Denodo Cloud Cache Load Bypass Stored Procedure - User Manual

Revision 20200629

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

CONTENTS

1 OVERVIEW

Virtual DataPort incorporates a system called cache module that can store a local copy of the data retrieved from the data sources, in a JDBC database. This may reduce the impact of repeated queries hitting the data source and speed up data retrieval, especially with certain types of sources.

The Cache Engine, available in Virtual DataPort, allows two main modes: Partial and Full. When you enable the former, the cache only stores some of the tuples of the view and, at runtime, when a user queries a view with this cache mode, the Server checks if the cache contains the data required to answer the query. If it does not have this data, the Server queries the data source. However, if you use the latter the data of the view is always retrieved from the cache database instead of from the source.

The Denodo Cloud Cache Load Bypass Stored Procedure is focused on giving support to cache Full initialisation processes in scenarios in which Amazon Redshift or Snowflake is used as cache, and the data has to be loaded from another Amazon Redshift or Snowflake instance which is configured as data source.

2 IMPORTING THE STORED PROCEDURE INTO VIRTUAL DATAPORT

2.1 IMPORTING THE EXTENSION TO THE VIRTUAL DATAPORT SERVER

For running the Denodo Cloud Cache Load Bypass Stored Procedure, first of all, you have to load the `denodo-cloud-cache-load-bypass-{vdpversion}-{version}-jar-with-dependencies.jar` file using the Jar Management option of VDP Administration Tool located in the `File > Jar Management` menu. Once you have imported the Jar file you can add the stored procedure.

2.2 ADDING THE STORED PROCEDURE USING VQL LANGUAGE

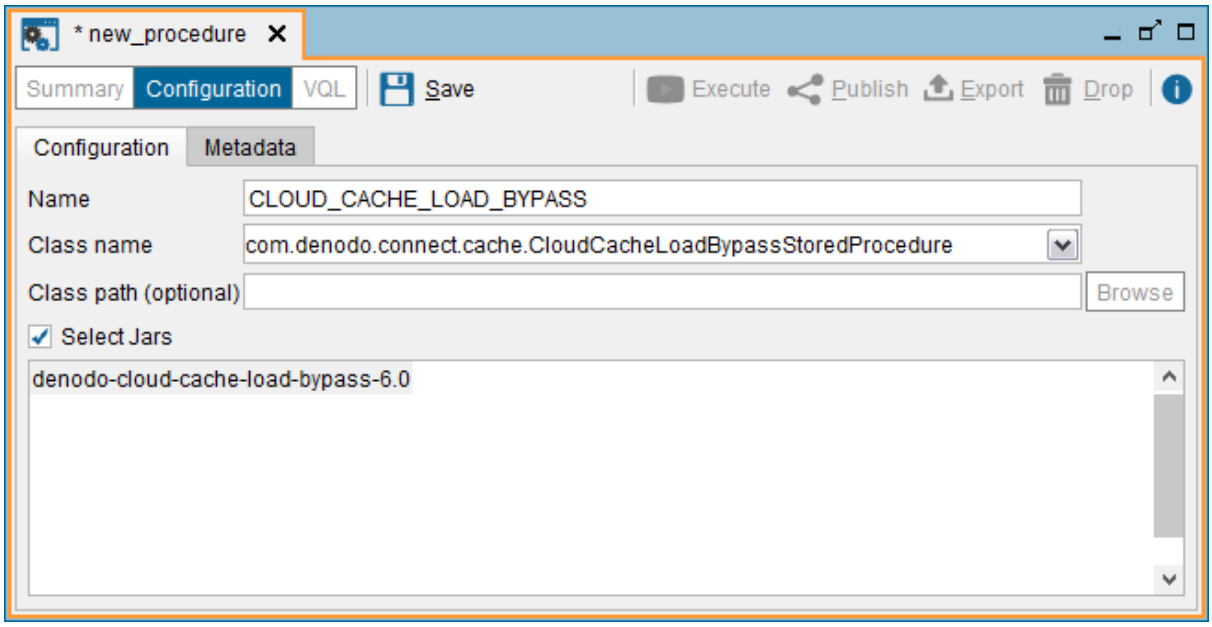
You can add a new stored procedure with the statement `CREATE PROCEDURE`:

```
CREATE [OR REPLACE] PROCEDURE <name:identifier>
CLASSNAME=
 'com.denodo.connect.cache.CloudCacheLoadBypassStoredProcedure'
    JARS 'denodo-cloud-cache-load-bypass-<vdpversion>';
    [ FOLDER = <literal> ]
    [ DESCRIPTION = <literal> ]
```

The classname must be `com.denodo.connect.cache.CloudCacheLoadBypassStoredProcedure` and the JARS value must be the Jar file, `denodo-cloud-cache-load-bypass-<vdpversion>`, previously added to the Virtual DataPort Server (see the **Importing the extension to the Virtual DataPort Server section** for more information).

2.3 ADDING THE STORED PROCEDURE USING THE VIRTUAL DATAPORT ADMINISTRATION TOOL MENU

You can add a new stored procedure clicking `Stored procedure` on the menu `File > New`:



You must set a name in the `Name` field and select the `Select Jars` checkbox in order to use the Jar file, `denodo-cloud-cache-load-bypass-<vdpversion>`, previously added to the Virtual DataPort Server (see the **Importing the extension to the Virtual DataPort Server section** for more information).

3 GENERAL PROCESS AND REQUIREMENTS

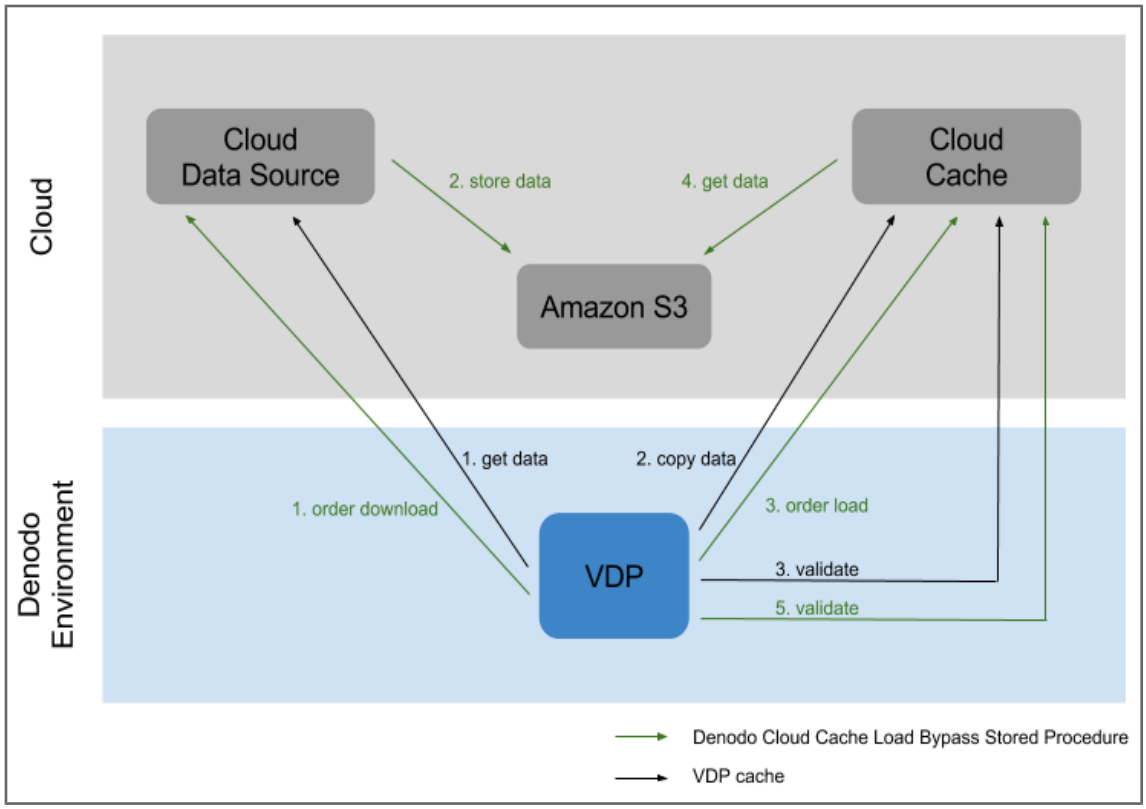
When you enable the Cache Engine in the Virtual DataPort the default DBMS (Database Management System) is the embedded Apache Derby database but it is highly recommended to change it and use an external one. **Snowflake** or **Amazon Redshift** can be configured as external DBMSs in order to use them as cache systems. In addition, both databases can be used as data sources in Virtual DataPort.

If you have this scenario, with Snowflake or Amazon Redshift as data source and Snowflake or Amazon Redshift as a cache, and you use **Full cache mode** in some of your base or derived views, the Denodo Cloud Cache Load Bypass Stored Procedure will allow you to cache the data faster than when you invoke the process with the cache preload query. Nevertheless, when you are caching derived views, you should be aware of:

- Derived views must be fully delegable to the data source in order to be cached using the Denodo Cloud Cache Load Bypass Stored Procedure.
- The Denodo Cloud Cache Load Bypass Stored Procedure does not take into account the cache configuration of the views that have been combined in

order to obtain the target view of the caching. Therefore, although a derived view uses a base view with a cache configured but without valid data, the stored procedure will charge its data from the source.

In order to carry out the data caching, the stored procedure requires **Amazon S3** (Amazon Simple Storage Service) to store the exported data from the data source. Denodo Cloud Cache Load Bypass Stored Procedure will download in parallel the data to cache into a specified Amazon S3 bucket, in GZip files, and the DBMS used as cache will load the data from the Amazon S3, also in parallel. The files will be deleted from Amazon S3 at the end of a successful loading process.



3.1 NULL FIELDS CONSIDERATIONS

During the data unloading and loading process, null values are represented as '\\N' in the Amazon S3.

This decision is based on:

- Snowflake converts SQL NULL values to '\\N' by default; as mentioned in the Snowflake documentation.
- The default null_string is '\N' in Amazon Redshift but it can also be escaped in the input data as '\\N'.

3.2 AMAZON REDSHIFT CONSIDERATIONS

The Amazon S3 bucket where Amazon Redshift will write the unloaded files must be created in the same region as the Amazon Redshift cluster used as data source.

However, if you use Amazon Redshift as cache, the Amazon S3 bucket that holds the data files may be in a different region than the region in which the Amazon Redshift cluster, used as cache, resides.

4 INVOKING THE DENODO CLOUD CACHE LOAD BYPASS STORED PROCEDURE

The stored procedure needs several input parameters:

- **DATABASE_NAME** (nullable text): the Virtual DataPort's database name where you have the view with the data that you are going to cache or the Virtual DataPort's database name where you have configured the data source from which that view gets the data. You can set this parameter as `null` if both `VIEW_NAME` and `DATA_SOURCE_NAME` include their database name. In another case, the `DATABASE_NAME` cannot be `null` and it will be considered as the database of the view and/or the data source that does not contain another one specified in its name.

- **DATA_SOURCE_NAME** (non-nullable text): the database's data source name where you have created the base view or derived view that you are going to cache. You can also specify its database name, in which case you must use the dot notation: `<DATABASE_NAME>.<DATA_SOURCE_NAME>`. Note that this specification is mandatory when the parameter `DATABASE_NAME` is null.

- **VIEW_NAME** (non-nullable text): the view name that the Denodo Cloud Cache Load Bypass Stored Procedure will cache. You can also specify its database name, in which case you must use the dot notation: `<DATABASE_NAME>.<VIEW_NAME>`. Note that this specification is mandatory when the parameter `DATABASE_NAME` is null.

- **QUERY_CONDITIONS** (nullable text): the `WHERE` part of the query to retrieve the data from the source to the cache system.

- **INVALIDATE_BEFORE_LOADING** (non-nullable boolean): a boolean value that indicates whether the existing data in the cache will be invalidated before the current loading.

- **BUCKET_NAME** (non-nullable text): the bucket name where the stored procedure is going to temporarily save the data to cache.

- **AWS_ACCESS_KEY_ID** (nullable text): the AWS access key to sign the requests in order to be able to use the Amazon S3 storage. This value may be null only if you use Snowflake as data source and as a cache because it allows .

- **AWS_SECRET_ACCESS_KEY** (nullable text): the AWS secret access key to sign the requests in order to be able to use the Amazon S3 storage. This value may be null only if you use Snowflake as data source and as a cache.

- **OBJECT_PREFIX** (nullable text): the key name prefix that is going to be used to group data files (folders), objects, in the Amazon S3 bucket. For example, if your bucket name is `my_bucket` and you set `my_cache_files` as object prefix, your files will be created in `my_bucket/my_cache_files`. If it is not

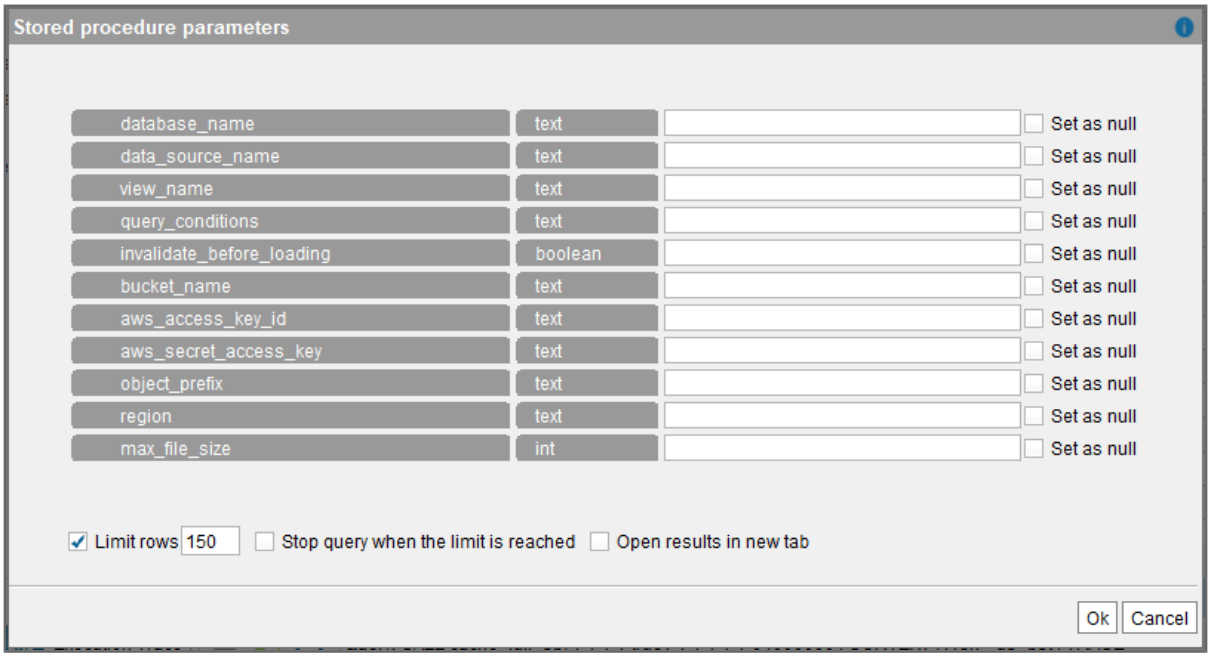
specified, null or empty, the files will be created in the bucket without grouping them. Whether or not this parameter is specified, the files will be created and deleted in the process.

- **REGION** (nullable text): the bucket's region where the source data is located. This parameter only makes sense when the cache engine is configured to use an Amazon Redshift DBMS and in this scenario it is mandatory to specify the region in which the data is placed during the process, in the Amazon S3.

- **MAX_FILE_SIZE** (nullable decimal): the maximum size of each file created in the unload part of the process expressed in megabytes (MB). Using Snowflake data sources, if you do not establish this parameter the default value is 16 MB. Using Amazon Redshift data sources the maximum size for an unloaded file by default is 6.2 GB and although it is configurable the value must be between 5 MB and 6.2 GB.

After installing the stored procedure there are four ways of invoking it:

1. Using the Execute button in the dialog that displays the schema of the stored procedure. The tool will show a dialog where you have to enter the input values.



2. With the `CALL` statement :

```
CALL CLOUD_CACHE_LOAD_BYPASS('database_name',
  'data_source_name', 'view_name', 'id > 3 and id < 20', true,
  'bucket_name', 'aws_access_key_id', 'aws_secret_access_key',
  'amazon_s3_object_prefix', 'region', max_file_size);
```

3. Invoking the procedure on the `FROM` clause of a `SELECT` statement:

```
SELECT * FROM CLOUD_CACHE_LOAD_BYPASS('database_name',
  'data_source_name', 'view_name', 'id > 3 and id < 20', true,
  'bucket_name', 'aws_access_key_id', 'aws_secret_access_key',
  'amazon_s3_object_prefix', 'region', max_file_size);
```

4. Invoking the procedure on the `FROM` clause of a `SELECT` statement but providing the input parameters on the `WHERE` clause:

```
SELECT * FROM CLOUD_CACHE_LOAD_BYPASS()
    WHERE DATABASE_NAME = 'database_name'
    and DATA_SOURCE_NAME = 'data_source_name'
    and VIEW_NAME = 'view_name' and QUERY_CONDITIONS = 'id > 3
    and id < 20' and INVALIDATE_BEFORE_LOADING = true
    and BUCKET_NAME = 'bucket_name'
    and AWS_ACCESS_KEY_ID = 'aws_access_key_id'
    and AWS_SECRET_ACCESS_KEY = 'aws_secret_access_key'
    and OBJECT_PREFIX = 'amazon_s3_object_prefix'
    and REGION= 'region'
    and MAX_FILE_SIZE = max_file_size;
```

The Denodo Cloud Cache Load Bypass Stored Procedure also has an output parameter called `LOADED_ROWS` that indicates the number of rows loaded in the cache system.

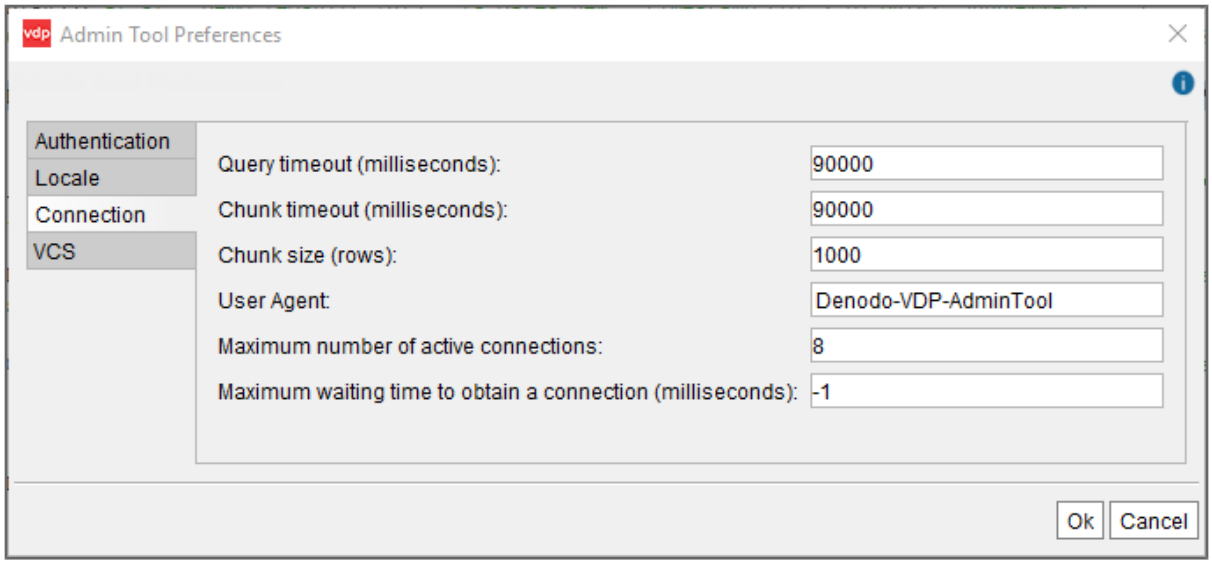
5 LIMITATIONS

Synchronous process

The execution of Denodo Cloud Cache Load Bypass Stored Procedure is synchronous. The stored procedure prevents Virtual DataPort Server from processing the data but it will wait until the end of the process.

Timeout considerations

The process of loading the cache can be quite long, depending on how much data you need to move. If you invoke the Denodo Cloud Cache Load Bypass Stored Procedure using the Administration Tool you must revise the Admin Tool Preferences where you can change the query timeout. If the value 0 is specified, the tool will wait indefinitely until the process ends.


Admin Tool Preferences
Authentication
Locale
Connection
VCS
Query timeout (milliseconds):
90000
Chunk timeout (milliseconds):
90000
Chunk size (rows):
1000
User Agent:
Denodo-VDP-AdminTool
Maximum number of active connections:
8
Maximum waiting time to obtain a connection (milliseconds):
-1
Ok
Cancel