Relativity®

Infrastructure Solutions for Very Large Workspaces

May 20, 2024 | Version 12.3.805.2

For the most recent version of this document, visit our **documentation website**.

Table of Contents

1 Overview

This guide explains the best approach for identifying and fixing problematic Relativity workspaces. Typically, these workspaces are very large and likely exhibit symptoms due to one or more of the following conditions:

1. **Number of users** – the number of users relates to the total concurrent connections to the database. Experience has shown that scalability issues begin when a client scales up to 200 users. This often occurs very suddenly.
2. **Document count** – a database is considered “large” if it has more than 1 million records and a server that hasn't scaled to accommodate it. Very large workspaces may contain more than 15 million records.
3. **Database size** – the total size of all of the databases (tables and indexes) on a single server that may be accessed simultaneously. This may cause excessive calls to disk and result in poor performance. Typically, this begins when this size exceeds 150 GB over the total amount of RAM available to SQL, and the disk subsystem can't keep up.

This guide provides scalability best practices for each of these three conditions. These guidelines can help infrastructure managers understand how to scale environments for very large Relativity workspaces (VLRWs).

Note: "Successful" environments have average long running simple queries of less than two seconds.

For the purposes of this guide, a database can be considered a VLRW when it begins to warrant the support staff’s attention and put pressure on the system it’s occupying. Measures must be taken to alleviate the pressure.

The scaling approach should balance the activity in a database against its size, number of users, and underlying environment. To help make the recommendations useful, we VLRWs by size and by the symptoms its host exhibits. Symptoms are caused by data size, the number of records, and the number of users.

In general, you should be most concerned with performance when you observe the following:

- The amount of data that SQL tries to read into memory approaches an amount that results in long running queries and complaints from users.
- Due to heavy requests, file and disk latency exceeds standards.
- User complaints increase.
- Query durations increase.

There are several factors that you must consider when determining the minimum system requirements needed to support an environment that may reach many terabytes in size. For more information about the systematic causes of infrastructure pressures, see Overview above.

2 Identifying the need to scale

Note that these criteria only apply after all best practices have been verified as executed, and any other potential source of trouble has been eliminated.

2.1 Preliminary considerations

Any database can be overwhelmed by excessive queries, index builds, overly complex searches, and too many users. The following issues may bloat a database unnecessarily or cause poor responsiveness:

- Audit record growth
- Schema changes, such as adding and removing fields
- Slowness caused by too many fields added to the full-text catalog
- Index builds causing considerable drag
- Excessively complex searches
- Large number of folders
- Unauthorized actions or poorly controlled actions on a database server

You should resolve these issues prior to purchasing additional equipment. Ultimately, user demand-driven scaling only temporarily appeases users' needs, which scale in tandem to your efforts.

2.1.1 Are you at capacity?

Many SANs and servers may possess untapped CPU sockets, RAM buses, and internal disk configurations. Prior to purchasing new equipment, determine whether or not the existing equipment meets requirements.

For example, SQL Server connected to a SAN via only four iSCSI ports when there are 16 available is perhaps only operating at 25 percent of its potential. However, keep in mind that as systems become busier, their ability to ingest additional load decreases at an inverse rate.

This guide provides performance solutions for each of the relevant hardware arenas, including network, storage, RAM, and CPU. Solutions are addressed in terms of users, record counts, and database size. For example, assume you're operating at capacity, there's no more room for additional hardware, and network traffic (inter-server communications) attributed to Relativity is necessary. If the network card on a webserver is running at 1 Gbps and it only has a 1 Gbps card, consider scaling. This may mean adding additional web servers, moving to a virtualized web farm with fast internal drive speeds, or installing a 10 Gbps-capable SAN. Your approach depends on observed bottlenecks and what you're trying to achieve.

No one condition creates a VLRW. Rather, several factors converge create a critical mass event in your infrastructure. You might not even realize that you have a VLRW until you begin to experience problems. In fact, a single document table that contains 1,000 records may require the same level of attention as a database that houses 10 million or 100 million rows.

3 Scaling Relativity

In a small workspace, data exists in discrete, known quantities. Basically, the data "fits" in the environment in terms of disk I/O and memory use. It is then balanced against raw processing power and network transfer speeds to create the perception of a "fast database."

In such an environment, storage, RAM, and processing power come together well and are appropriate to the case. At the center of this "fast" environment are two main points of concern:

- The largest Document table
- The largest auditRecord table

Much of this guide has focused on how to handle the various types of load placed on these objects. While other objects may require equal consideration, these two objects will nearly always exceed any other object in size and use. These two tables are the two objects to which the entire environment must be tailored.

To scale Relativity efficiently, we must account for the purpose behind each of these large objects. The Document table serves as the repository for all document metadata for one workspace, as well as some types of coding fields. Other coding data and object data exist in other tables, such as the zCodeArtifact_###### and File tables. The auditRecord_primaryPartition table serves as a case-level repository for all activity and changes that occur to the workspace.

If either reaches a certain tipping point, then you must adjust the infrastructure and not the product. The primary bottleneck on a single machine is almost always throughput from SQL to the disk or SAN. Whatever SQL can't store in memory, it needs to grab from storage. A workspace that contains a document table and/or an audit record of a size where many queries begin to exceed the tipping point will require significant scaling of its I/O and RAM.

Note: If you anticipate having to manage especially large cases in Relativity, you should consider enabling your new workspaces for Data Grid™. Data Grid provides a reduction in SQL Server database sizes, easier database maintenance, backup, and upgrades, reduced memory requirements, automatic workspace distribution across available servers, increased visibility into Relativity audit data, and an increase in the natural limit of case sizes. For more information, see Workspaces on the documentation site..

3.1 Tipping point

MS-SQL tries to keep as much of the database in RAM as possible. Information in RAM always represents the most recent change data. People are usually surprised by how much RAM SQL consumes and the massive I/O churn they may experience. The briefest explanation is that the operating system reads data from disk in certain sized chunks, and, at a certain percentage of estimated rows to be returned, SQL decides it's more efficient to read everything from disk in a large "scan" operation, and then perform searching operations against all of the data, at once, in memory. It does this because it decides that it's more efficient to perform one large read, as opposed to doing many short reads.

The question becomes, then, how long will it take to pull x amount of data (GB or TB) from the storage subsystem? Longer than 2 seconds? How many search queries are selective enough to leverage an index rather than performing a scan? Microsoft Fast Track requirements detail these types of scenarios in detail for participating hardware vendors, including Dell and HP. The expense required to deploy the Fast Track recommendations may be prohibitive, but they do provide a list the equipment required to support these types of huge data sets.

Note: We consider any document query that runs longer than 2 seconds to be a long-running query.

To support large SQL operations, offload as much of the I/O to local PCI-E flash or SSD wherever possible. For instance, Fusion-io cards could be used to house the tempDB data files and/or database log files. Note that local tempDB storage is not fully supported with failover clustering in Microsoft SQL Server 2008 R2, but it is fully supported in SQL Server 2012.

3.1.1 SQL tipping point

The SQL tipping point is much lower than most people think, which is why SQL needs so much RAM. Operating systems read data from disk in certain sized chunks, and if SQL decides it needs to read in every sector, it scraps the index and reads it all into RAM. When queries are not "selective" enough to satisfy this requirement, this is when SQL will convert an efficient index seek to a scan. An index scan is bad because it reads in the entire index. Scans can happen for any number of reasons, non-selectivity being just one of them, and certainly if a non-selective query is written with the intent of exploring data, as often is the case in searching work, then such scans cannot be helped. The easiest way to determine if an index scan is occurring due to selectivity is by forcing the query to be more selective, and observe the query plan.

An index scan that occurs when a large date range is selected should convert to an index seek when a single date is selected. If it does not, then there is something else wrong with the index or the query that is beyond the scope of this document. SQL stores the data in a table in a physical way on disk. This data is ordered by and on a clustered index. A clustered index scan should be avoided at all costs. By prohibiting activities that trigger clustered index scans, or by planning for them, large amounts of data can be stored. Many non-clustered indexes can be stored in RAM far more easily than can be a many terabyte Document table, so care should be taken to either purchase enough RAM for large clustered index scans, or to prohibit and engage in a campaign of testing and planning searches and indexing.

Note: It's recommended that you enable new workspaces that will require a lot of RAM for Data Grid. Among the many benefits of using Data Grid is the fact that the scaling strategy involves simply adding more nodes/servers to accommodate large amounts of text/audit instead of adding more RAM to a SQL license. For more information, see Workspaces on the documentation site..

4 Scaling for number of users

When a workspace has a large number of users, you must scale the environment to the large number of concurrent connections to the database. This section reviews important considerations and best practices for scaling workspaces with many users.

4.1 Web servers

While a SQL Server is very sensitive to the size of the data tables it hosts, web servers are very sensitive to the number of users. The primary method of scaling Relativity web servers is to add more web servers.

The web server is the first line of defense against system saturation. A surge in review often causes a need for additional web resources before any other server begins to feel pressure. While web servers are usually the first to show signs of stress due to a large number of users, a large user load could also strain SQL.

4.1.1 Application pools

Relativity enhances scalability through the use of application pools. By creating purpose-driven pools, you can move services that become resource intensive and redistribute them . The three main application pools are listed below, along with the user activities that place load on the server:

- **Relativity.distributed**
  - Large documents or frequent document viewing (offload specific, heavy cases to single servers, or offload to a near-proximity web and file server for geocentric user populations).
  - Many users in a distant location cause high latency
- **RelativityWebAPI**
  - Importing many load files through the Relativity Desktop Client (RDC) concurrently (offload to a dedicated import/export server)
- **Relativity**
  - Large, complex coding layouts (offload with RULB and NLB to create a server farm)
  - Offload to a hardware load balancer that supports sticky sessions.
  - Event handlers
  - Extracted text in large item list views (list view set to 1,000)
  - Large, complex folder views and choice trees

You can move the following functions (by offload) to separate servers. Ultimately, the best approach is to add more web servers and to implement a proper load balancing strategy. Relativity comes with the built-in Relativity User Load Balancer, but you must configure it. If a particular workspace has a latency issue, set up a separate web server to act as a download handler. You can easily configure this at the workspace level.

4.1.2 Download handlers

Typically, infrastructure managers build Relativity web servers into a single, load balanced farm. To offload the strain incurred by a single workspace with many users, it's common to use a different case download handler for just that workspace. You can then dedicate a web server that you can scale vertically.

When under load, Relativity web servers respond well to scaling either up or out. In other words, either increase the RAM and CPU allocation to the existing web servers, or add more web servers. Load balancing is also a great way to maximize computing resources, but you must take care in certain areas.

4.1.3 Load balancing

Load balancing applies to the following categories:

4.1.3.1 Physical

Successfully implement physical devices to physically load balance web servers. It's important that the physical device not interfere with, rewrite, or otherwise alter the actual web calls made by the Relativity.distributed service.

Physical devices must use IP affinity. Users can't migrate to less loaded servers after logging on to the server. Note that Relativity doesn't innately support physical load balancers. While we can provide advice on Relativity and authentication, you should contact the appliance vendor regarding any problems with a physical device.

4.1.3.2 Microsoft NLB

Microsoft Network Load Balancing (NLB) works best when you distribute the user base equally across multiple external networks. Due to Relativity's persistent session requirement, if too many users access Relativity from one network, the load distribution may become lopsided. This means one web server bears too great a load. For this reason, we recommend using Relativity User Load Balancing (RULB) in tandem with NLB.

4.1.3.3 Relativity User Load Balancing (RULB)

Relativity User Load Balancing (RULB) technology is a Relativity-core technology that ensures number of users on any given web server doesn't exceed the number of users on any other web server. Once a user is on the server, they stay there until they've logged off and re-accessed the system. When you implement RULB with SSL, RULB requires a secure certificate for each server in the farm.

4.2 Bandwidth

The amount of bandwidth consumed depends heavily on both the number of reviewers and the number documents being reviewed. Understand the following conditions when assessing bandwidth:

- The average size of a document being reviewed
- The speed of the review
- The number of reviewers

Reviews are slower when reviewers examine documents for longer periods of time. Slow reviews can create a slower connection because the next document downloads in the background. This is true if you enable the Native Viewer Cache Ahead setting for each user.

Sometimes, due to bandwidth considerations and the potential lack of high-speed internet for the reviewers, you can deploy a Citrix/TS server farm. You can virtualize Citrix and TS, but be sure to avoid over-commitment.

Carefully monitor the number of users on a single Citrix server. Too many users on a single machine may quickly drain resources in certain conditions, including the following:

- Very large images
- Heavy redacting

- Very large spreadsheets
- Users opening multiple tabs in a browser window
- Massive persistent highlight sets

You should scale Citrix outward by deploying more servers. When operating Citrix machines in a virtual environment, adding more cores to a single server is viable only if physical cores are available, and commitment thresholds aren't breached. You must also consider the impact of having a single, large, multi-cored machine on the same virtual host as many, fewer cored machines.

4.3 Conversion recommendations

Environments with a large number of users will most likely review a large number of documents. This equates to more conversion request being made to the conversion agents.

4.4 SQL recommendations

A good rule to follow in a highly transactional environment is to have 1 GB of RAM per 1,000 queries per second. Another popular approach is to buy as much RAM as you can afford. In an environment with many users, the number of cores may be the first thing that maxes out. Adding more cores may be the answer, but you also need more RAM.

The following considerations may pertain to SQL environments that experience heavy traffic:

- Use SQL Enterprise Edition
- Network bandwidth availability to the SQL Server
- Disk scalability – readily available SAN LUNs - know your IOPS and your workload profile.
- Test and benchmark your disk IO so you know when you exceed capacity.
- Large document table that experience frequent scans need larger amounts of RAM.

4.5 Workspace administration

4.5.1 Training

Attend our training sessions as necessary to understand Relativity best practices. In addition, conduct frequent internal training sessions to ensure that the case team has a consistent process for adjusting the workspace and workflow.

For a schedule of our training sessions, visit our website at http://relativity.com/relativity/support/training.

4.5.2 Workspaces with many users

In workspaces with many users, ensure that users with access to create views, fields, indexes, and other objects in the workspace are aware of best practices for large workspaces. The following are some of these best practice guidelines:

4.5.2.1 Adding fields

Relativity provides users with the ability to add or delete fields on the document table on-the-fly. This operation momentarily locks the entire table while adding the field.

You can also add and remove fields to the full-text index on demand. This can cause increased loads on the SQL Server, especially if the fields contain a lot of data, or a lot of new records are suddenly added. The following are guidelines for working with custom fields in a large workspace:

- Understand the advantages of using fixed-length text versus long text fields. See the Environment optimization guide.
- Understand your data. Don't make fixed-length text fields longer than necessary. Don't use long text fields when a fixed-length text field works. You can easily correct these mistakes, but they can have a heavy performance impact if the problem is excessive.
- Keep track of the fields you add to the full-text index. If you use keyword search, only index the necessary fields.
- Limit list filter options for fields. Be aware of the filter options of fields in document list views. List filters on columns with many unique values can tax the web server.
- Understand how to change certain field properties, such as enabling Pivot. You should include fields that are used for pivot group by function in full text index.

4.5.2.2 Searching

- Eliminate "is like" queries whenever possible.
- Ensure users understand that conditions are SQL-based queries. You can only search a dtSearch index through the search textbox in the Search Conditions section. See the Searching guide.
- Work with solutions@relativity.com.
- Avoid excessively nested searches (generally three or more).
- Code existing searches to avoid re-running the same search multiple times.

5 Scaling for document count

Multi-million record cases present some unique management challenges. This section reviews the primary areas that need special attention when dealing with workspaces that contain many documents.

5.1 Web servers

Consider the following best practices when scaling your web servers for cases with very large document counts.

Note: Thoroughly test any custom scripts or assemblies to verify that they can handle a workspace of this size.

5.1.1 Web API

When loading documents into a workspace:

- Ensure that you're running in Direct Mode and that the loading server is close to the Relativity farm.
- Use the 64-bit version of the Relativity Desktop Client (RDC).
- Design the workflow so that you can load with pointers.
- Understand that a heavy data load causes heavy SQL load. This can impact the review.
- Consider using workflows that can separate non-indexed data, and run the full-text service in an automated fashion.
- Use one or more dedicated loading servers.
- Verify the connection speed between all servers involved in loading data.
- Ensure that the URL of the webAPI doesn't route through any other devices, such as a firewall.
- Verify that the load runs in Direct Mode.
- Do NOT load ExtractText first.

SQL stores up to the first 8000 characters of all Long text (nvarchar(max)) fields on-row. This means if you load ExtractedText first, any document that contains less than 8000 characters Relativity stores on-row. Any subsequent long text fields you load, may roll off-row and the ExtractedText column, reside in two different locations on disk.

Entries that are less than 8k are on-row, and ones that are greater are off-row. This means that any operations that attempt to sequentially read large amounts of ExtractedText, for things like index builds, become random seeks. Random seeks perform far more poorly than sequential reads. Having a DBA on staff that understands these types of data mechanics is essential to ensuring the smooth operation. This also ensures you achieve efficiency of large sequential reads and writes.

To increase the speed of loading, add additional RDC instances. However, too many RDC instances cause degradation of the load and can ultimately lead to failure. Monitor SQL for blocking and slowdown if necessary.

Due to auditing, it's always faster to append (insert) data than to overlay (update) existing data. Appending doesn't audit the entire record, but overlays create an audit of the data as it existed both before and after the

change. If you perform an overlay, ensure that you first index the overlay identifier in SQL. Typically, someone who has been tasked with managing SQL, such as a DBA, performs this task.

After a large data set is added to a table in the SQL database, the ongoing performance of inserts and updates to the table depend the health of its indexes and the concurrence of its statistics. At certain points, you may need to run maintenance with online index builds while loading data.

5.1.2 Relativity.distributed

An environment with many millions of documents typically has hundreds of reviewers. See Scaling for number of users on page 8 for information on scaling for large numbers of users.

One way to deal with both a large number of users and large documents is to set the case up with its own download handlers. To establish a dedicated domain of web servers for a case, set the download handler to a network load balancer, or other type of single-point access hardware or software load balancer that allows for sticky sessions.

In cases with extremely large documents, requests to the Relativity.distributed service queue, and you can download multiple docs simultaneously. However, when you queue many large documents, they all consume bandwidth, which eventually becomes saturated. For this reason, configuring a single download handler may not be sufficient.

Monitor bandwidth carefully. High latency on an otherwise close web server is a strong indicator that the server is under-powered or that its connection is saturated. You may need more servers.

User behavior can be as important as the number of users. A single user can perform actions that increase the load on the web server more than expected. Scale according to load as well as user count. The following common user actions can substantially increase the load on the web server:

- Many simultaneous windows or unnecessary opening and closing of windows
- Frequent and large printing jobs
- Frequent mass operations
- Frequent deleting

While some of these operations are sometimes necessary, all users should consider how their actions may impact the current load in an environment when they kick off large operations.

5.1.3 Folders

A large numbers of folders may cause memory pressure on the web server. To alleviate this, either increase the RAM or decrease the number of folders. Extremely deep folder structures can also cause very long scan times. Excessively large strings (more than 2,097,152 bytes) cause serialization errors.

5.1.4 Layouts

Very complex layouts with many choices and fields don't perform as well as simpler layouts that target more-specific coding. Complex layouts place a rendering load on the web server and a heavy query load on the SQL Server. Use popup pickers for fields with a large number of choices.

5.2 Conversion recommendations

The Conversion Agent handles most of the actual work for a conversion request. More documents in the environment means more conversion requests made to the conversion agents. Conversion agents scale around 100 users per conversion agent, similar to conversion workers.

There should be no more than two Conversion Complete agents per environment. While only one Conversion Complete agent is necessary, a second may be added to a separate agent server for redundancy.

When documents are converted Relativity saves a copy of the document in the Viewer Cache location. You can warm the cache location by performing a Mass Convert mass operation. This pre-converts a large set of documents into HTML5 format before review. By using mass convert you can eliminate document load time in the viewer.

You can create as many cache locations as you need. For workspaces with a large amount of documents we recommend creating a new location for each workspace.

5.3 SQL recommendations

5.3.1 SQL Enterprise Edition

The Enterprise Edition of SQL offers many features that assist with the operation and performance of an environment that meets or exceeds very large workspace conditions.

5.3.2 Maintenance plans

Maintenance is a critical operation in a busy SQL environment. Failing to run maintenance results in extreme performance declines followed by catastrophic failures of many core activities in Relativity.

5.3.3 Index strategy

It's not uncommon for a single document table in a very large workspace to have several hundred fields. In such cases, you can often improve query performance by adding indexes. However, be careful that the number of indexes doesn't become excessive. Extreme slowness may occur in any workspace that has dozens of indexes. You must update all indexes whenever data changes, including both updates and inserts.

You should plan indexes and put a strategy in place to manage them. This strategy should ensure that you evaluate the usefulness of the indexes and remove unused indexes.

We recommend a robust test environment. SQL Server Management Studio provides a Database Tuning Advisor (DTA) tool that can be very helpful. Environments that are hosting very large workspaces should never run the DTA in production. Run the tool against a copy of the database in a development or test environment.

Finally, don't simply accept the Microsoft Database Engine Tuning Advisor's (DTA) recommendations. You should fully understand the results of the DTA before you apply them. Some recommendations that the DTA makes might not improve performance and may cause a large resource drain as you apply recommendations. Avoid duplicate indexes, and remove old, stale indexes you're no longer using.

Someone who understand the I/O subsystem should review all tuning recommendations for very large databases. They should also understand how different columns in the database are used, and whether or not the index will actually help.

Hire a qualified DBA who understands how to manage a rapidly changing database. The DBA should understand the need for proper index management, performance tuning, backups, and DBCC checks.

5.3.4 Monitoring utilities

There are number of third party software packages that provide SQL monitoring. These applications can provide valuable insight by collecting information on an ongoing basis. They gather information pertaining to

page life expectancy, long running queries, CPU load, RAM use, blocking, and deadlocks. These applications gather and present this information in user-friendly charts and graphs.

Take care to use a reputable tool because such utilities can have a significant impact on the environment. Microsoft SQL Server comes with a very feature-rich set of Database Management Views (DMVs). These are views that are already managed and maintained inside of SQL. In fact, software designed to monitor SQL runs scripts that query these views. A DMV gathers the following useful metrics:

- Latency by data file
- IO by file
- Longest running queries
- Queries by CPU
- Objects in RAM

The Relativity performance testing team uses a tool called Load Runner to benchmark Relativity, but hosts who seek to truly push hardware performance threshold should seek to perform their own tests. Relativity provides no load/stress testing tool that integrates with the application.

6 Scaling for database size

Massive cases require very large databases that can be accessed from a single SQL Server. This section reviews the best practices relating to the SQL Server when workspaces grow to several terabytes.

6.1 Web servers

In some workspaces documents can be larger than average. Large documents may require many coding fields, which increases the size of layouts. When a coding layout becomes excessively large, it places excessive load on the web server. You may need to create more coding layouts and adjust your workflow accordingly.

The size of the document folder tree may also affect performance. Web servers experiencing such a load show large RAM reservations in the Relativity application pool. To alleviate this you can add more RAM, trim the number of folders, or add more web servers.

Large folder counts also increase the time it takes for a workspace to load in the Relativity Desktop Client and in the web browser. To reduce load time, increase the bandwidth available between the load server and the web server as well as to users.

6.2 Audit record management

Depending on the type of review, the following activities may result in the auditRecord table rapidly increasing in size:

- Many overlays of data. This causes the auditRecord table to increase in sizes equivalent to the size of the data being overlaid, plus any XML overhead.
- Propagation. When propagation is active in databases with large families, it causes tremendous increases in audit size.
- Deleting. Deleting is incredibly audit intensive, because Relativity audits all deleted data. Disable snapshot auditing if possible.

The Database Partitioning for Archiving Audit Records Guide provides a way to archive auditRecord rows while keeping the records accessible. In addition, you can disable certain auditing on the object level so that deletes occur more quickly.

Note: You can download the Database Partitioning for Archiving Audit Records Guide from the Relativity Community..

When adjusting any level of auditing, it's important to thoroughly communicate the changes and completely understand the repercussions of adjusting audit levels.

6.3 Conversion recommendations

Large documents take longer to convert and use up more bandwidth as they move between the conversion agent, viewer cache location, and web server. Each conversion agent can accommodate up to 100 concurrent users

6.4 Data Grid

In very large environments the driving factor for large databases are the document audit record tables. The amount of extracted text in the environment drives the document table size.

Use Relativity Data Grid to store, search, and analyze extracted text and audit data at massive scale. The benefits of using the Data Grid data store include Scalability and more efficient review workflows on case sizes with 100 million or more documents and billions of audits; more performant database maintenance, backup, and upgrades; a reduction in SQL server database sizes, leading to better performing SQL databases; and increased visibility into reviewer productivity and system performance.

In very large environments we recommend to use SSDs for Data Grid storage.

As more data moves into Data Grid, the network bandwidth becomes a bottleneck. Use 10 GB Ethernet for the Data Grid server network.

6.5 SQL recommendations

In large documents, large extracted text fields can slow the responsiveness of SQL. Large extracted text fields also affect the amount of time it takes to build dtSearch indexes. This can also affect the amount of time it takes to load and update data through the Relativity Desktop Client.

When the amount of data in a table exceeds the available RAM, consider the following:

- Latency is related to disk I/O speeds. A slow database means slower query performance as data loads from disk.
- You can achieve faster queries with more RAM. Consider installing the maximum amount of RAM.
- You can achieve faster queries with more disk I/O.
- Better disks, such as SSD, also improve or eliminate latency problems.
- Lower latency on tempDBs and log files means better overall performance.
- Well-tuned indexes and statistics allow for faster CRUD.

6.5.1 Common mistakes

SQL Server CPU is precious. Any executable bins on your SQL Server may install components that are always in an on state. These components steal cycles. In addition, you must tightly control user access to SQL. Submit custom queries for DBA review and then schedule.

In addition, avoid these common mistakes when managing a very large workspace:

- Don’t alter instance settings in SQL without knowing exactly what they do. For example, Priority Boost is not a turbo button.
- Don’t neglect to take database backups and perform DBCC checks. When a workspace is very large, backups may fail for various reasons.
- Remember to manage log files in Relativity.
- Don't keep logs, tempDBs, the full-text catalog, and data files (or any combination of them) together on the same drive.

Large amounts of data in a single table can also slow maintenance. This is because you may need to index many SQL indexes to improve search and sorting performance. Inserts, deletes, and updates all become slower when there are many indexes. You must balance the number indexes against database maintenance performance, and apply indexes judiciously. In large environments, employ a DBA is who understands query optimization and stays ahead of need by understanding the project and its drivers.

Finally, we do not recommend or support use of Microsoft's Database Engine Tuning Advisor's (DTA) recommendations . All tuning recommendations for very large databases should be reviewed by someone who understands the I/O subsystem, how to use different columns in the database, and whether or not the index actually helps. DTA recommendations are often completely oblivious to the size of the columns it recommends indexing, and can lead to catastrophic results.

Hire a qualified DBA who understands how to manage a rapidly changing database. The DBA should understand the need for proper index management, performance tuning, backups, and DBCC checks.

6.5.2 Managing indexes

In very large Relativity workspaces, place additional indexes on any fields that require them in the SQL database. This helps to optimize workspace performance for the review team. However, having too many indexes can slow down insert, update, and delete statements. Your team should understand how to analyze the SQL Dynamic Management Views (DMVs) to identify index candidates and those that should be dropped. Note that auto-collected DMV data is flushed with each SQL instance reboot.

To manage your SQL indexes over time:

- Assign responsibility for regular index analysis and maintenance to an internal staff member.
- Set aside some time (weekly or biweekly) to analyze and maintain custom indexes.

6.5.3 Disks

SQL houses the majority of data reviewers search and tag in a large-scale review. The total size of frequently requested data shouldn't be larger than the total RAM available to deliver that data in a reasonable amount of time.

SQL data should live on the fastest available storage, with the tempDB data and Log files on SSDs and at least 400 MB/s concurrent throughput each on the Data, Log, and FT catalog volumes. This means the server should sustain a cumulative of 1600 MB/sec write. High-performance sequential write speed (cache optimized) may be desirable for backups, but it generally doesn't impact Relativity performance.

For workspaces where the document table and its indexes exceed the size of available SQL Server physical RAM by an unacceptable amount, add additional fiber or iSCI connections to maintain performance levels. For example, in the Fast Track requirements, Microsoft recommends 200-400 MB/s throughput per processor core for big data stored in data warehouses.

In an environment intended to store many TB of data, data throughput on the I/O subsystem must meet the demands of the application's capacity to serve the data. If it doesn't meet the demands, users experience a slowdown during review. In cases where disk latency has increased beyond acceptable levels, increase the I/O throughput to the SQL Server to match demand. It's also important to properly manage SAN cache optimization and manage user expectations when necessary.

Relativity SQL Servers with 512 GB to 2 TB RAM are more common as the amount of data being hosted grows. When workspaces get this large, you must either cache most of the workspace in RAM, or build out an array of very fast disks with Fast Track-like quality I/O throughput. Both approaches are expensive.

6.5.4 SQL Latency

SQL Server disk write latency should be 20 ms or less. Reads may be a problem if they exceed 50 ms, with 100 ms being a "panic" number. You should always evaluate latency in tandem with SQL wait stats. Some Relativity clients set their system specifications with the goal of less than 15 ms latency on the data drives, (which should be blocked at 64k) and less than 5 ms on the tempDB and log files. Typical latency on the tempDBs, in a well-tuned environment, average less than 1 ms. SQL Server disk read/write latency for translation Log SAN volumes should be 10 ms or less, according to Microsoft best practices. Sequential writers to the translation log can be as fast as 1 ms.

We recommend following the best practices outlined in the Environment optimization guide. You may also choose to work with your storage vendor to ensure you configure the device is configured for optimal performance. Work with your storage vendor or follow Microsoft best practices to determine where different SQL data should live and how to optimize the cache. Random seeks are best for database data files, while sequential is best for logs. Make these determinations with your vender because different SANs behave differently, and different workspaces and activities may place different demands on the disk subsystem.

Work continuously to ensure that the link between SQL and the storage is configured for optimal performance. Work with your storage vendor to ensure that the multipath I/O (MPIO) is set up correctly and provides optimal performance. This setup becomes much more complicated and less efficient when virtualizing SQL.

Note: Only a Relativity-dedicated team should attempt virtualization of SQL for extremely large workspace. The team should includes both SAN and virtualization experts. Carefully assess the use of blade servers, because many of them have configuration limitations that adversely impact SQL.

Before going live with SQL in any Relativity environment, test the throughput using CrystalDiskMark or SQL Disk IO (free utilities), Iometer, or something similar. If your I/O profile doesn't meet your expectations, performance isn't going to meet end user requirements. For other general best practices, see the Environment optimization guide.

6.5.5 dtSearch

In a distributed environment, where workers are going to be building an index on multiple machines, verify the throughput of your network. Network problems are the biggest bottleneck for dtSearch builds. When scaling your building, more dtSearch agents equal better performance until your environment reaches an I/O bottleneck.

6.5.6 Scaling dtSearch workers and search agents

Each dtSearch index worker that lives on an agent machine uses 1 core and up to 1 GB of RAM. For this reason, you must carefully size the dtSearch index worker box. The total size of the index that dtSearch builds for any given batch (dtSearchindexBatchSize) doesn't vary, and is set by default to 250,000 documents. The total RAM required may vary as the size of the documents vary and if the size of the batch is adjusted upward. Start with at least 1 GB of RAM per core, monitor for excessive paging, and then scale accordingly. There can be many unknowns in a very large workspace, which is why this RAM recommendation may need higher than the standard 1 GB/core.

Note: If you experience faster speeds when you increase something, or vice-versa, don't assume that it's good. Changes you make to the size of the sub-indexes to improve performance in one area may adversely impact another. There is also a point of diminishing returns. It’s a balancing act, and the only way to gain mastery of it is to practice, watch, keep records, and know the throughput and latency figures for your environment.

It’s very important that the data set used in a dtSearch build is real data. The presence of many long strings of random characters hurt search performance, as well as the performance of the build and merge process. For more information on dtSearch index building, see Workflow solutions for very large workspaces.

Searches are multi-threaded, and spawn as many threads as there are sub-indexes or cores—whichever number is lowest is the constraint. For example, if you have four cores and eight sub-indexes, you see four cores performing the search. If you have eight cores and four sub-indexes. you see four active cores. If you have two searches running simultaneously against the same index that has four sub-indexes, and you have eight cores, you see eight active cores.

An instance setting controls the default size of the sub-index. You can adjust this per each index creation through a setting on the index creation configuration screen. The default value is 250,000 documents.

The dtSearch agent is not throttled. If you put one search agent on a server with eight cores, it uses all of them as load demands. Set up a dedicated dtSearch Search Agent Server if you know you have cases that have multiple (more than one or two) sub-indexes that have multiple searches executed against them. You aren't limited to just one search server, either. If you have five, four-core search servers in the resource pool, that’s 20 cores total.

The amount of hardware needed to support the dtSearch architecture is highly subjective. For example, if you set your sub-index size to be 250,000, and none of your indexes have more than 250,000 total records, then you never have a dtSearch that multi-threads. In other words, multiple processes never search a single sub-index. Additionally, cores aren't process bound—for example, while waiting for an I/O response, a single core may go off and do other things. dtSearch is highly I/O bound and the first bottleneck encountered is I/O, and not CPU.

So how do you know when to scale? For the moment, our recommendation is that you understand your existing tier, and then deploy per these recommendations (you can determine your level by referencing the System Requirements.

Data centers host larger workspaces likely have a very robust infrastructure. With enterprise grade I/O, faster CPU, and industrial strength bare metal, single agents to tackling larger batches of documents may be preferable. For workspaces with 15m + documents, a batch size of 500k may be preferable. There is no substitute for experience, so be sure to test with different batch sizes in a quiet environment, and discover where the sweet spot is.

With the dtSearch grid, you can have additional search agent machines, so it may make sense to spread shares, indexing agents, and searching agents out across a wider footprint to improve network latency, as sub-indexes can only live on network shares. Using local drive letters isn't supported.

6.5.7 Other

Files (natives and images) and dtSearch/Analytics indexes can live on slower, tier-3 (SATA) storage. An NAS is often used to serve up the files. Alternatively, you can set up a VM as a file server and connect it to a SAN.

6.5.8 Database maintenance

For even the largest, most active workspaces, it’s important to set aside some time each day for maintenance. Key maintenance tasks include the following:

- Eliminate index fragmentation each night.
- Update statistics each night.
- Set Auto Update Statistics for the database to On.

- Back up the transaction log for the database at least once an hour. This way the database can reuse the file and limited auto-growth.
- Either conduct full backups nightly. Or conduct full backups weekly with nightly differentials.
- Schedule database integrity checks to run weekly.
- Set all maintenance jobs to send email alerts on completion or failure of each task to more than one person.

6.5.9 Audit record partitioning

The audit record table in a workspace continues to grow over time. You can download the Count of rows and data in current database (.zip) script to identify and determine the sizes of the largest tables in a database.

You can reduce the number of audit records stored in a workspace database by manually partitioning it, while still providing users with access to these records through the History views in Relativity. For more information on this process, see the Database Partitioning for Archiving Audit Records Guide.

In addition, there is a procedure in which you can convert a workspace to an auditSpace. This specialized approach requires additional setup and configuration from a typical audit partition.

Partitioning audits to another database reduce maintenance windows (such as backups, index rebuilds, statistics updates) for the large workspace. You can also improve performance by storing the archived audit database on a different SQL instance. Using this approach, the audit database doesn't use up available RAM on the SQL instance housing the workspace data files. However, it slows querying of the archived audit table/database. Some partners archive the audit record table past 30 days for all of their databases to a separate SQL instance.

6.6 Workflow and searching

There are three main methods of searching in Relativity. Each method encounters challenges in environments with very large search sets and document counts.

6.6.1 Keyword search (full-text index)

The Search Conditions search textbox searches all fixed-length and long text fields that have Include in Text Index set to Yes. Marking a field with this property adds the field to the full-text catalog indexes in SQL. This way the Relativity can use the Contains operator to search that field.

Keyword searching is different from other Relativity search methods. This is because the SQL Server, by default, begins to populate the full-text index as soon as it detects new data. In certain circumstances, this consumes an unacceptable level of resources. For example, if you add 30 fields to the index simultaneously, the environment could suffer severe performance degradation. Or, if millions of documents are loaded into an environment that has many fields in the catalog, the resulting build may strain the available resources.

When users import data into one of the indexed fields or add or remove a field from the full-text catalog in a very large workspace, it triggers an automatic operation to update the catalog. If there is a large number of indexed fields in the catalog and many new records (perhaps as many as 1 mm per hour), the integrated full-text service (iFTS) experiences load.

Updating data in indexed fields also triggers an update of the index. In very large workspaces, this can be slow and taxing. To avoid potential document table locking and other performance issues, consider the following:

- Limit the number of fields added to this index to 20 or fewer, if possible.
- Schedule a nightly job to update the full-text catalog.
- Use dtSearch when possible.
- Move the catalog to separate drives.
- Delay the full population until an off-peak time when you create a full-text index. This is especially important if the base table of a full-text index is large.
- Disable automatic track changes of the full text catalog. Instead, schedule a job to re-enable automatic mode after the load completes. As a result of this, changes made to any fields in the catalog don't appear in search results until the following day, unless you manually trigger an update.

Note: For instructions on how to stop automatic population, see the following Microsoft article: http://msdn.microsoft.com/en-us/library/ms142575.aspx

6.6.2 SQL Searching

Relativity offers users the ability to create and execute ad hoc SQL searches. These searches may hit against date columns, email fields, custodian data, and various single and multi-choice fields. Often, these queries can be very complex, poorly indexed, and poorly performing. A combination of sizing the environment properly and ensuring a good SQL indexing strategy,provides a certain level of comfort. This is because indexed searches always perform vastly better than non-indexed ones. Use tools like VARSCAT and Looking Glass to locate and correct issues with saved and running searches. You can view current long running queries (CLRQs) in Looking Glass. The threshold for them is set by default to 10 seconds. You can change this threshold in the Performance Database configuration table. Looking Glass reviews one server at a time. If you wish to view a separate server, you need to install the script on that SQL Server. It depends on VARSCAT to return results. You install VARSCAT in each workspace typically through Procuro.

Additionally, for better performance, using multiple files for heavy use objects alleviates disk I/O pressure. This provides the disk I/O exists to support the additional I/O load.

The CLRQthreshold value in the Instance setting table doesn't exist until the first time you run Looking Glass in an environment.

6.6.3 Analytics

Contact Customer Support to work with our Analytics experts to implement best practices for a large workspace index.

7 Infrastructure checklist for very large workspaces

Use this table as a checklist to understanding the infrastructure of working with very large workspaces.

Item	Done
Understand scaling for number of users. See Scaling for number of users on page 8.	
Ensure you're able to quickly scale web servers. See Web servers on page 8.	
Create separate application pools. See Application pools on page 8.	
Dedicate a web server to the very large workspace. See Download handlers on page 8.	
If needed, set up load balancing. See Load balancing on page 9.	
Create a Terminal server.	
Ensure you're able to scale the SQL Server appropriately. See SQL recommendations on page 10.	
Attend training to understand Relativity best practices. See Training on page 10.	
Follow best practices for creating fields. See Adding fields on page 11.	
Understand scaling for document count. See Scaling for document count on page 12.	
Follow loading best practices.	
Follow best practices for creating folders. See Folders on page 13.	
Follow layout best practices. See Layouts on page 13.	
If possible, license for SQL Enterprise Edition. See SQL recommendations on page 14.	
Ensure maintenance plans are set up correctly. See Maintenance plans on page 14.	
Review the indexes on SQL tables; delete unnecessary ones. See Index strategy on page 14.	
Understand scaling for database size. See Scaling for database size on page 16.	
Understand the tipping point. See Tipping point on page 6.	
Scale RAM and disk IO comparably to database size. See Web servers on page 16.	
Manage the Audit record table. See Audit record management on page 16.	
Follow SQL recommendations for large tables. See SQL recommendations on page 17.	
Ensure you are not making any mistakes. See Common mistakes on page 17.	
Ensure SQL data is living on the fastest disks. See Disks on page 18.	
Ensure disk latency is tested and monitored on servers running Relativity. See SQL Latency on page 19.	
Understand workflow investigation of searching and subsequent load. See Workflow and searching on page 21.	
Follow searching best practices. See Keyword search (full-text index) on page 21.	
Follow Analytics best practices. See Analytics on the previous page.	

Proprietary Rights

This documentation ("**Documentation**") and the software to which it relates ("**Software**") belongs to Relativity ODA LLC and/or Relativity's third party software vendors. Relativity grants written license agreements which contain restrictions. All parties accessing the Documentation or Software must: respect proprietary rights of Relativity and third parties; comply with your organization's license agreement, including but not limited to license restrictions on use, copying, modifications, reverse engineering, and derivative products; and refrain from any misuse or misappropriation of this Documentation or Software in whole or in part. The Software and Documentation is protected by the **Copyright Act of 1976**, as amended, and the Software code is protected by the **Illinois Trade Secrets Act**. Violations can involve substantial civil liabilities, exemplary damages, and criminal penalties, including fines and possible imprisonment.