Vertica Knowledge Base Article

Deletes in Vertica: The FAQs

Document Release Date: 2/24/2020

Legal Notices

Warranty

The only warranties for Micro Focus International plc products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus | shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Micro Focus | required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Contents

Deletes in Vertica: The FAQs 4
Document Overview 4
Delete Basics 4
How does Vertica handle deletes and updates? 4
What is the AHM epoch? 4
What is replay delete? 5
Which system table tracks delete vectors? 5
Delete Life Cycle 5
What are the steps in the delete life cycle process? 5
Projection Design Considerations 9
How should I design projections for best delete and replay delete performance? 9
How can I evaluate projections for replay delete performance? 10
How can I avoid recovery from spending hours running replay delete on projections? 10
Can the presence of a large number of deleted records affect query performance? 11
Purging Deleted Records 11
How does Vertica purge deleted records? 11
When and how should I perform a manual purge? 11
What best practices should I follow when deleting large volumes of data from a single table? 12
For More Information 13

Deletes in Vertica: The FAQs

Document Overview

This document answers the most important questions you might have about deletes and how they affect your Vertica database performance. These frequently asked questions are organized into the following categories:

- Delete Basics
- Delete Life Cycle
- Projection Design Considerations
- Purging Deleted Records

Delete Basics

How does Vertica handle deletes and updates?

When you run a DELETE statement in Vertica, the data is not immediately removed from storage containers. Instead, the DELETE statement adds a delete vector that points to each WOS and ROS container that contains records marked for deletion. Each delete vector contains the position of a deleted record in a container and the epoch where the DELETE statement was committed.

An UPDATE statement performs a DELETE followed by an INSERT. When you run a SELECT query on a table, Vertica filters out records marked in delete vectors, omitting those records from the results.

What is the AHM epoch?

The *ancient history mark (AHM)* is the epoch prior to which deleted data can be purged from physical storage. By default, Vertica advances the AHM epoch at an interval of 180 seconds. When a node in a cluster is down, or a database contains unrefreshed projections, the AHM epoch does not advance. The AHM is never greater than the last good epoch (LGE).

You can verify the AHM epoch, the LGE epoch, and the current epoch using the following command:

```
=>SELECT get_ahm_epoch(),get_last_good_epoch(),get_current_epoch();
      get_ahm_epoch | get_last_good_epoch | get_current_epoch
 -------------------+---------------------+-------------------
             492840          492840                492841
(1 row)
```

What is replay delete?

A DELETE statement creates a delete vector that is stored in WOS and labeled DVWOS. An individual DELETE statement creates one DVWOS object for every storage container that contains records that were deleted by the DELETE statement. A DELETE statement with a DIRECT hint creates a delete vector that is stored directly in a file system and labeled DVROS.

When the Tuple Mover moveout operation moves data out of WOS memory into ROS, it combines multiple WOS containers into a single large, sorted ROS container. The Tuple Mover moveout operation also moves out delete vectors from WOS to create new DVROS containers.

If some of the records in the data that was moved out were deleted, then these records have a new position in the newly created ROS container. After the delete vectors are moved out, the delete vectors capture the new positions of deleted records in ROS containers. *Replay delete* is the process of rebuilding delete vectors to adapt to the movement of records across storage containers.

The replay delete operation occurs during the Tuple Mover mergeout operation. When the mergeout operation merges ROS containers that have deleted records, it purges any records that were deleted prior to the AHM epoch. Any records that cannot be purged have a new position in the newly created ROS container. This process requires rebuilding delete vectors that point to the positions of deleted records that were not purged in the new ROS container.

Vertica uses replay deletes in node recovery, reorganization, rebalancing when adding or removing nodes from a cluster, and during projection refresh. These operations also purge deleted data prior to the AHM epoch.

> Important Only deleted records that cannot be purged participate in replay delete.

Which system table tracks delete vectors?

The Vertica system table DELETE_VECTORS contains information about delete vectors and the number of deleted records. For more information, see DELETE_VECTORS in the Vertica documentation.

Delete Life Cycle

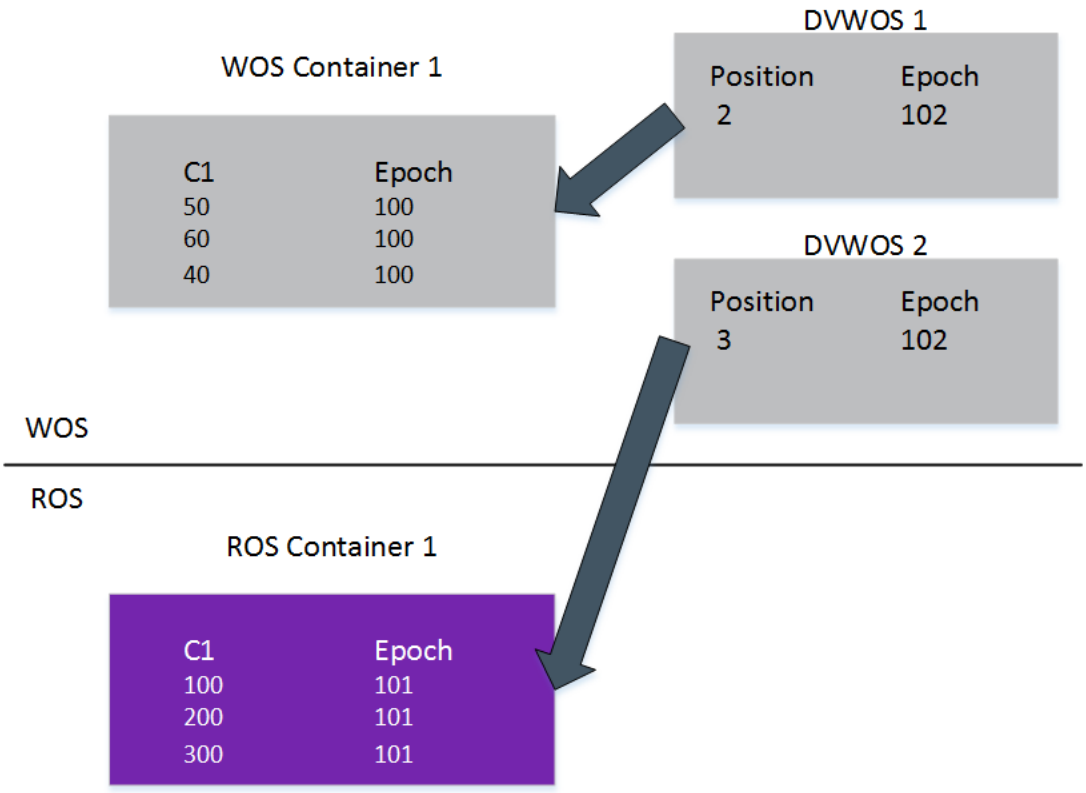
What are the steps in the delete life cycle process?

The following figures and steps show an example of the delete life cycle process. This example shows a table with one WOS container and one ROS container.

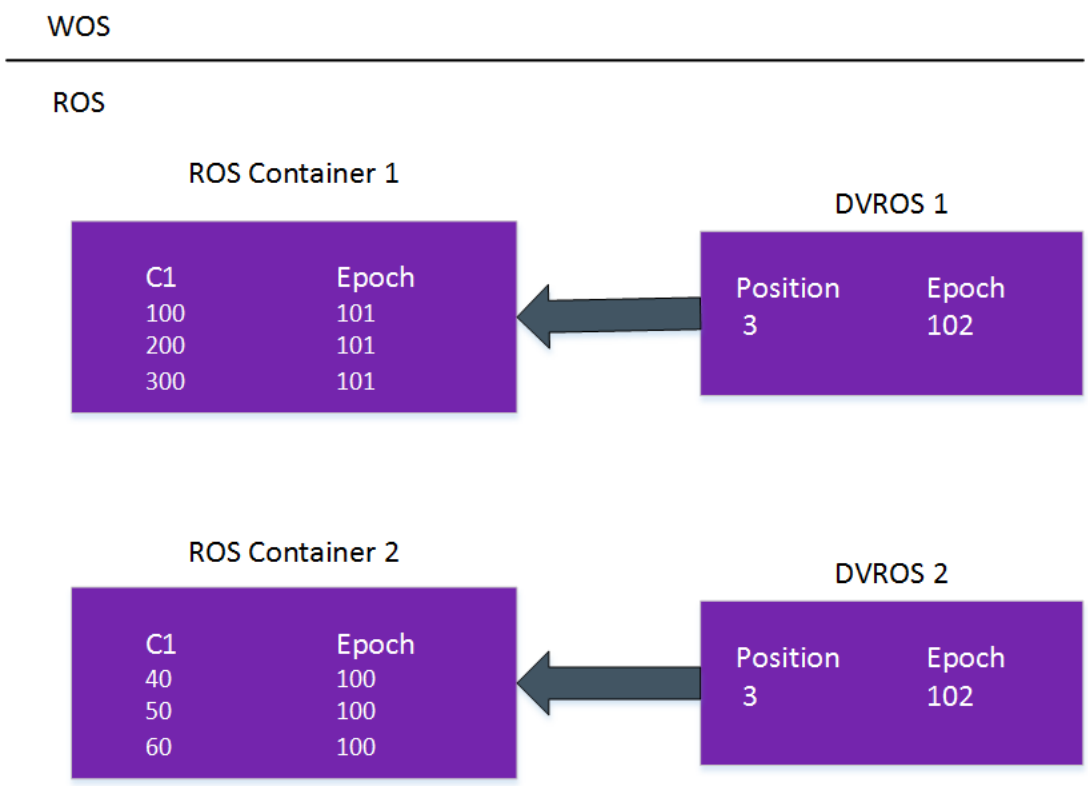
Step 1: Run the following DELETE statement:

```
=>DELETE from Table1 where C1 in (60,300);
```

This statement creates a delete vector (DVWOS) for the WOS container and ROS container. Vertica commits this statement in epoch 102:



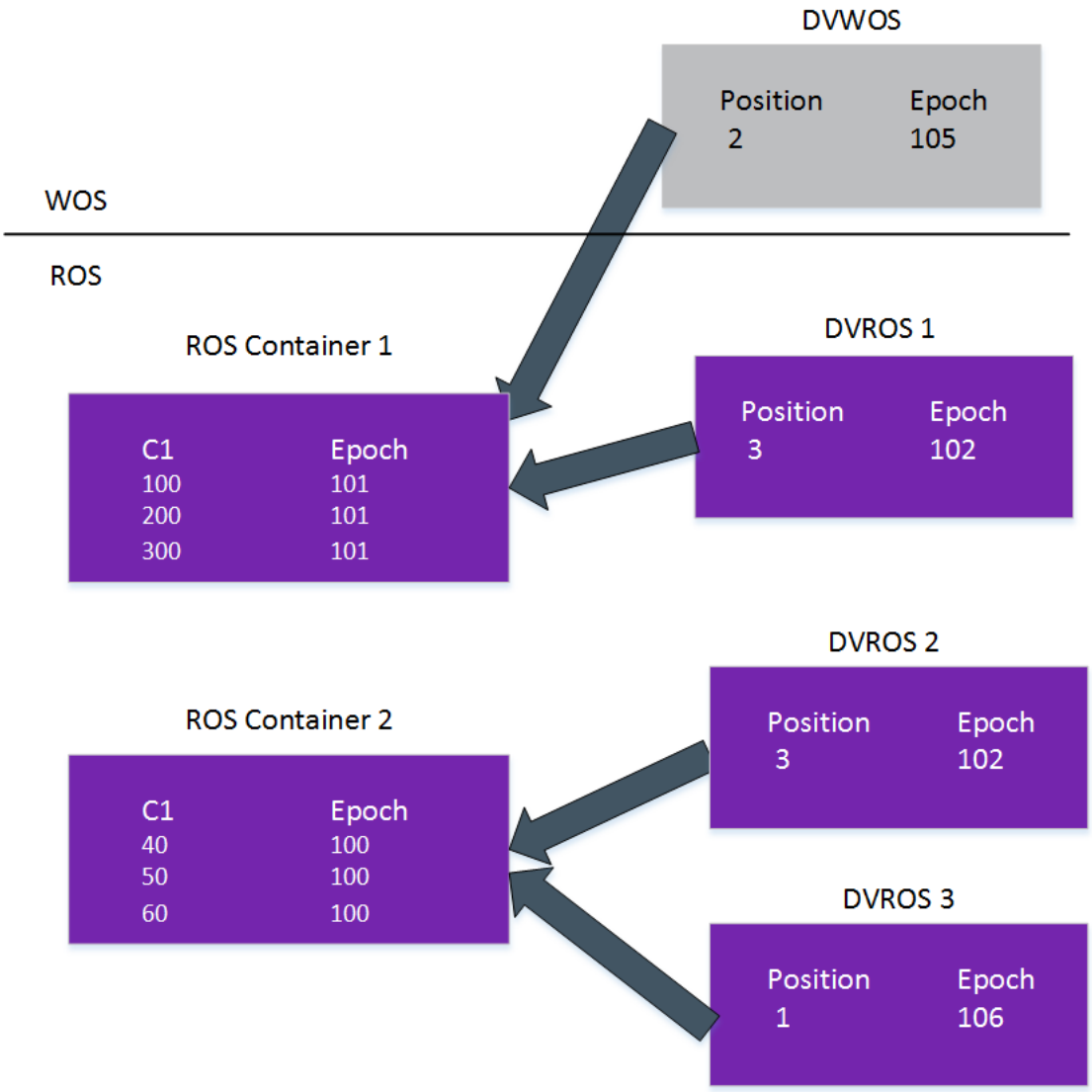
Step 2: The Tuple Mover responds with a moveout operation that moves data from WOS into ROS and creates delete vectors in ROS (DVROS). As the figure shows, the position of the data changes after the moveout operation and DVROS captures new positions:



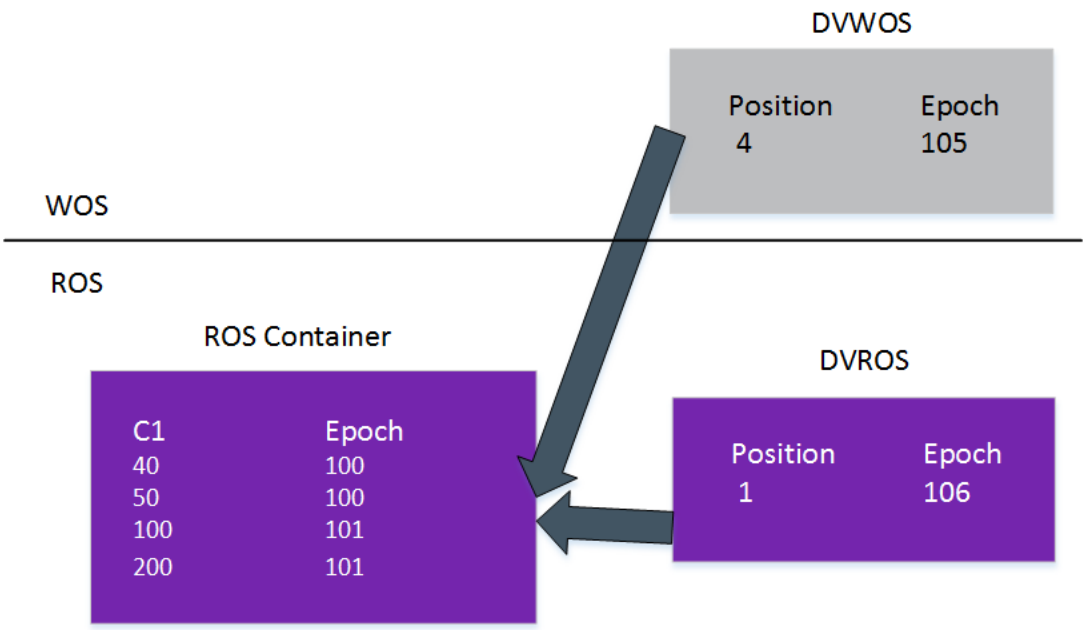
Step 3: The AHM advances to 103. Run the following delete statements:

```
DELETE from Table1 where C1=200; commit;
DELETE /*+direct*/ from Table1 where c1=40; commit;
```

These statements are committed in epoch 105 and epoch 106, respectively. The DELETE statement creates a new DVWOS container, while the DELETE statement that uses direct hint creates a DVROS container.



Step 4: The Tuple Mover responds with a mergeout operation that merges the ROS containers into a single, sorted ROS container. The mergeout operation purges any records deleted prior to the AHM. The following figure shows that the records deleted in epoch 105 and 106 could not be purged. The purge did not occur because the commit epoch of the DELETE statements is greater than the AHM epoch.



Projection Design Considerations

How should I design projections for best delete and replay delete performance?

Consider the following projection design considerations for tables that contain DELETES or UPDATES. If your projection design does not meet these considerations, create new projections, perform a refresh, and drop the old projections.

> Note You can use Database Designer to obtain optimized projections by including DELETE statements in the list of queries passed to Database Designer.

Delete Performance

The performance of a DELETE statement is optimal if the predicate column used in a DELETE statement is present in all projections anchored on the table. Vertica chooses an optimized delete algorithm if your query has a predicate column in every projection anchored on the table. With the optimized delete algorithm, Vertica selects records marked for deletion and the positions of the records for each projection anchored on the table. The algorithm uses the positions to create delete vectors for the respective projections. This algorithm is fast and is the default behavior, unless you have a projection that does not have a predicate column from the DELETE statement.

If you use a non-optimized delete algorithm, Vertica selects records marked for deletion from one projection anchored on a table. The database then searches through all projections to find the positions of the deleted records required to build a delete vector. Using the non-optimized algorithm results in a slower operation than if you used the optimized algorithm.

Replay Delete Performance

For optimal replay delete performance, the column with the highest cardinality should appear at the end of the projection sort order. The replay delete operation requires that Vertica finds the position of the deleted record in the new storage container. Vertica first searches for deleted records using columns in the projection sort order. If there is a high cardinality column in the sort order, Vertica finds the new position more quickly.

In some cases, Vertica cannot identify the exact position of the deleted row using the cardinality columns in the sort order. In these situations, Vertica matches all the projection columns, slowing down the replay delete operation.

How can I evaluate projections for replay delete performance?

Use the following statement to evaluate replay delete performance for a single projection or for all projections anchored on a table.

```
=>SELECT EVALUATE_DELETE_PERFORMANCE ('projection or table name')
```

The output is either:

- No projection delete performance concerns found

  or

- Projection exhibits delete performance concerns, followed by a list of concerns

How can I avoid recovery from spending hours running replay delete on projections?

When a node is down, the AHM epoch does not advance. Deletes that occur when a node is down or when a node is recovering have a higher commit epoch than the AHM epoch. As part of the recovery process, these deletes must be replayed on the recovering node. Issues can occur if the target table is large and projections on the target table were not designed for optimal delete or replay delete performance. In such cases, the projection can get stuck in recovery while performing replay delete.

If you encounter this situation, replace the existing projections with projections that follow the recommendations for optimal projection design. You can do this by adding a high-cardinality column to the end of the sort order.

You can evaluate replay delete performance using the following command:

```
=>SELECT EVALUATE_DELETE_PERFORMANCE ('projection or table name')
```

Can the presence of a large number of deleted records affect query performance?

Tables with hundreds of millions of records that have 20% or more deleted records can impact query performance. Performance issues can occur with such tables if the queries perform a full table scan and if deleted records were not pruned during query processing.

Purging Deleted Records

How does Vertica purge deleted records?

Purging deleted records in a storage container requires a rewrite of the storage container. Purging deleted records more often than is necessary results in a significant increase in I/O. Because the I/O can increase significantly, Vertica purges deleted records only when storage containers are rewritten by node-based operations, such as:

- Recovery
- Reorganize
- Rebalance
- Refresh
- Tuple Mover mergeout

The mergeout operation merges ROS containers that qualify for mergeout based on a strata algorithm for active partitions and for tables without partitions. During mergeout, the Tuple Mover purges any records from qualifying ROS containers deleted prior to the AHM epoch.

When a new partition is created, the current active partition becomes inactive. ROS containers of the most recent inactive partition are merged into a single ROS container. Records deleted prior the AHM epoch are purged during this process.

When deleting records from an inactive partition, if the percentage of deleted records in the ROS container exceeds 20%, the mergeout algorithm purges deleted records from ROS containers. The PurgeMergeoutPercent configuration parameter controls this percentage.

The mergeout operation for inactive partitions has a lower priority than the mergeout operation in both active partitions and tables without partitions.

This purge policy means that some ROS containers with deleted records may not be purged unless you perform a manual purge.

When and how should I perform a manual purge?

The *manual purge* operation rewrites a storage container even if there is one deleted record in a storage container. Unless you have a large table with hundreds of millions of rows of which 20% or more is deleted records, avoid running a manual purge. You can run the following query to identify all tables that have more than 100 million records, of which 20% or more is deleted data:

```
=>SELECT projection_name, sum(deleted_row_count)*100/sum(total_row_count)
as percentage deleted
 FROM STORAGE CONTAINERS
 GROUP BY 1 HAVING sum(total_row_count) > 100000000 and
 sum(deleted_row_count)*100/sum (total_row_count) > 20
 ORDER BY 2 DESC;
```

Verify that the AHM is advancing smoothly and projections are optimized for both delete and replay delete.

Before you decide to perform a manual purge, use the DELETE_VECTORS system table to check the commit epochs of the deleted records you want to purge. Verify that the commit epochs are less than the current AHM epoch. If you want to purge deleted records from a specific partition, use the PURGE_PARTITION function because it targets a specific ROS container.

> Note Using the PURGE_PROJECTION function targets all ROS containers that contain deletes and slows performance.

You can run the following query to detect projections and partitions that contain more than 20% deleted data. The following example shows a partition with 100 million records:

```
=>SELECT table_schema_projection_name, partition_key
 sum(deleted_row_count)*100/sum(ros_row_count)
 as percentage deleted FROM PARTITIONS GROUP BY 1,2,3
 HAVING sum(ros_row_count)> 100000000 and
 sum(deleted_row_count)*100/sum(ros_row_count)> 20
 ORDER BY 2 DESC;
```

What best practices should I follow when deleting large volumes of data from a single table?

To delete a large volume of data from a table, follow these steps:

1. Move out data from WOS to ROS, using the command:

   ```
   => SELECT do_tm_task ('moveout',<table_name>);
   ```

2. Run a DELETE or an UPDATE statement using DIRECT HINT:

   ```
   =>DELETE /*+direct*/
    FROM <table_name> where <column1> = 1; commit;
   ```

3. Get the current value of the AHM epoch, using the command:

   ```
   =>SELECT get_ahm_epoch();
   ```

4. Manually advance the AHM, using the command:

```
=>SELECT make_ahm_now();
```

5. Verify that the AHM advanced:

```
=>SELECT get_ahm_epoch();
```

6. If the DELETE statement affected a single partition or a few partitions, run the PURGE_PARTITION function. Otherwise, run the PURGE_TABLE function. Run the PURGE command using a resource pool that has at least 4GB of query budget. A higher budget improves performance of the PURGE command, especially if the table is wide. You can find the query budget of a resource pool by running the RESOURCE_POOL_STATUS statement.

> Important if you delete most of the data in your table, you can create a new table by running the following command:
>
> `=>CREATE TABLE <new_table_name> like <table_name>`

You can insert data that must be retained in the new table and then swap the tables.

For More Information

For more information about deletes in Vertica, see Best Practices for Deleting Data in the Vertica Knowledge Base.

Deletes in Vertica: The FAQs