION_DAYS`
The Platform Maintenance Scheduler deletes completed or obsolete DataConnect jobs that are older than this number of days. If you do not use DataConnect, then changing this value does not affect your system.

- Recommended value: 5
- Default value: `10`

Extended formula agent

Ensures that records are not put into the Extended Formula queue when it is not required. Objects that are being loaded must have their formulas calculated once at the end of the process.

Incoming mail agent

Invalid email addresses in the TO: or CC: line can impact the acceptance and delivery of the individual email messages. The agent attempts to process all mail that is sent to the inbox. If an email contains an invalid address, that message might be removed from the inbox and not processed, and the agent continues processing the other messages in the inbox.

For outgoing mail, an invalid email in the TO: or CC: line might cause that message to not be sent, and an error is written in the log. Other outgoing notifications in the queue continue to be processed.

Reserve performance

Apply best practices to increase the performance of reserve queries within your Reserve application. The query response time can vary greatly depending on the data composition and setup of the application.

Filtering

When running a reserve query, apply filters to ensure the query searches against the smallest result set possible. Fewer records in the result set mean the reserve query needs to check the availability of fewer records.

For example, floor pagination and floor filtering might be added to a building to increase performance. If a reserve query is changed to search a floor instead of the entire building, it improves performance if those spaces are spread out across many floors. Changing the application design to require a user to pick a floor before performing a search and providing a next floor option reduces the reserve query time. Apply filtering of all kinds at every possible instance to ensure that the reserve query checks the smallest result set for availability.

For more information, see Managing reservations and reservable resources.

Empty filter

In many applications, a reserve query is run when the form initially loads, and immediate results are returned. This behavior impacts the form load time. Eliminate this behavior by setting up the application to filter against a building that has zero spaces. This placeholder "building" can be named <Select a building>, and this building can be added to the form and set as the default building when the form loads. This filter results in faster form loading times. When the user changes the building selection, the query then searches for available spaces, which removes the execution of the query when the form loads.

Enable directed search

By setting the quantity in the triAvailabilityResultsLimitNU numeric field, the direct search feature enables a reserve query to stop processing and return results when a set number of records are found. Enable direct search by adding the triAvailabilityResultsLimitNU field to any object and form where the reserve query is being run. The platform looks for this field and uses the value to limit the results.

For example, if a site that is being searched has 1000 available spaces, then set the triAvailabilityResultsLimitNU field to 25. The search stops processing when the first 25 results are found. This feature allows users to search a large results set and maintain performance. If the field is 0, then the query returns all results. This field can also be incorporated into the application design to enable a user to specify the number of records they want to be returned. For consistent searches, the field can also be hidden with a default value. From a usability perspective, the user sees immediate results and the reserve query does not process any longer than necessary.

Enable multi-threading

Multi-threading enables a system administrator to specify the total number of threads that are available to the reserve query and how many threads each reserve query can consume. Tune the following properties based on how many concurrent reservations you expect to run at one time.

Add the following properties to the `TRIRIGAWEB.properties` file:

`ENABLE_CONCURRENT_AVAILABILITY`
If set to `true`, multi-threading is enabled.

`CONCURRENT_AVAILABILITY_POOL_SIZE`
Specifies the total number of threads that are available to the reserve query.

`CONCURRENT_AVAILABILITY_REQUEST_BATCH_SIZE`
Specifies how many threads each reserve query can consume.

The following are example values from the TRIRIGAWEB.properties file:

```
# Determines whether process Reserve Queries concurrently.
#
ENABLE_CONCURRENT_AVAILABILITY=true

# If ENABLE_CONCURRENT_AVAILABILITY is true, this is the maximum number of system wide
# threads that will be used in processing availability.
#
CONCURRENT_AVAILABILITY_POOL_SIZE=200

# If ENABLE_CONCURRENT_AVAILABILITY is true, this is the maximum number of threads each
# availability request can use to process availability.
#
CONCURRENT_AVAILABILITY_REQUEST_BATCH_SIZE=10
```

Debugging reserve

To debug the performance of directed search and multi-threading, add the following categories as manual categories, to the Platform Logging Manager in the Administrator Console.

- com.tririga.platform.concurrent
- com.tririga.platform.query.reserve

Performance indexes

The following indexes improve Reserve performance in IBM test environments:

- *Reserve indexes* in Tuning the IBM Db2 indexes
- *Reserve indexes* in Oracle database
- *Reserve indexes* in Tuning the Microsoft SQL Server indexes

Avoid all-day meetings

When you use the Reserve application, do not use all-day meetings, especially when integrating Reserve with Microsoft Exchange. Microsoft Exchange treats all-day meetings as floating meetings with variable start and end times that are based on the viewer's time zone. This treatment is problematic when you are coordinating the usage of shared resources across multiple time zones. TRIRIGA treats all-day meetings as 24 hour meetings that are based on the organizer's time zone. However, it is not a best practice to reserve shared resources for 24-hour increments. So, if possible, train your users to avoid all-day reservations.

Room search page size

By default, the TRIRIGA Room Search add-in for Microsoft Outlook displays 10 available meeting rooms per page. To change this default value, you can select **Tools** > **System Setup** > **General** > **Application Settings**. Open the **Reservation Settings** tab, scroll down to the **Perceptive Reserve App Settings** section, and specify a new value for the **Search Rooms Page Size** field.

For more information, see Installing the Room Search add-in in Microsoft Outlook.

Availability section maximum settings

When the **Availability** section is rendered, the **AVAILABILITY_SECTION_ROW_LIMIT** property in the TRIRIGAWEB.properties file limits the maximum number of results to the specified value. If there are too many results, then a warning about exceeding the row size is displayed. The default value is 50 rows. If the value is set to 0, -1, or another invalid value, then the default value is used.

Important: If the property value is too high, performance issues might occur when the **Availability** section is rendered. Large values might cause memory issues. It is recommended that you update the underlying reserve queries to reduce the number of results or design the filters to limit the number of results under this value. Any specified value greater than the maximum value of 500 is set to 500.

In the **Availability** section, the **CALENDAR_EVENT_MAX_OCCURRENCES** property in the `TRIRIGAWEB.properties` file limits the maximum number of occurrences that are displayed and created for recurring meetings to the specified value. The default value is 20 occurrences. For example, if you create a recurring event that occurs every week for a year, which is 52 occurrences, then only 20 occurrences are displayed and created in the **Availability** section. To change this limitation, you can change the default value to another value.

Important: If the number that you specify for the property value increases, then the probability that performance degrades also increases.

For more information, see Setting up reservation results in the Availability section.

Query and report performance

Use query and report performance solutions for accessing and presenting the data that TRIRIGA maintains.

Overview

The Report Manager can produce simple tabular reports, queries, and graphs that combine data from multiple records into a single presentation. If you need a multiple-record presentation that is beyond the capabilities of the Report Manager, you can use the Advanced Reporting feature. TRIRIGA uses Eclipse Business Intelligence Reporting Tools (BIRT) as the enabling technology for the Advanced Reporting feature. For more information on reporting for TRIRIGA, see Reporting in IBM TRIRIGA.

When isolated application processes seem slow, conduct a performance log analysis. If you identify slow-running queries as the primary bottleneck, the following recommendations can help with tuning.

Query tuning

Review custom queries and reports for efficient SQL and indexes. Improving the efficiency of user queries also improves the efficiency of the workflows that use them.

Many long-running queries and reports are not properly filtered. Also, users generate most of the filters in queries and reports with run-time filter operators and filter operators that are defined in the Report Manager. Long-running queries and reports can produce inefficient SQL.

In the TRIRIGA Administrator Console, use the **Platform Logging** page to turn on debug-level SQL query information to determine the origin of a query from the TRIRIGA platform.

Threshold: 100 milliseconds:

- Any query that takes more than 100 milliseconds is a candidate for review. Verify the following tips and make changes wherever possible.
- For more ways to track performance, see *Using report run history to track query performance*.

Tips for creating well-tuned SQL queries

IN versus NOT IN
Using NOT IN or != results in a full table scan. Instead, use IN or NOT EXISTS.

Wildcard character
Do not use a leading % wildcard character. % triggers a full table scan.

Index columns
Order index columns so that the number of rows that is returned is reduced earlier. For example, using a Boolean type column in the first position of the index eliminates about half the rows. Using a column such as `SPEC_ID` in the first position also gives a smaller result set.

Reverse association
Setting the reverse association as No provides a significant performance improvement. If the reverse association must be Yes, then understand why it is needed. For more information, see section *Reverse association queries*.

Association filter
Using a separate association per module or per business object is better than against all modules or all business objects.

Association filter queries
Check whether the query can be changed to a multiple business object query by using an association. This query is faster than using association filter queries.

Multiple business object (BO) queries versus direct smart section
Depending on some factors, you might need to decide on when to use a multi-BO query versus choosing the smart section fields directly. Try running queries both ways, either by using the smart section directly in the field chooser or by adding the BO using the exact association as a multi-BO query.

Nested query optimization
If a query has more than two nested levels, analyze the requirements and determine whether the levels can be reduced by adding filters.

Use system variables correctly
Make sure that system variables such as *$$RECORDID$$*, *$$USERID$$*, and *$$ORGANIZATION$$* are correctly used.

Portal Section, Manager, Master Detail, and Find Queries
Select the **Prompt Before Query** option in the query definition for large datasets. Avoid returning all records with all statuses. Instead, use as many design-time filters as possible to avoid full table scans.

Sorting
Reduce the number of sorting elements to reduce the system resources that the query uses.

Where used
Using the **Where Used** tab in queries helps to determine the places it impacts before the change.

Home portal and other portals

The Home portal has sections that contain queries and reports. Ensure that any portal section queries are tuned to avoid performance issues. Tune all predefined queries for optimal performance across the system.

Many portal sections on the Home portal or any other portal can contribute to slow performance. In the portal, set the portal section to be collapsed by default whenever possible, so a limited number of sections are open on any portal. Portals that are set to auto refresh can also cause slow performance.

Fields and extended formulas

Threshold: 100 milliseconds:

- Review any extended formula that takes more than 100 milliseconds. Edit the formula for improvements.
- Association queries might take a longer time for calculations. Sometimes, improving the association queries increases the performance calculations.

Review the following checklist to optimize the performance of fields and extended formulas:

Association queries
Check the association queries used in the formula. See whether the queries can be improved.

Regular formula
See whether it is possible to change the formula to a regular formula.

Field requirements
Verify whether the field with the extended formula is required for the functions of the business object or form. Extraneous extended formula calculations can be introduced when a field is copied from a business object in which the extended formula calculation is used into a place where the result is not needed. For example, straight-line calculations are required in a lease abstract but may not be required for the triRentStraightLineNU field

Associated data
Consider using the query on an extended formula to get the associated data instead of using the field type to get the data that is being associated. Using a query instead of a field improves performance.

Sorting
Avoid sorting on the query when used for an extended formula. For more information, see *Query Sort Order*.

Where used
Using the Where Used information on fields helps to determine whether this field is used on a form.

Reverse association queries

Module Level Associations (MLA):

If your database is converted to enable Module Level Associations (MLA), the **ALLOW_REVERSE_ASSOCIATION** property is no longer supported. The Reverse Association flag in queries and reports is ignored, and only forward associations are allowed in queries and reports.

For performance reasons, the Reverse Association flag is deprecated.

The **ALLOW_REVERSE_ASSOCIATION** property is added to TRIRIGAWEB.properties. The default for this setting is TRUE, which allows reverse associations to be run in queries and reports as defined in Report Manager. If you want to disable reverse associations from being run for testing purposes, set **ALLOW_REVERSE_ASSOCIATION** to FALSE so that no queries are run with reverse associations. Perform this action on a test environment first to verify the integrity of data coming back from queries. Usually, the flag is not necessary, and in cases that it is, some minor metadata changes can be made to render it unnecessary.

For any queries or reports that are defined in the Report Manager with the Association Filter set with the Reverse Association flag, the **ALLOW_REVERSE_ASSOCIATION=FALSE** setting ignores this flag and interprets and runs queries as if this flag was not set, so only forward associations are returned.

The query that is generated to return both the forward and reverse association can lead to poorly performing queries that can consume significant CPU, which can slow down other queries and operations on the database. Usually, it adds unnecessary overhead without affecting the query results. The platform does not allow new queries to be created with the Reverse Association flag, nor does it allow queries to be copied with the flag.

If you want to run your system with this property set to FALSE to evaluate the performance implications, do so in a development or test environment first. You might encounter a query on a manager, query section, or workflow that no longer has the results you were expecting. If this event happens, first examine the **Association** tab of the records in question and note the correct forward association string. Then open the query in Report Builder, and in the **Association Filter** examine the association string that is used. If the two strings are different, update the report to use the correct association string. If your records have only a one-way association, create the forward and reverse association in Association Manager and then update your data by using a workflow to set the forward and reverse association.

Query sort order

Case-insensitive sorting on query results cause reduced performance when processing a large volume of data. You can turn off the forced case-insensitive Order By on query results by using the **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT** property and setting in the TRIRIGAWEB.properties file.

Restart the application server for changes in the TRIRIGAWEB.properties file to take effect. Turning off the case-insensitive Order By might result in improved database performance. But the sort results might be ordered differently.

The following examples show how Db2 case-insensitive and case-sensitive sorting can give different results:

- **REPORT_CASE_SENSITIVE=FORCE_CASE_INSENSITIVE**: Case-insensitive sorting. Note the `UPPER(T1.NameTX)`.

```
SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM
T_TEST T1 WHERE T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY
UPPER(T1.NameTX)
```

- **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT**: Native database case-sensitive sorting.

```
SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM
T_TEST T1 WHERE T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY
T1.NameTX
```

Table 2. Example of Db2 sort: Case-insensitive versus case-sensitive

Case-insensitive name	Case-sensitive name
Las Vegas	Las Vegas
Lasiter	Lasiter
PHEASANTON	PHEASANTON
Philadelphia	PHILADELPHIA
PHILADELPHIA	PLEASANTON
Pleasanton	Philadelphia
PLEASANTON	Pleasanton
Riverside	Riverside
RIVERSIDE	RIVERSIDE
Testing	Testing

The following are examples where SQL Server case-insensitive and case-sensitive sorting give the same result:

- **REPORT_CASE_SENSITIVE=FORCE_CASE_INSENSITIVE**: Case-insensitive sorting. Note the `UPPER(T1.NameTX)`.

```
SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM
T_TEST T1 WHERE T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY
UPPER(T1.NameTX)
```

- **REPORT_CASE_SENSITIVE=NATIVE_DB_CASE_SORT**: Native database case-sensitive sorting.

```
SELECT T1.NameTX AS T1_1002, T1.ControlNumber AS T1_1001, T1.SYS_TYPE1 AS
T1_SYS_TYPE1, T1.SYS_GUIID AS T1_SYS_GUIID, T1.SPEC_ID AS T1_SPEC_ID FROM
T_TEST T1 WHERE T1.SYS_PROJECTID = 1 AND T1.SYS_OBJECTID > 0 ORDER BY
T1.NameTX
```

Table 3. Example of Sql Server sort: Case-insensitive versus case-sensitive

Case-insensitive name	Case-sensitive name
Las Vegas	Las Vegas
Lasiter	Lasiter
PHEASANTON	PHEASANTON
Philadelphia	Philadelphia
PHILADELPHIA	PHILADELPHIA

Table 3. Example of Sql Server sort: Case-insensitive versus case-sensitive (continued)

Case-insensitive name	Case-sensitive name
Pleasanton	Pleasanton
PLEASANTON	PLEASANTON
Riverside	Riverside
RIVERSIDE	RIVERSIDE
Testing	Testing

Using report run history to track query performance

Track the performance and monitor who runs queries and reports in the Report Manager. Set a flag on the report definition that tracks who runs the report and the time it took to retrieve the information from the database.

To enable the report run history, open the report in the Report Manager, click the **General** tab, and select **Track History**.

Report Run History:

Enabling this tracking causes a slight overhead, but use it with caution. After tracking is enabled, tell your users to use the system for a while so that some data is saved. After a few days, analyze what is happening in your system.

The data can answer the following questions:

- Which reports run the longest overall?

```
SELECT MAX(duration), rth.rep_name
FROM REGRESSION35X.rep_history rh, REGRESSION35X.REP_TEMPLATE_HDR rth
WHERE rth.REP_TEMPLATE_ID = rh.REP_TEMPLATE_ID
GROUP BY rth.rep_name
ORDER BY MAX(duration) DESC;
```

- Of the reports that run, which have the average longest run times?

```
SELECT AVG(duration), rth.rep_name
FROM REGRESSION35X.rep_history rh, REGRESSION35X.REP_TEMPLATE_HDR rth
WHERE rth.REP_TEMPLATE_ID = rh.REP_TEMPLATE_ID
GROUP BY rth.rep_name
ORDER BY AVG(duration) DESC;
```

- For the users in the system, who runs the most reports?

```
SELECT COUNT(*), uc.user_account
FROM REGRESSION35X.rep_history rh, REGRESSION35X.user_credentials uc
WHERE rh.user_id = uc.user_id
GROUP BY uc.user_account
ORDER BY COUNT(*) DESC;
```

- Of the users in the system, who ran the longest reports?

```
SELECT MAX(duration), uc.user_account
FROM REGRESSION35X.rep_history rh, REGRESSION35X.user_credentials uc
WHERE rh.user_id = uc.user_id
GROUP BY uc.user_account
ORDER BY MAX(duration) DESC;
```

Advanced BIRT reporting

Follow the same principles for reporting when designing and building BIRT reports. Use the following optional features of the TRIRIGA platform to improve the performance of the front-end application server.

BIRT Report offloading

Specify BIRT reports to be offloaded and processed by a separate TRIRIGA server that is configured as the BIRT process server. If you do not configure a separate BIRT process server, the report processing is done locally.

The following system properties enable BIRT report offloading. Report offloading can eliminate system resource usage by BIRT reports on a server with user sessions by sending the report processing to another JVM. The offloaded server is simply another TRIRIGA server that can be dedicated to this task or be the process server that is already set up for backend agents.

- **BIRT_PROCESS_SERVER_HOST_NAME**
- **BIRT_PROCESS_SERVER_PORT**
- **BIRT_PROCESS_SERVER_LISTENING_PORT**

For more information, see Configuring BIRT Process Servers or Reporting in IBM TRIRIGA.

Asynchronous queued reports

To run a BIRT report asynchronously and notify the user when the report is ready to view, mark the report as queued. For reports that take more than one minute to run, consider running them asynchronously instead of synchronously. Running a report asynchronously runs the report on the server when the server is not busy with other tasks. If a queued report has runtime filters or BIRT filters, they are presented to the user before the report is queued, as they are for a nonqueued report.

You can start and access queued BIRT reports in My Reports, Report Manager, manager query lists, and query sections. You cannot start and access queued BIRT reports in portal sections and form actions. But if a queued report is attached to a portal section, the system starts it in a nonqueued manner.

When a queued report finishes processing, the system sends a notification to the user's Notifications portal. To access the report, the user clicks the hyperlinked subject line. A queued BIRT report that is attached to a workflow notification runs with the notification. However, if the report contains filters that do not have a default value, the report generation might end in error because there is no user to provide the values. You need to set the default values for all filters when you define such a report.

Maximum available server memory

Change the maximum percentage of available server memory that can be used to create the BIRT report query results. Use the **BIRT_MEMORY_USAGE_LIMIT** system property. For information on using this property, see *TRIRIGAWEB.properties* in "System properties" on page 77.

Geography and organization security check

You can choose an alternative way to run a Geography and Organization security check on a query. The generated SQL is less resource-intensive and might perform more efficiently depending on your Geo-Org hierarchy structure. To use this alternative method, you must comply with the following limitations:

- The triPathSY field must be part of the Organization and of all Geography business objects.
- The values of triPathSY fields on Organizations and all Geography records must be consistent with the hierarchy.
- The Organization and Geography that is used within any security group must contain fewer characters than the size of triPathSY field on the Organization and Geography business objects.

This alternative method uses the **GEO_ORG_SECURITY_CONTRIBUTE_BEHAVIOR** property in the `TRIRIGAWEB.properties` file. By default, this property is set to `COALESCE` to use the

```
IBS_SPEC_STRUCTURE
```

and the recursive subselects to contribute Geo-Org security. However, setting this property to PATH offers the alternative method by contributing simpler SQL that runs a compare against the triPathSY field, which is why triPathSY must be consistent with the `IBS_SPEC_STRUCTURE` to ensure proper Geo/Org security.

You can determine whether your Geography and Organization hierarchies are in sync with their triPathSY values by using the Database Manager in the Administrator Console. In the Database Manager, click**Database Admin Tasks**. Then select **Check Organization Hierarchy** or **Check Geography Hierarchy**. Read the message and click **OK** to confirm. This check process might take a long time to run depending on the size of your Geo and Org hierarchy. You can monitor the `server.log` file for feedback regarding any out-of-sync entries and process completion.

Workflow performance

Benchmark and analyze workflow performance by using methods such as platform logging and performance log analysis.

Workflow instance data

Workflow instance recording saves the step-by-step execution information to the database. This data also shows the path that is taken in a graphical display, which includes links in task details and information on nested workflows.

Non-production environments:

The primary use of workflow instance recording is for debugging in a nonproduction environment.

Do not use workflow instance recording in a production environment. It causes significant processor usage. It can increase workflow response time by 50% or higher and writes a significant amount of data to the database.

Do not use workflow instance recording when IBM TRIRIGA is performing data loads or batch processing. The saving of instances slows down the processing, uses excess database space, and impacts the Platform Maintenance Scheduler cleanup.

Configure the following properties and values:

- WF_INSTANCE_SAVE
- WF_INSTANCE_SAVE_24HR_THRESHOLD
- USE_WF_BINARY_LOAD

For more information, see “System properties” on page 77

Optimizing workflows

500 and 2000 milliseconds thresholds:

- Review any synchronous workflow that takes more than 500 milliseconds. Based on your user requirements, check whether the workflow can be run asynchronously. Make sure that a workflow does not require synchronous processing.
- Review any asynchronous workflow that takes more than 2000 milliseconds.

Use the following checklist to optimize workflow performance:

Transaction control and multi-record processing:

The Create Record and Modify Records tasks can cause many database commits, but these tasks can also cause excessive heap usage. The specific situation depends on the records, data, and number of records that are being processed. Reducing the number of commits by increasing the number of records per commit can improve performance by reducing the load on the database. However, attempting a large set of records, for example, more than 10,000 records, at one time can cause excessive heap usage. Heap usage is excessive because the task does not commit until all the records are created or modified. To determine the best balance, use the following methods:

- Record Sets: Use record sets whenever possible to minimize the number of database commits.
- Transaction Section: Use the task's **Transaction** section when the record set is large or unknown to commit every *N* records, for example, every 100 records, to prevent excessive heap usage.

Query Task versus Retrieve Records Task

- Use the Query Task when any of the following applies:
  - Using associated records as filters.
  - Getting general information, such as open orders or occupied spaces.
  - The intermediate record set might be large.
- Use the Retrieve Records Task when any of the following apply:
  - Getting records with the largest or smallest value.
  - Accessing financial records with a token.
  - Filtering an existing record set from the result of another step.

End Task versus Stop Task

- A Stop Task indicates the abnormal and immediate stopping of a workflow. When a workflow reaches a Stop Task, no more of its tasks are performed.
- An End Task task indicates the end of a workflow. When a workflow reaches an End Task, the workflow is complete and there are no more tasks to perform. The workflow completes normally, so saving of instances follow the server's or the start task's instance save.
- When developing workflows, use End Task for most instances when you want to exit the workflow gracefully, and all work is done. Use the top Task in only worst-case scenarios because Stop Task causes the entire workflow instance stack to be saved, which is not desirable, especially in data-load scenarios, because saving workflow instances is resource-intensive. If you are not sure which task to use, use the End Task.

Query Task Filter

If possible, when filtering records in a Query task using the list comparison, use the **Associated To** option to reduce the tasks in a workflow.

Associate Records Task

Use the Associate Records Task task only for the forward association. The platform does the reverse association automatically if the association is defined in the Association Manager. However, for performance reasons, the Reverse Association flag is being deprecated in queries. For more information, see *Reverse association queries* in “Query and report performance” on page 87.

Switch task

Use the Switch task to call a workflow only when needed. For example, in a Permanent Save Validation workflow.

Extended Formula

Trigger an Extended Formula only when needed with the**Formula** property. For more information, see *Fields and extended formulas* in “Query and report performance” on page 87.

In-Memory Smart Object (IMSO)

If possible, use the IMSO for helper records that do not need to be saved.

Cascade Read-Only

Use the **Cascade Read-Only** option to eliminate the need to trigger separate workflows to make the child record read only. The Cascade Read-Only flag is set in the Association Manager.

Workflow Analysis Utility

Use the Workflow Analysis Utility to identify any potential workflow issues. The Workflow Analysis Utility analyzes workflow performance and process execution. The utility reads IBM TRIRIGA performance logs and displays performance analytics for workflows, including the following:

- Workflow execution time, which is how long different workflows take to run.
- Process flow, which is the order in which workflows run and what triggered them to run.

Integration framework

Tune the mechanisms that IBM TRIRIGA Application Platform uses to integrate with external applications.

Integration components

The following components are the main integration components:

- Connector for Business Applications (CBA)
- DataConnect (DC)
- Data Integrator (DI)
- Integration Object (IO)
- Open Services for Lifecycle Collaboration (OSLC). For more information, see Integrating data by using the OSLC REST API.

For more information about the integration components, see Integrating data with external applications.

These integration components can significantly affect the performance of the system. Consider these components when you are sizing and tuning your system. The incorporation of both multi-threading and clustering greatly increases throughput in an integrated environment. The TRIRIGA platform uses a multi-threaded integration model. The Connector for Business Applications (CBA) model is set up to process multiple inbound transactions at once. In a large and integrated system, tune the application and the overall environment parameters, including the parameters for the application server.

Also consider the following components when you are tuning integrations for performance:

Dedicated server
Use a dedicated server for multiple integrations.

Off-peak scheduling
Schedule regular integration jobs during off-peak user times.

Session tracking
Turn off the session tracking for integrations.

Workflow execution
Minimize the number of workflows that the integration processes run.

Data loading
For large data loads, migrations, and batch jobs, take the following steps:

1. Schedule a maintenance window when you plan to optimize the system for data loading.
2. Notify users of the maintenance window.
3. Lock out users during the maintenance window.
4. Configure the system for optimal data loading, for example, maximize JVM heap, increase workflow agents and threads, and enable only essential components.
5. After the data load or batch migration is complete, reverse the configurations from step 4.
6. Reverse the user lockout and notify users.

Integration workflow processing and performance

All of these integration components use the workflow engine. The integration components can also be tuned by following the best practice guide lines in the *Agents* section in “System properties” on page 77 and “Workflow performance” on page 93.

Disabling unused operation workflows

When TRIRIGA performs integration operations, such as importing large numbers of floor and space records by using DataConnect (DC), increase the throughput by analyzing which workflows are run as part of the process. Then disable those workflows that are not needed by the operation.

Find out which workflows are run by enabling all workflow instances to be recorded and then inspect them after you go through the process once. However, you can miss instances in many workflows that are run as side effects of creating and associating records. In addition, enabling workflow instance recording might cause process and system overhead. Therefore, avoid enabling all workflows except when you are debugging a part of the processing. Disable all instance recording at all other times. You can set the **WF_INSTANCE_SAVE** property in the `TRIRIGAWEB.properties` file and temporarily override the proprty if needed by using the Workflow Agent Manager in the Administrator Console. For more information, see "Workflow performance" on page 93.

You can also which workflows are run and why by enabling synchronous and asynchronous workflow performance logging and collecting the `performance.log` file from the process server that runs the Workflow Agent. Then, process and analyze the `performance.log` file by using the Workflow Analysis Utility to get an interactive view into what is running, why, and for how long.

Also, if integration processing is done regularly, consider using the **Integration** flag in the workflow start conditions to skip workflows instead of temporarily retiring them.

Limiting the number of workflow agents

Multiple Workflow Agents: Do not use multiple Workflow Agents unless the additional agents are configured for dedicated users or groups.

Running multiple workflow servers allows workflow processing to be fair to all users. It does not necessarily increase the throughput of the number of workflows that are processed. Adding multiple Workflow Agents to a single environment can slow down processing and cause undesirable results if workflows are not written with multi-threading in mind.

The optimum approach is to assign secondary Workflow Agents to dedicated power users who run more workflows than a typical user. Take this approach by using the Workflow Agent Manager in the Administrator Console.

If multiple Workflow Agents remain open, that is, not bound to a user, then a set of workflow instances are picked up in parallel and some might be processed out-of-order. Increasing the number of threads on a single process server results in higher throughput than splitting the threads across two servers. Typically, the bottleneck of performance is the database server rather than the process servers.

If you already have a system that is deployed with multiple Workflow Agents, consider the following options:

- Stopping the secondary agents and increasing the threads on the primary Workflow Agent server to be the sum of the threads across the other servers.
- Restricting the secondary agents to be exclusive for the set of power users.

Hierarchy path mapping

If a database server is slow and you create many hierarchy records by using Data Integrator (DI) and an asynchronous workflow, the triPathSY hierarchy path of some records might not be mapped properly. To resolve this issue, either delete the incorrect records and re-create them or create a patch helper to fix those records.

Graphics

Encourage your users to adopt improvement practices to improve graphics display performance.

Drawing size

The main factor that affects graphics performance is the number of entities that are visible when the graphic is rendered to the user. Often, the relative or perceived physical or digital drawing size has little correlation to these factors.

An entity is an individual object or element, such as a line segment, that exists in a CAD drawing. In MicroStation, entities are referred to as elements. A polyline, which can be composed of multiple line and

arc segments, is also an entity. A polyline is a single entity regardless of how many vertices and segments it contains. Often, CAD drawings contain layers with information about walls, doors, and furniture, that are composed of exploded singular entities rather than polylines.

For example, the chair in Figure 1 looks simple. Visually, it seems to contain only 4 distinct shapes and is represented by 4 polylines. However, in this case, it is composed of 27 individual entities, consisting of disconnected arcs and line segments. The difference between 4 and 27 is not large, but if a furniture layer on a floor plan that contains 500 chairs, then a layer that can be composed of only 2000 entities has 13,500 instead, which increases the load time for that single layer by a factor of nearly 7 times. Normally, any drawing that contains furniture blocks in this state can have the same issue with other components of the drawing, which can have a major overall impact on performance.

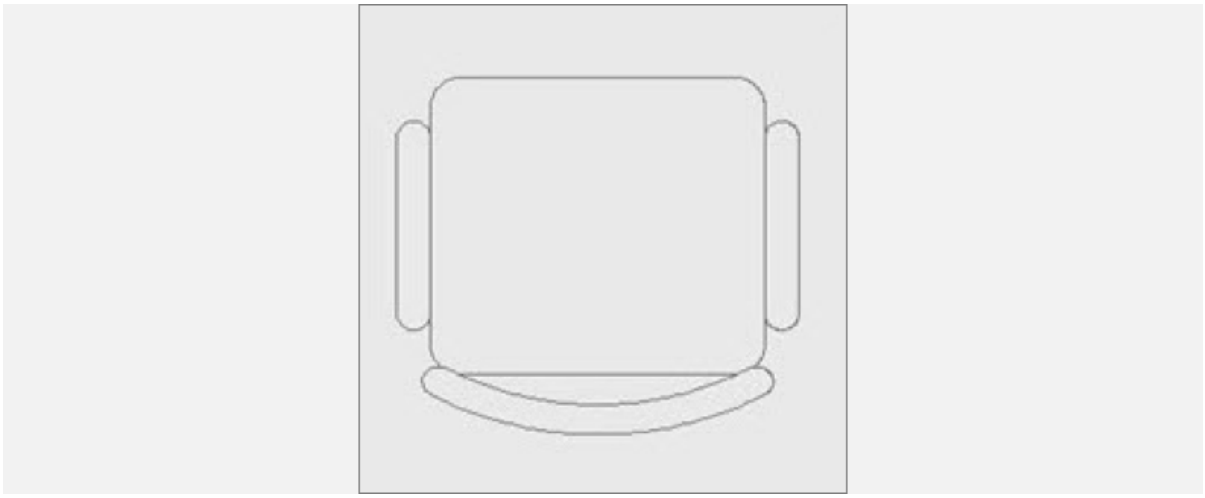

Figure 6. Example of Chair: 4 polylines versus 27 entities

Graphic layer configuration

Improve performance on the TRIRIGA side by limiting what is initially loaded for all users in the system. Limit loading by using Graphic Layer Config records, which are in the Tools menu under Administration, Graphics, Layer Configuration. By default, a record is included that initially turns off all layers with an entity count greater than 1000.

Reduce this count to improve performance. Graphic Layer Config records have no impact on attached layers, so even if you set this value to 1, all of your attached polyline layers still load completely, which is ideal for most use cases, and as a result, graphics across the system loads quickly regardless of their perceived size. Users who require more information at the time of display can use the Layer Manager to turn on the layers they need to view or export.

Simplifying drawings

If the more complex layers that Graphic Layer Config normally turns off are used often, the best way to limit load times is to clean your CAD drawings by joining exploded blocks and other unconnected entities. While this might seem like a large effort, the process can dramatically improve performance. The best way to demonstrate this process is with a real-world example, such as an AutoCAD floor plan.

The AutoCAD architectural reference file in Figure 3 contains excessive geometry, especially on the furniture layer. When published as-is to a graphics section, the drawing renders all the layers in about 50 seconds, where 45+ seconds of that is the furniture layer. To improve the time, start by reducing the number of entities in the furniture blocks.

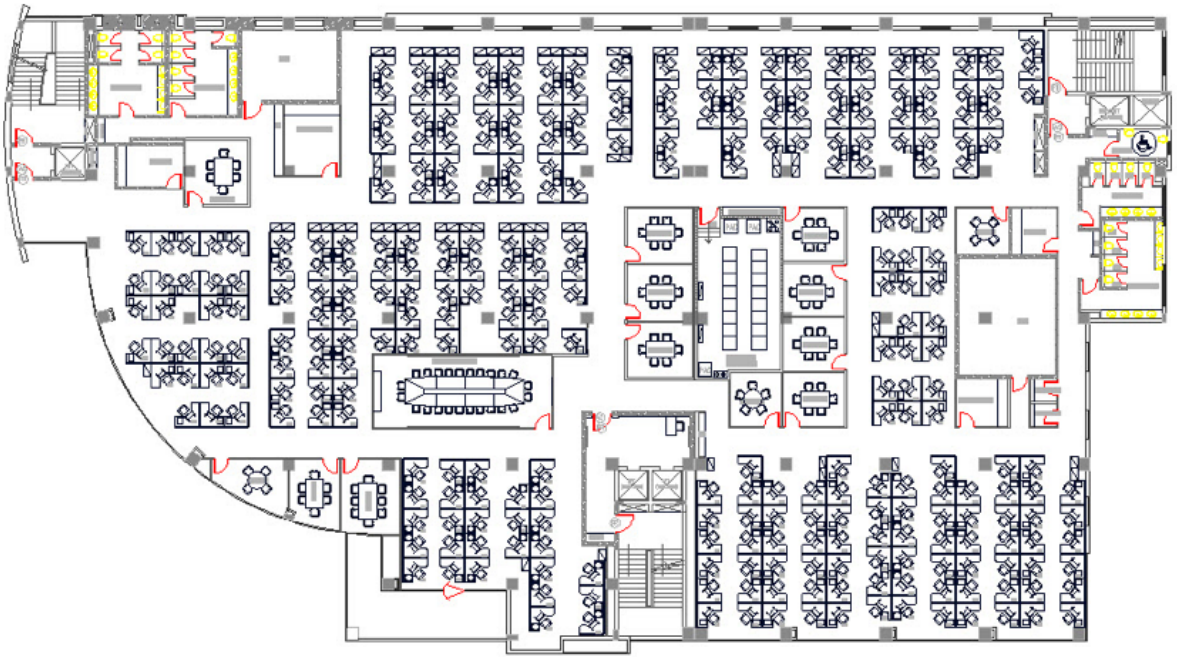

Figure 7. Example of AutoCAD Floor Plan

CAD commands

Repeated furniture vs. reusable blocks:

If furniture is repeated in your drawing that is not in blocks or cells, create blocks or cells so you are not editing the same furniture many times. This example assumes that blocks are already created for the furniture.

In this example, the following desk-and-chair workstation is represented by a block and is repeated in each cubicle 350 times. To display this simple workstation takes101 individual entities for this block, or over 35,000 entities on the furniture layer. In AutoCAD, one quick way to simplify this block is to take advantage of the **JOIN** option of the **PEDIT** command. After running the **PEDIT** command, you can select all of the entities, convert them to polylines if applicable, and join them at a threshold.

In less than 10 seconds, this **JOIN** option reduced the number of entities for this block from 101 to 29. The command also reduced the total number of entities on the furniture layer by over 25,000. Because you use the command line, the command can be automated by using a LISP routine. If you are unfamiliar with writing LISP that can open a set of drawings in a batch, use existing utilities such as StarBatch to do this work for you. Beyond this step, you need your script to call the noted commands on the requisite layers. For more information, see AutoCAD Command Line API Specification.

Manual editing and cleanup

Remove unneeded geometry or redraw some of it by hand. Although this approach might take some time, the desk and chair in their simplest forms can be represented with 2 polylines alone.

The PEDIT JOIN option

To reduce the load time in the graphics section with all layers turned on, use the **PEDIT JOIN** option and as some manual simplification across the entire floor plan.

For the initial display of a graphic, it is best for the typical use case to load only what you need with Graphic Layer Config records. Usually, users use the graphics section for reporting, interacting, or locating attached entities such as spaces. In these scenarios, extraneous detail often clutters the graphic. For users who want the option to display more complex layers with furniture or architectural detail, the load

times across the system can be improved by simplifying your drawings through the built-in features of your CAD application, or through manual editing and cleanup.

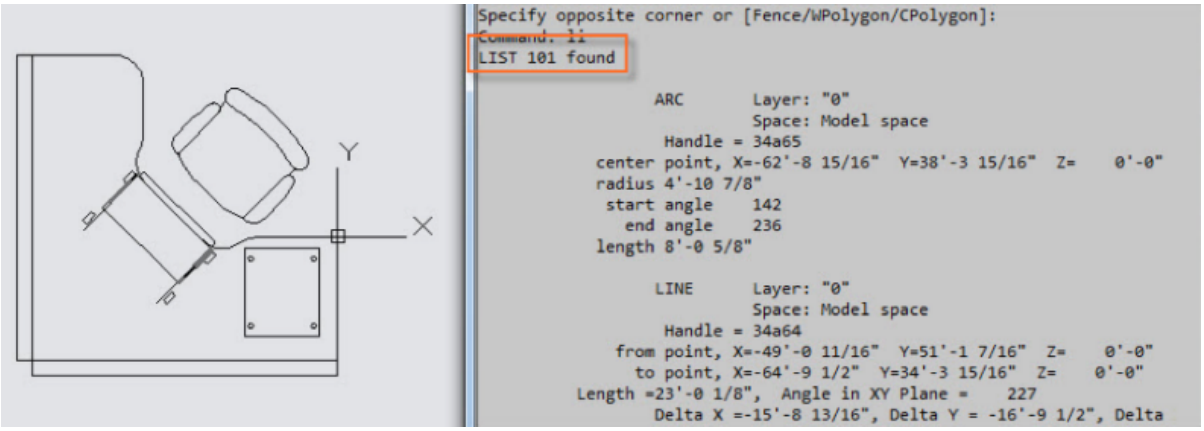

Figure 8. Example of desk and chair workstation: 101 Entities

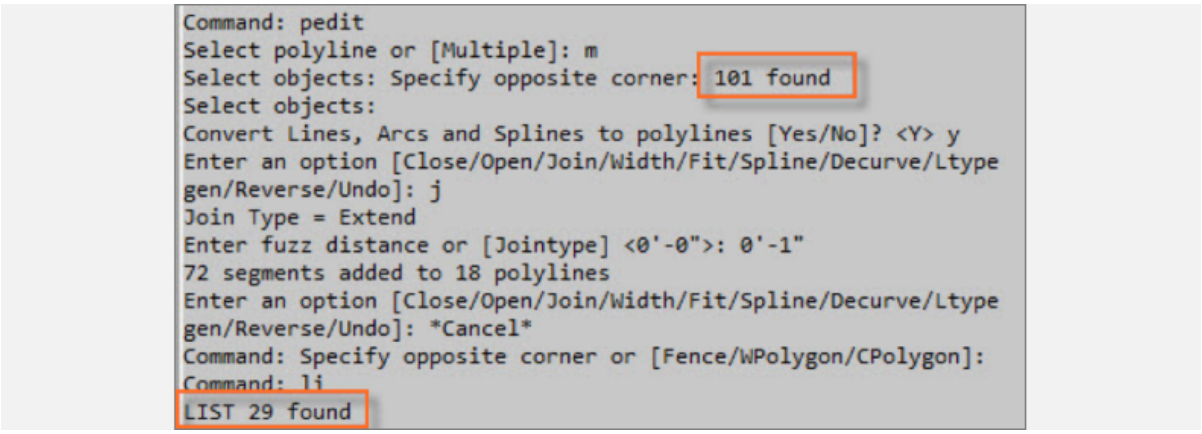

Figure 9. Example of AutoCAD Command Sequence: 29 Entities

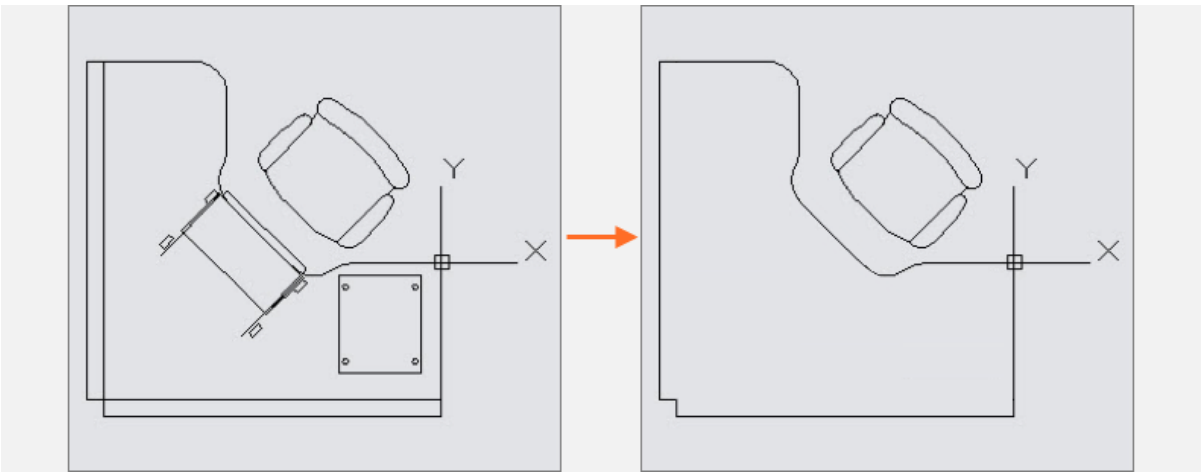

Figure 10. Example of desk and chair workstation: Manual Editing & Cleanup

Chapter 8. Tuning the web server

Use HTTP Server settings to optimize web servers, such as Apache Server or IBM HTTP Server (IHS).

HTTP Server settings

The following settings provide optimum performance in IBM test environments. However, your environment might require different settings. Customize the settings to your environment requirements. For more information on tuning web servers, see your web server documentation.

Configure the following options and values by editing the `httpd.conf` configuration file:

1. Enable the Multi-Processing Module (MPM).
2. Apply the following starting parameters for your testing. The parameters are dimensioned for a server configuration of 32 CPU and 240 GB RAM. Calculate your own values based on your server configuration.

   Note: Values that are based on incorrect calculations can cause memory leakages.

   ```
   <IfModule mpm_event_module>
       StartServers                4
       MinSpareThreads             25
       MaxSpareThreads             75
       ThreadLimit                 64
       ThreadsPerChild             64
       MaxRequestWorkers           128
       MaxConnectionsPerChild      100
   </IfModule>
   ```

3. Set the following KeepAlive timeouts:
   - **`KeepAlive`** On
   - **`KeepAliveTimeout`** 5
   - **`MaxKeepAliveRequests`** 100
   - **`Timeout`** 600
   - **`Proxy timeout`** 1800

Note: Proxy server configuration is outside the scope of this documentation.

Enabling CORS on a web server

If you are using multiple domains in your environment, configure your web server for Cross-Origin Resource Sharing (CORS).

For example, on web servers such as Apache, set the following headers in `/etc/httpd/conf.d/virtual.conf`:

```
Access-Control-Allow-Origin: <TRIRIGA server domain name>
Access-Control-Allow-Credentials: true
```

Chapter 9. Troubleshooting and monitoring

The IBM TRIRIGA Application Platform has platform logging features that log system health statistics. These features and their associated data help you troubleshoot and monitor for performance problems. Troubleshoot in a development or test environment for performance analysis and debugging. If you cannot isolate the problem in a test environment, then you might need to troubleshoot in a production environment.

Preparing to troubleshoot performance problems

Configure a stand-alone server for debugging and enable application platform logging.

Configure a stand-alone server for debugging

Debugging SQL or business object problems, or using detailed logging when you reproduce other problems, can generate large logs and slow the system down. Instead, set up a stand-alone application server on a separate computer for debugging. Using a separate computer means that you can easily stop and start the server to change logging parameters and reproduce problems with a single user.

Enable IBM TRIRIGA Application Platform application platform logging

Platform logging enables real-time debug-level logging.

Start the administrator console. For more information, see Administering with the Administrator Console.

When you select an option and click **Apply**, debug-level logging starts immediately. When you clear an option and click **Apply**, debug-level logging stops immediately. When the server is restarted, the system reverts to the `log4j.xml` configuration settings.

Click the **Roll Log** icon or select one of the **Roll** actions. The system renames the current log file with the current date and time and begins entering new messages in a new file. You can add your own custom logging categories in the `CustomLogCategories.xml` configuration file that is located in the TRIRIGA installation `\config\` folder.

Performance logging is included in the startup classes. To enable the logger, in the `log4j.xml` in the `\config\` folder, add the following category and set the priority value to either INFO or DEBUG:

```
<category name="Startup Duration Log" additivity="false">
   <priority value="[Value]" /> <!-- Set to DEBUG to enable logging -->
   <appender-ref ref="FILE" />
</category>
```

The following example shows log output for the INFO level:

```
2016-06-17 18:23:18,959 INFO [Startup Duration Log](Default Executor-thread-6)
   TRIRIGA Application Started in: 33 seconds
```

The following example shows log output for the DEBUG level:

```
2016-06-17 18:23:18,959 DEBUG [Startup Duration Log](Default Executor-thread-6)
*************************************************************************
                TRIRIGA Application Initialization Timing
-------------------------------------------------------------------------
    HTTPClientConnectionHandler completed in:.....................00 seconds
    UpgradeScripts completed in:..................................02 seconds
    SpringWebInitServlet completed in:............................26 seconds
    DispatcherServlet completed in:...............................00 seconds
    Begin StartupServiceManagerServlet
     - Method 'Upgrade Classes' completed in:.....................04 seconds
    StartupServiceManagerServlet completed in:....................06 seconds
    WidgetServlet completed in:...................................00 seconds
-------------------------------------------------------------------------
TRIRIGA Application started in:...............................43 seconds
*************************************************************************
```

For more information about platform logging, see Administrator console.

Apply the latest patches

Apply the latest available upgrade, patch, fix pack, or test fix for your release of TRIRIGA. Upgrades, patches, and test fixes contain fixes for platform and application issues and can contain features that might improve system performance. Patches also contain new features and parameters that you can use to monitor and debug system performance issues.

Identifying performance problems

Use problem determination techniques and tools to identify any performance problems.

If possible, perform load testing during the implementation phase to identify performance problems. To determine whether there is any performance impact from patches or from data growth over time, you can use load testing after TRIRIGA is in production. If you encounter problems, use the following techniques and tools to identify any performance problems.

Determining the source of the problem

Application server logs

Review IBM TRIRIGA Application Platform server logs for any errors. For load-balanced implementations, pay attention to the distribution of users across the JVMs. Monitor the JVM memory utilization on the application server by enabling verbose garbage collection.

Log files are in the TRIRIGA installation subfolder that is named `\log\`. Log files are appended with the creation date. You can also download and roll over log files from the error logs in the Administrator Console. Several tools are available that can help you to analyze the following log files:

`server.log`
Contains standard TRIRIGA logging.

`security.log`
Contains security-related logging, user logins, and admin user changes.

`ObjectMigration.log`
Contains Object Migration (OM) tool logging.

`performance.log`
Contains performance DEBUG-level logging from the platform logging mechanism.

`systemmetrics.log`
Contains performance information from the Administrator Console.

Web server logs

Review the web server logs for errors. View the maximum connections and total connection attempts to see whether your web server can use as many connections as it needs. Compare these numbers to memory and processor usage figures to determine whether a connection causes the problem and not some other component.

View the following IBM HTTP Server log files:

- `access.log`
- `admin_access.log`
- `admin_error.log`
- `error.log`
- `http_plug.log`

The **`ThreadsPerChild`** parameter specifies the number of threads that are created by each child process. You might need to increase the **`ThreadsPerChild`** setting in the `httpd.conf` configuration file for IBM HTTP Server. The value for**`ThreadsPerChild`** in Windows environments is `2400`.

Database server

Monitor database server memory and instance memory. To assist with tuning issues, gather database traces and snapshots. For more information about Microsoft SQL Server, see “ Microsoft SQL Server database” on page 65. For more information about general database tuning, see Tuning the database server.

Network

Sometimes you need to understand the network speed throughput from different client locations. In the Administrator Console, use the Performance Monitor to test different client locations and compare network speed throughputs. The network Speed Throughput Test link tests the network speed from the server to the current user's computer. In the results, you can see your speed throughput in kilobytes, and then you can see an average comparison with other network structures.

Provide the network Speed Throughput Test link to users without the need for them to log in. To identify a potential network overload, monitor the bytes per second that are processed by the network interface card. If you need more detailed network information to understand bandwidth requirements, there are bandwidth-monitoring tools that provide the ability to analyze HTTP requests, the number of roundtrips between tiers, and TCP/IP packet information. For more information, see Tuning the network.

CPU

Monitor the CPU to ensure that all processors are being used as expected and that overall CPU utilization remains healthy.

Memory

Monitor total memory usage. JVM heap size is the most important memory metric to monitor on application servers. There are several parts to the TRIRIGA platform that, when unchecked or poorly configured, can contribute to a large memory footprint on the application and process servers and can cause the server to encounter an `Out of Memory` situation in which the TRIRIGA server crashes. For more information, see Tuning the application server and Tuning IBM TRIRIGA components.

Manage the following memory-related performance issues:

Memory footprint contributors

The following processes can increase the heap memory on the application and process servers:

- Workflow Instances: When set to ALWAYS, the **`WF_INSTANCE_SAVE`** property consumes a large amount of memory on the application server and slows down the performance of workflows and actions. Set workflow instance saving to ALWAYS only if you are actively debugging workflows. Do not set to ALWAYS for longer than needed.
- BIRT reporting: The BIRT engine consumes a large amount of heap memory when exporting large datasets.
- DataConnect task: When you build a workflow with the DataConnect task, commit no more than 10 records at a time in the **Transaction** section. Tune this setting to reflect on the degree of integration for your IBM TRIRIGA implementation.

Diagnosing `Out of Memory` errors

When an `Out of Memory` error occurs, restart the application and process servers. Generate a heap dump at the point of the `Out of Memory` error.

In WebSphere Application Server Liberty profile, the heap output file is created in the default directory, `${server.output.dir}`.

Memory Analyzer

Analyze the heap dump by using the Eclipse Memory Analyzer Tool (MAT) to find the memory leaks and reduce memory consumption.

If your heap dump is larger than 6 GB, the MAT consumes a lot of memory. Your workstation needs at least 16 GB of RAM. Close all other applications and configure the Eclipse `config.ini` file to set its own max heap size to 15 GB by using **`-Xmx15G`**.

The **Overview** section gives you insight into what the heap contains. Typically, the first- level or second-level objects explain what consumed the heap. Figures 1 and 2 are examples of BIRT report

and workflow instance `Out of Memory` heaps. You can identify the main area that consumed the heap in the **Problem Suspect** section.

The Memory analyzer workflow instance `Out of Memory` heap message is as follows:

```
1,051,855 instances of "com.tririga.platform.workflow.runtime.WFContext", loaded
by "war:IBM-TRIRIGA_Build-227459/ibm-tririga.war" occupy 3,637,218,776 (86.70%) bytes.
```

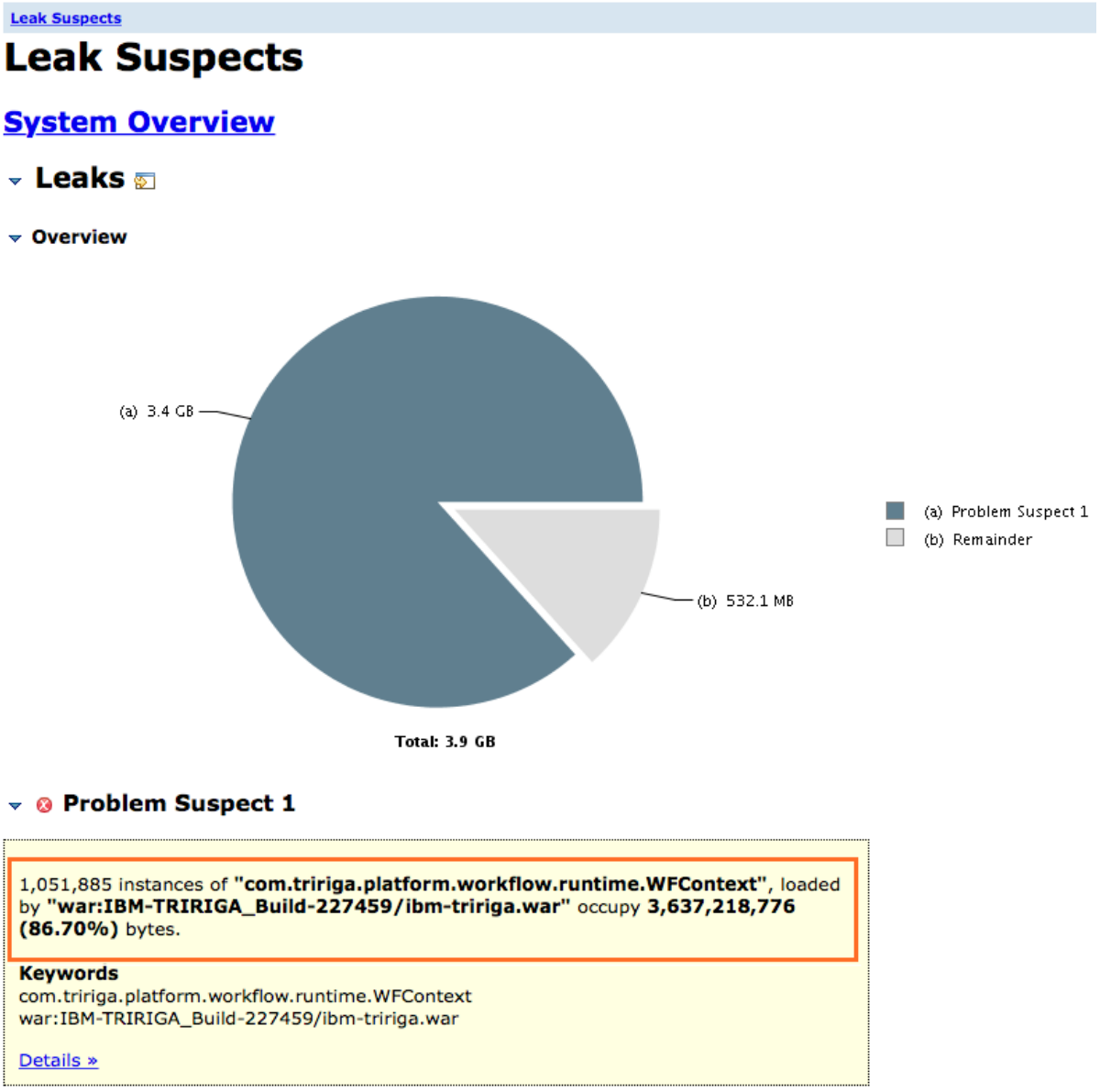

Figure 11. Memory analyzer workflow instance `Out of Memory` heap

The memory analyzer BIRT report `Out of Memory` heap is as follows:

```
One instance of "org.eclipse.birt.report.model.core.DesignSession", loaded
by "<system class loader>" occupies 8,471,008,096 (89.43%) bytes. The memory is accumulated in
one instance of
"Java.Lang.Object[]" loaded by "<system class loader>".
```

Figure 12. Memory analyzer BIRT report `Out of Memory` heap

Problem determination tools

Use the following tools to determine any performance problems with your IBM TRIRIGA Application Platform.

Application platform tools

Use the following tools to analyze system performance, workflow performance, and other areas of the platform:

IBM TRIRIGA Administrator Console
Provides a collection of administrative tools to analyze and optimize system performance.

Workflow Analysis Utility (WAU)
Analyzes workflow performance and process execution. The utility reads IBM TRIRIGA performance logs and displays performance analytics for workflows, including workflow execution time, and process flow, which is the order in which workflows run and what triggered them to run. The Workflow Analysis Utility is available as a separate utility from the product.

Performance Analyzer

Analyzes issues in system performance. The Performance Analyzer provides a more streamlined approach to troubleshooting performance issues than the traditional IBM TRIRIGA performance log analysis. The Performance Analyzer helps you to better isolate and analyze the causes of performance issues by generating a log that is more targeted for the problem area.

Access the Performance Analyzer in IBM TRIRIGA under **Tools**. You can generate a log from the Performance Analyzer that is more targeted at the problem area. When you select the performance timings that you want and start a performance run, the analyzer writes to the performance log while you do the process or actions that cause the slowness.

When you stop the performance run, the analyzer automatically loads and analyzes the performance data to show the most time consuming events of each category type in the performance run. A result summary of the performance run shows the top events that took the longest time to run. You can further analyze the data by reviewing the result details. You can also upload an existing performance log to the analyzer and the analyzer does the same analysis of the data.

Heap dump, thread dump, and garbage collection utilities

Use the following tools to debug Java code:

IBM Thread and Monitor Dump Analyzer for Java

Analyzes javacore files and diagnoses monitor locks and thread activities to identify the root cause of hangs, deadlocks, and resource contention. IBM Thread and Monitor Dump Analyzer for Java also monitors bottlenecks.

IBM Pattern Modeling and Analysis Tool for Java Garbage Collector

Parses verbose GC trace, analyzes Java heap usage, and provides key configurations based on pattern modeling of Java heap usage.

Eclipse Memory Analyzer Tool (MAT)

Finds memory leaks and reduces memory consumption. If your heap dump is larger than 6 GB, the MAT consumes a large amount of memory. Your workstation needs at least 16 GB of RAM. Close all other applications and configure the Eclipse `config.ini` file to set its own max heap size to 15 GB by using **`-Xmx15G`**.

Application profiling utilities

Use the following tools to profile and debug Java code:

Health Center

A GUI-based diagnostics tool for monitoring the status of a running Java virtual machine (JVM).

YourKit

A CPU and memory Java Profiler that supports J2EE/J2ME.

OProfile

A system-wide profiler for Linux systems that can profile any running code with low overhead.

Database utilities

Each of the database platforms contains tools to analyze database health and SQL queries to assist with any long-running SQL statements. For more information, see your database documentation. Some of the more useful tools include the following:

- Oracle Automatic Workload Repository (AWR) snapshots
- Oracle Automatic Database Diagnostic Monitor (ADDM) analysis reports
- IBM Data Studio
- IBM Db2 monitors and snapshots
- IBM Db2 Design advisor db2advis and execution plans

Monitoring the system

The IBM TRIRIGA Application Platform has platform logging features that monitor system health statistics. These features and their associated data help you troubleshoot and monitor performance problems.

The ongoing monitoring of your system can prevent performance issues from arising. Put a monitoring strategy in place before you put TRIRIGA into production.

Monitoring tools

Middleware monitoring tools

Use the following tools to monitor the middleware components that are associated with TRIRIGA:

- Tivoli Performance Viewer (TPV): Monitors the health of the application server from within the administrative console.
- Database Monitoring: Each database platform contains tools that can be used to monitor the database and to provide useful information.

System resource monitoring tools

The following tools monitor system resources during testing:

PerfMon
Gathers performance metrics on Windows-based systems. It provides access to all of the Windows performance counters.

nmon
Gathers performance statistics on AIX-based or Linux-based systems.

- **nmon** is included with AIX starting in versions 5.3 TL09, AIX 6.1 TL02, and Virtual I/O Server (VIOS) 2.1 and is installed by default. **nmon** for older versions of AIX can be found on the nmon for AIX wiki.
- **nmon** for Linux is released to open source and is available on the nmon for Linux wiki.

rstatd
Gathers performance metrics from the system kernel. **rpc.rstatd** is shipped with AIX and can be downloaded for Linux platforms from the rstatd 4 Linux site.

sysstat
A package of utilities that contains the following commands for AIX and Linux.

- **sar** provides system information that is related to I/O, memory, paging, processes, network, and CPU utilization.
- **sadf** formats data that is collected by the **sar** command.
- **iostat** gives CPU and I/O data disks and TTY devices.
- **mpstat** reports global and per-processor data.
- **pidstat** reports process data.

vmstat
Reports statistics about kernel threads, virtual memory, disks, traps, and CPU activity on UNIX systems.

Bandwidth monitoring tools

Use the following tools to monitor network and HTTP bandwidth during testing:

Wireshark
A network protocol analyzer that captures network traffic for bandwidth analysis.

WinPcap
A packet capture and filtering engine that is used by many network protocol analyzers, including Wireshark.

HttpWatch
An in-browser HTTP sniffer that logs HTTP and HTTPS traffic and allows you to inspect the traffic. Use HttpWatch for bandwidth analysis to understand the requests and responses between a browser client and a web server.

Fiddler
A web debugging proxy that logs all HTTP and HTTPS traffic between your computer and the internet. It allows you to inspect the traffic, set breakpoints, and manipulate incoming or outgoing data.

Gathering information for support

Use the MustGather Tool in the IBM TRIRIGA Administrator Console to gather information on your application server, web server, database, and TRIRIGA implementation.

Review the application stack

Review each tier of the stack as a potential source of the performance bottleneck:

- TRIRIGA
- Software services
- Operating systems
- Hardware servers
- Network architecture

For more information, see the infrastructure pyramid in Chapter 1, “Planning for optimized performance,” on page 1. Identify the segments of the pyramid where you isolated the performance issue. Also see Gathering troubleshooting data.

System architecture

Do you have a basic, typical, or advanced system configuration? Review the relevant sections in Chapter 3, “Tuning system architecture and hardware,” on page 7.

Application server

Record the following application server parameters:

- All application server and process server initial and maximum JVM heap sizes.
- Any additional JVM arguments that are added.
- JDBC minimum and maximum connection pool values.

If you are using IBM Db2, confirm that the REOPT(ONCE) bind option and isolation level are set at the application-server level. For more information, see Tuning the IBM Db2 server.

Web server and load balancing

Confirm the following details:

- If you are using IBM HTTP Server, confirm that you followed the tuning recommendations. For more information, see Tuning the web server.
- Confirm if HTTP compression is used.
- If you are using a load balancer, confirm that you reviewed the load balancing recommendations. For more information, see Tuning the application server.

Database server

Confirm the following details:

- If you are using Oracle, confirm the values for **CURSOR_SHARING**, **NLS_LENGTH_SEMANTICS**, **PGA**, and any other settings. For more information, see "Oracle database" on page 62.
- If you are using IBM Db2, confirm the values of all registry, database manager, and configuration settings. For more information, see Tuning the IBM Db2 server.
- If you are using IBM Db2, confirm the use of automatic buffer pool size and auto extends. For more information, see "Tuning the IBM Db2 server" on page 21.
- Use standard database tools to check which database monitoring is enabled. Also check which database indexes were added or customized based on the best practices.
- Specify the table spaces that are used and whether you added a third table space for binary large objects (BLOBs).
- Check that statistics are updated regularly.
- Did you perform any reorganization of tables or indexes? For more information, see 5 Database Server Tuning and Maintenance.
- Does the database use multibyte character sets (MBCS)?
- Is there an archiving strategy in place and how frequently is archiving performed?

Network

Confirm the following details:

- Confirm if hardware compression is used.
- List the results of a network speed throughput test on a sample client machine that exhibits slow performance. For more information, see Tuning the Network.

TRIRIGA

Confirm or provide the following details:

- Provide a list of values for every property. For more information, see "System properties" on page 77.
- Provide a list of every running agent on each TRIRIGA server that is indicated in your system architecture.
- Confirm the thread count values of all multithreaded agents. For more information, see "System properties" on page 77.
- Confirm all Workflow Agent settings and properties such as **WFAgentMaxThreads** and **WF_AGENT_MAX_ACTIVE_PER_USER**. For more information, see "System properties" on page 77.
- Confirm that the Platform Maintenance Scheduler is running. Note the activities that it runs and the value of the **CLEAN_HOUR** property.
- If the performance issue is determined to be at the TRIRIGA tier, run and analyze performance logging. Determine the primary TRIRIGA components that are running slowly and why. If the slow-running components are customized, identify these customizations in the logs.

Evaluate the performance of optimized components based on the following items:

- Confirm that any slow-running reports that are identified in the performance logging are optimized. For more information, see "Query and report performance" on page 87. If the report is still slow, include information from the database about the query execution plan for any SQL running during the report execution. For example, an AWR report or execution plan XML file.
- Confirm that fields and extended formulas that are identified in the performance logging are optimized or are required for the business process. For more information, see "Query and report performance" on page 87.
- Confirm that any Business Intelligence Reporting Tool (BIRT) reports are offloaded to a separate BIRT process server. Confirm that custom BIRT reports use the appropriate filtering to limit the result set.
- Confirm that workflows that are identified in the performance logging are optimized and trigger an extended formula in areas, such as Query Task versus Retrieve Records Task, only when needed. For

more information, see “Workflow performance” on page 93. Also, did you run the Workflow Analysis Utility, and what are the results?

- Detail any integrations that run during slow performance periods. Identify which APIs are used, what they do, and when they run. Confirm that you followed the tuning recommendations. For more information, see “Integration framework” on page 95.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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 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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work
must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., 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.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at https://www.ibm.com/privacy/details/us/en/ in the section entitled “Cookies, Web Beacons and Other Technologies.”

Part Number:

(1P) P/N: