Expert Trail: Cache

Revision 20220308

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 LOOKOUT

Expert trails guide Denodo users through all the relevant materials related to a specific topic, including official doc, KB articles, training, Professional Services offering, and more. The main goal is to give users a single place with references to all the information that they need to become a Denodo expert on any specific topic.

“Why do I need caching?” I thought Data virtualization is a real time integration but caching is replicating data. These are some of the questions that might come up with respect to cache.
In this Expert Trail, you will gain a knowledge of caching with respect to how to model your cache with the necessary best practices in mind, the different purposes of cache and advanced configurations of caching.

2 THE HIKE

Stage 1: Purpose of Caching

Denodo incorporates a cache module to store local copies of the source data as required. It is important to understand the difference between cache and other forms of replication like ETL. Some of the cache purposes are:

- **For Performance**: When you integrate various data sources, there might be a situation where some of the data sources are slower than the others (for e.g. REST API) . In these situations, you can cache data from the slow data sources and use the cached data in response to any queries against that data source.
- **To optimize frequently used queries**: When there is a pattern of queries with a high frequency of users calling for the same data, these queries can be cached. The real-time needs of such queries must be analyzed to determine the time-to-live in the cache.
- **To minimize source system impact**: What happens if anyone and everyone starts querying my operational systems in real-time? What will be the performance impact on my operational users who depend on these systems? The Denodo Platform can serve a real-time view of data to certain priority users and partially cache data to others to minimize source impact, while meeting differentiated user needs.
- **To protect against intermittent system availability**: Consider a scenario where a data source in a regional sales office might only be available during local office hours. Caching data from these sources can help mitigate against the actual source data not being available.

Let's continue this path by checking the Best Practices and recommendations for different aspects of the Cache Module such as, how to choose the cache database, how to decide what views to cache, or what is the best cache mode and refresh strategy for each particular use case:
Best Practices to Maximize Performance III: Caching

For information in other performance optimization techniques, review the Expert Trail: Query Performance Optimization

Stage 2: Cache Configuration

Initial setup:

To make use of the cache, the first and foremost step is to configure it. The document Configuring the Cache explains how to enable the cache engine in the Server.
After this, the cache can be configured for a specific view that you want its data to be cached. The document Configuring the Cache of a View explains how to do this

Loading/refreshing the cache:

Cached data is liable to become out of date. Therefore, you may need to establish a data refresh plan periodically. Our recommendation is to use Denodo Scheduler to automate the cache load process, may it be an incremental or complete load of the cache.

- For Partial cache, you only need to configure cache refresh processes when using the 'explicit' option.
- For Full cache, you always have to fill in the cache with explicit loads. After that, you need to refresh the data periodically.
- For loading Full cache incrementally,
  - You can configure the Cache load process (introduced in Denodo 8.0 Update 20210715) to reference the time of the last execution when doing a incremental cache load and allows to have "maching_pk" for invalidation, with this the cache engine uses the values of the primary key to identify which rows of the cache table have to be inserted and which ones have to be updated. For example, the below syntax is supported:
    - `SELECT * FROM view WHERE <fieldName> > @LAST_REFRESH_DATE CONTEXT( 'cache_preload'='true','cache_wait_for_load'='true', 'cache_invalidate' = 'matching_pk')`
  - The Denodo Incremental Cache Load Stored Procedure can also be used. This component can be used to retrieve from the data source the new/updated rows since the last time the cache was refreshed, and to merge them with the cached content.

Cache Maintenance:
When the content of a cached view is invalidated or reaches its time to live, expired/invalidated rows will not appear in query results. Denodo provides a cache maintenance process that periodically removes these rows. But, in production environments, the recommended setting is to use the CLEAN_CACHE_DATABASE predefined stored procedure instead.

Stage 3: Modeling

Depending upon your scenario selecting the right cache mode plays a vital role.

- **When to use full cache:**
  - For example, If you want to delegate complex operations like joins, unions, group by, etc involving several views from different data sources to the cache database. In this way, the performance of these operations is significantly improved.
  - **When to use Incremental Cache mode:**
    - If you want to merge the cache data with the delta records (i.e) records inserted into source after cache is preloaded then incremental cache mode is a good fit.

- **When to use Partial cache mode:**
  - This mode does not force you to have all the tuples of the view in the cache. For example, a REST API with input parameters is a perfect fit. The document Partial Mode provides the details for configuring this mode.

- **When to select the 'Explicit Loads' Option:**
  - This option is a good fit when a relatively small subset of the data is queried much more frequently than the rest and, in addition, it is easy to predict which subset is the most frequently queried. The document Explicit Loads provides the details for configuring this mode.
- **When to select the 'Match Exact Queries Only' Option:**
  - This option should be used when the data source does not return all the results for a certain query. For instance, many websites and web services return only the top 'n' results for queries which have too many results. In such cases, unchecking this option could return incomplete results for certain queries. The document Match Exact Queries Only provides the details for configuring this mode.

Stage 4: Advanced Configuration

With a better understanding of what cache can do, now it is time now to take a look at some additional configurations or considerations.

Bulk Data Load: Some databases also provide bulk load API options to optimize loading big amounts of data, unlike the default 'batch insert' capabilities. This may also imply limitations on the atomicity of the cache loading operation and Caching Very Large Data Sets documentation provides several recommendations for it.

Cache Indexes: Indexes improve the speed of data retrieval operations on database tables. Thus, defining cache indexes for a view can take advantage of the indexes benefits.

Specific conditions: There are some specific conditions you have to take into account depending on the database you use to store cached data. The document Specific Information about Cache Databases explains in detail about this.

Estimate size of cache database:
There are several factors involved in estimating the space required for the cache, and they can be highly dependent on the characteristics of the selected database system. The document Cache database size estimate explains more in detail.

Stage 5: Summaries

Apart from the traditional cache, Denodo 8 includes smart query acceleration techniques using a new type of view called Summaries or Smart Caching. These store common intermediate results that the query optimizer can then use as a starting point to accelerate analytical queries.

Summaries do have some advantages over the traditional cache:

- Users need to explicitly reference the cached view to use the precomputed data.

Whereas, summaries are transparently used by the query optimizer, analyzes and rewrites the incoming queries to take advantage of the data in the summary whenever possible.

- Summaries are not restricted to the cache data source. Meaning, when you create a summary you can choose the best data source alternative for that particular data, and also store the same data in multiple data sources so the optimizer can choose the best option for each case.

However, when it comes to **caching** it does not require a preprocessing phase to decide whether to use it or not. So in cases where you know you always want to access a certain data in the same way as you usually do, it is better to continue using tradicional cache, to avoid any unnecessary overhead. Also, caching offers more refresh and management options, like implicit caching mode, incremental queries, atomic or non-atomic loads, selective invalidation.

Stage 6: Troubleshooting

Finally, the latest step is to learn to debug and identify the possible reasons when a certain problem arises.

For example, what if the cache server goes down for any reason, in this case the behavior of the VQL queries executed on cached views will depend on the cache mode:
What happens if the cache database goes down?

Monitoring the cache can also help to debug if any issue occurs when load processes of the cache on the Virtual DataPort server is performed. The document Monitoring - Cache shows the different fields that are available in the “Cache Load Processes” table which are useful for debugging. More information about monitoring can be found in the Expert Trail: Monitoring.

3 EXPLORATION

Fill up your backpack with additional gear:

Cache Best Practices

Official Documentation	● Configuring the Cache ● Monitoring - Cache ● Diagnosing - Cache ● Recommended Parameters for Queries that Load the Cache ● Caching the Result of Queries that Fail ● Generic Support for Other Databases
KB Articles	● "This connection is part of a global transaction" error ● ORA-01461: can bind a LONG value only for insert into a LONG column ● Double byte characters not stored correctly in the cache ● Optimizing Row Fetching in Denodo ● Using Apache Derby as cache
Additional Resources	● Agile performance - Tutorial | Denodo Community Site ● Workload Management - Tutorial | Denodo Community Site

Cache related Predefined Stored Procedures

Official Documentation	● Denodo Cloud Cache Load Bypass Stored Procedure ● Denodo Incremental Cache Load Stored Procedure ● CACHE_CONTENT ● GET_CACHE_CONFIGURATION ● GET_CACHE_TABLE ● GET_CACHE_COLUMNS ● COMPACT_HADOOP_CACHE ● CLEAN_CACHE_DATABASE

Smart Query Acceleration (Summaries)

Official Documentation	● Smart Query Acceleration Using Summaries — Virtual DataPort Administration Guide
KB Articles	● Best Practices to Maximize Performance III: Caching: Summaries vs Cache
Webinars	● Accelerate your Queries with Data Virtualization
Additional Resources	● Increase the Performance of Your Logical Data Fabric with Smart Query Acceleration - Post | Data Virtualization Blog

4 GUIDED ROUTES

4.1 DENODO TRAINING COURSES

Denodo training courses provide expert data virtualization training for data professionals, including administrators, architects, and developers.
If you are interested in Cache you should enroll the following course/s:

- **Denodo Data Management**: This course covers when and how Denodo can store data itself (Cache / Materialized tables / Temporary tables / Summaries) and how to manage that data (see the data, remove/invalidate data,...).
- **Advanced Configuration of Denodo Views**: This course covers all the advanced configuration options available in Denodo views including cache configuration.
- **Denodo Performance Best Practices:** This course will talk about the internal details of the Denodo Optimizer to learn how to maximize the performance of the queries executed in Denodo Platform.

4.2 TECHNICAL ADVISORY SESSIONS

Denodo Customers with active subscriptions have access to request Meet a Technical Advisory sessions.
These are the sessions available related to Cache:

Infrastructure implementation	**Cache: Configuration & Administration Best Practices**	- Configure the cache data source/s using a supported database. - Configure Bulk Load for cache. - Set up Cache Maintenance Task. - Monitoring the cache. - Smart Caching: Summaries
Architecture & Standards	**Cache Modes Overview**	Review and showcase how the Denodo Cache works: - The different cache modes (Partial, Full, Incremental) - Loading the cache and refreshing. - Invalidating the cache contents. - Indexes.
	Cache Best Practices	- Assist you in defining a policy to determine when to use the cache and what type to use, or review your current policy. - Advice on selecting the best cache strategy for a specific scenario. - Incremental caching strategies.
	Schedule Cache Load	Assist you on your first steps creating cache jobs in Scheduler.

4.3 PROFESSIONAL SERVICES

Denodo Professional Services can help you at the start or any part of your query performance trail. You can find information about the Denodo Professional Services offering in:
Professional Services for Data Virtualization | Denodo

- **Operations Quick Start**
- **Development Quick Start**
- **Virtualization Architecture & Design**

If you are a Denodo customer, you can reach out to your Customer Success Manager for details about any Guided Route that you need.

5 BIG HIKE PREP CHECK

Let's see if you are ready to start your big trail. Take this 3-question questionnaire to check your readiness for an enjoyable hike.
Read the questions below, think about the solution and check if you got them right by looking at the solution. Have you become an expert?

1. Consider you have a derived (join) view on three base views created over three data sources (1 - JDBC , 2- SAP , 3 - JSON (Rest API)). Looking at the execution trace you have identified the query performance is affected by JSON data source. Where would you apply caching? Is it at JSON base view level or at derived view level?
   Click here to check if you got it right

   > The recommendation in this scenario would be at JSON base view level. From the execution trace you have identified the post processing in Denodo is not a problem but the network latency of JSON affects the query performance.

2. Consider you have a Full cache on a base view and have preloaded with 1 Million rows. The data in the source does not change much, probably a few hundred rows are added every week. So you decided to schedule a cache preload once in a month. Now you have a requirement that the end users also need to view the changes from the data source (i.e those few hundred records). What would be the best approach here?
   Click here to check if you got it right

   > Incremental Full cache mode is a good option for this scenario. For more details, refer to Requirements a View Has to Meet to Support Incremental Queries.

3. In Production, you noticed there is a deadlock error in Denodo Scheduler between a cache preload job and cache maintenance job. What could have caused this deadlock?
   Click here to check if you got it right

   > This may happen when the cache maintenance job is configured to run at the same time of the cache preload job. There are 2 possibilities:
   > - The cache maintenance was set to ON, so Virtual Dataport is running the cache maintenance job automatically and it happens to run at the same time the cache job was scheduled.
   > - Or, the maintenance job (using the CLEAN_CACHE_DATABASE procedure) was scheduled to run at the same time as the cache preload.
   >
   > In Production environments, the recommendation is to set the "Maintenance" option to "Off" to avoid executing the Cache Maintenance task periodically and having control over when the maintenance task is executed.
   >
   > Thus, use the Denodo Scheduler to program the execution of the CLEAN_CACHE_DATABASE procedure at a period where the Server and the cache's database are not expected to be under heavy load, thus avoid having cache load job and CLEAN_CACHE_DATABASE to run in parallel.