Vertica での Delete：よくあるご質問

Applies to Vertica 7.2.x and earlier

March, 2017

原文は[こちら]

目次

ドキュメントの概要

本書は、削除に関するもっとも重要な質問と Vertica データベースのパフォーマンスに与える影響について解説しています。これらのよくある質問は、次のカテゴリに分類されています。

Delete の基本

Vertica はどのように delete（削除）や update（更新）を処理しますか？

Vertica で DELETE 文を実行しても、データはストレージコンテナからすぐに削除されません。代わりに、DELETE 文は、削除マークされたレコードを含む各 WOS および ROS コンテナを指すデリートベクターを追加します。各デリートベクターには、コンテナ内の削除されたレコードの位置と、DELETE 文がコミットされたエポックの情報が含まれます。

UPDATE 文は DELETE に続いて INSERT を実行します。テーブルに対して SELECT クエリを実行すると、Vertica はデリートベクターにマークされたレコードをフィルタリングし、そのレコードを結果から除外します。

AHM エポックとは何ですか？

*Ancient history mark (AHM)*は、物理ストレージから論理削除（delete）されたデータを物理削除（purge）する前のエポックです。デフォルトでは、Vertica は 180 秒間隔で AHM エポックを進めます。クラスタ内のノードが停止している、または、データベースにリフレッシュされていないプロジェクションが含まれている場合、AHM エポックは進みません。AHM は、last good epoch（LGE）を超えることはありません。

次のコマンドを使用して、AHM エポック、LGE エポック、および current epoch（現在のエポック）を確認できます。

```
=>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)
```

Replay delete とは何ですか？

DELETE 文は、WOS に格納され、DVWOS とラベル付けされたデリートベクターを作成します。個々の DELETE 文は、DELETE 文で削除されたレコードを含むすべてのストレージコンテナに対して 1 つの DVWOS オブジェクトを作成します。DIRECT ヒントを含む DELETE 文は、ファイルシステムに直接格納され、DVROS とラベル付けされたデリートベクターを作成します。

Tuple Mover の moveout 処理により、WOS メモリーから ROS にデータが移動されると、複数の WOS コンテナが 1 つの大きなソートされた ROS コンテナに結合されます。Tuple Mover の moveout 処理では、WOS からデリートベクターを移動して新しい DVROS コンテナを作成します。

移動されたデータ内のレコードの一部が削除された場合、これらのレコードは新しく作成された ROS コンテナ内の新しい位置にあります。デリートベクターが移動された後、デリートベクターは ROS コンテナ内の削

除されたレコードの新しい位置を取得情報します。*Replay delete* とは、デリートベクターを再構築して、ストレージコンテナ全体のレコードの移動に適応させるプロセスです。

Replay delete 処理は、Tuple Mover の mergeout 処理中に行われます。Mergeout 処理でレコードを削除した ROS コンテナがマージされると、AHM エポック以前に削除されたレコードはすべて削除されます。Purge できないレコードは、新しく作成された ROS コンテナに新しい位置を持ちます。このプロセスでは、新しい ROS コンテナで purge されなかった削除されたレコードの位置を指すデリートベクターを再構築する必要があります。

Vertica は、ノードのリカバリ、再編成、クラスタからのノードの追加または削除時のリバランス、およびプロジェクションのリフレッシュ中に replay delete を実行します。これらの処理は、AHM エポック以前に削除されたデータも purge します。

！重要！

Purge（物理削除）できない delete（論理削除）されたレコードだけが、replay delete に参加します。

どのシステムテーブルがデリートベクター情報を保持しますか？

Vertica の DELETE_VECTORS というシステムテーブルが、デリートベクターおよび削除されたレコードの数に関する情報を保持します。詳細については、Vertica ドキュメントの DELETE_VECTORS を参照してください。

Delete のライフサイクル

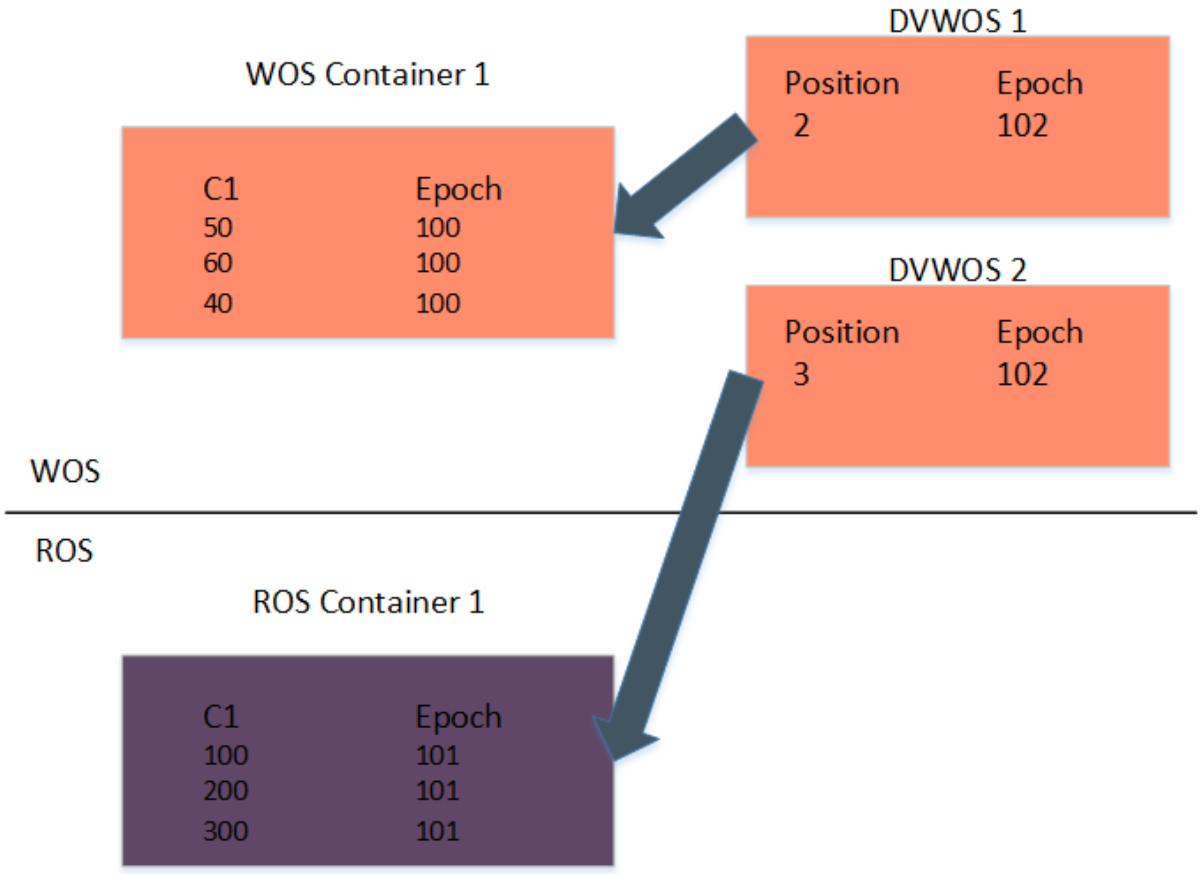
Delete のライフサイクルのプロセスの手順はどうなっていますか？

以下の図および手順は、delete のライフサイクルのプロセスの例を示しています。この例は、1 つの WOS コンテナと 1 つの ROS コンテナを持つテーブルを示しています。

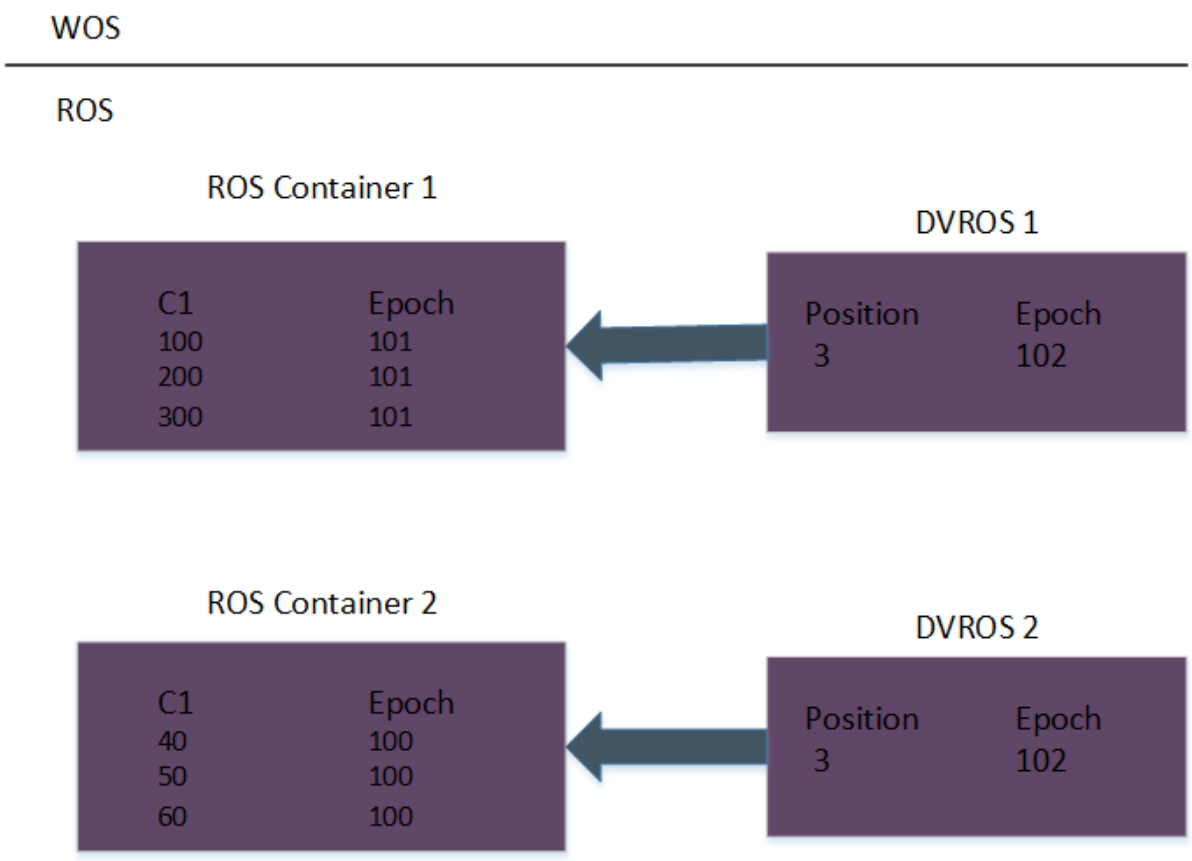
手順 1：次の DELETE 文を実行します。

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

このステートメントは、WOS コンテナと ROS コンテナのデリートベクター（DVWOS）を作成します。Vertica は、このステートメントをエポック 102 でコミットします。



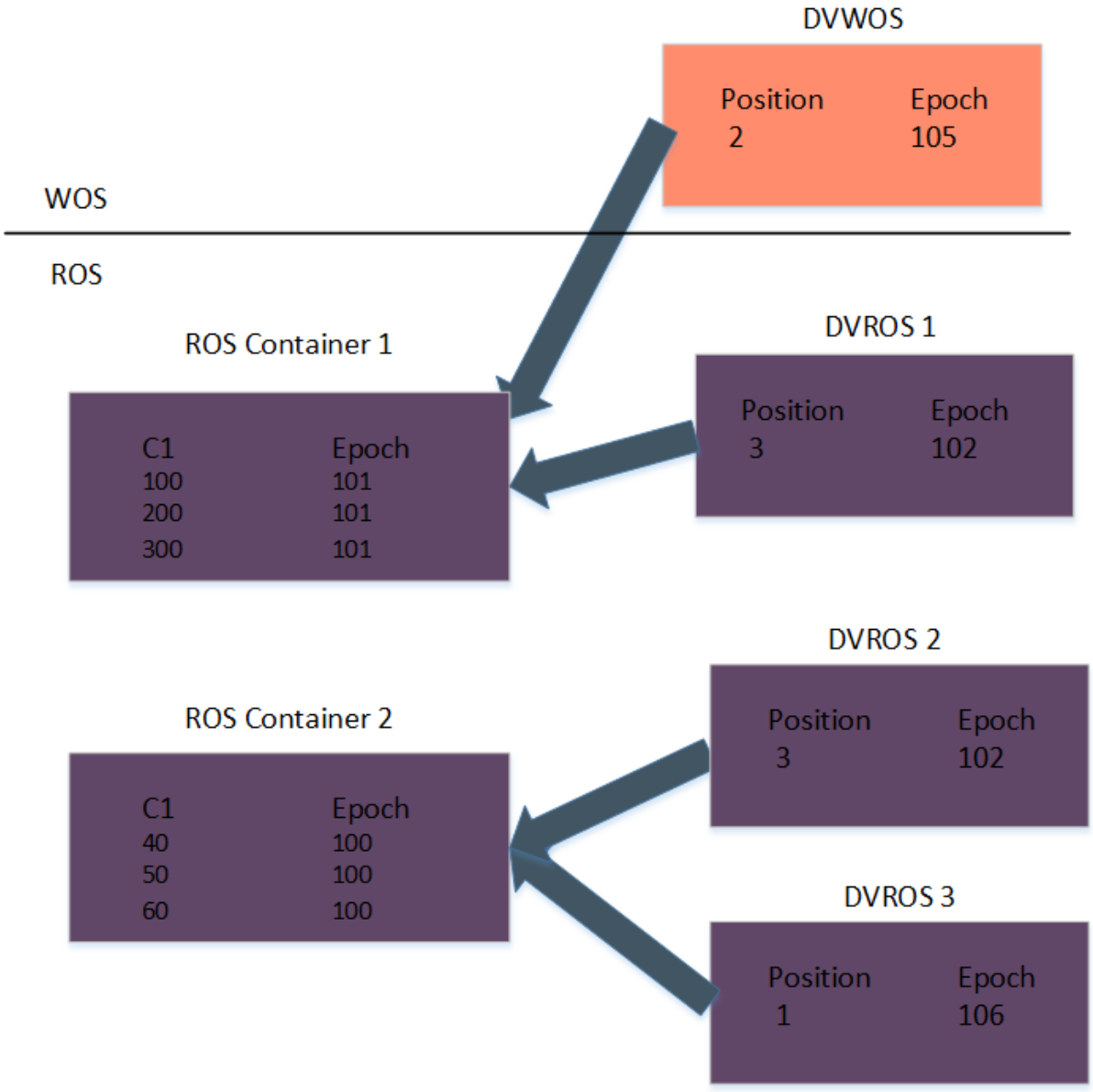
手順 2：Tuple Mover は、WOS から ROS にデータを移動し、ROS（DVROS）にデリートベクターを作成する moveout 処理で応答します。図のように、moveout 処理後にデータの位置情報が変更され、DVROS が新しい位置情報をキャプチャします。


WOS
ROS
ROS Container 1
C1
100
200
300
Epoch
101
101
101
DVROS 1
Position
3
Epoch
102
ROS Container 2
C1
40
50
60
Epoch
100
100
100
DVROS 2
Position
3
Epoch
102

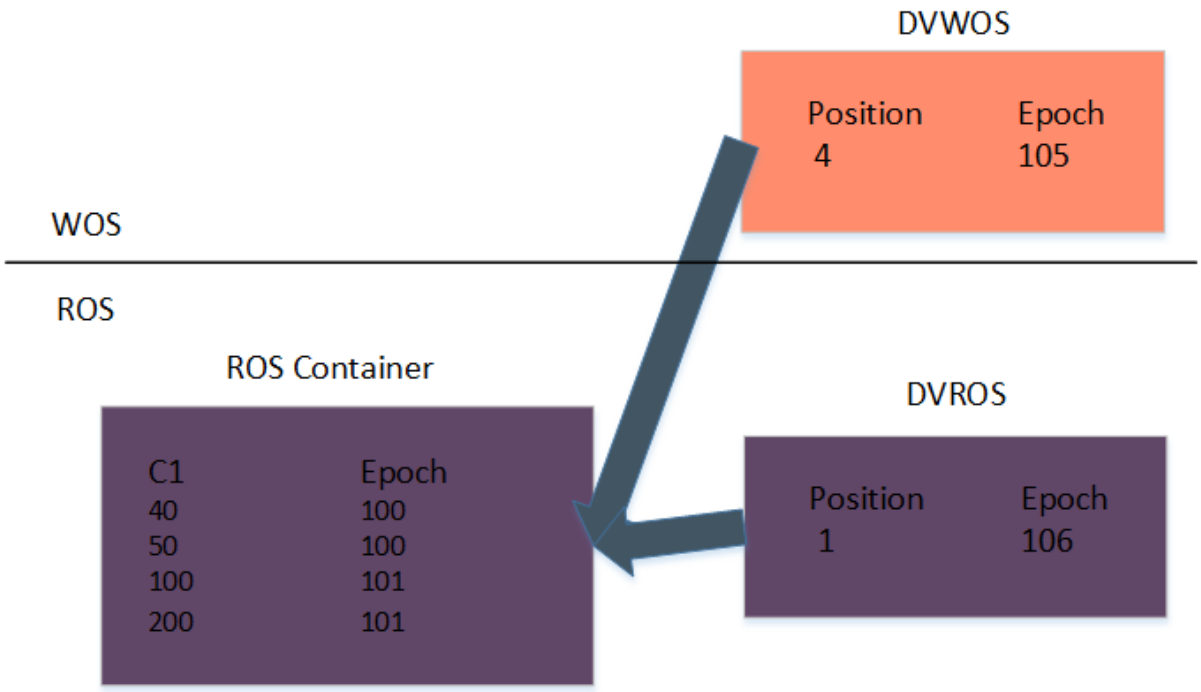
手順 3：AHM は 103 に進みます。次の delete 文を実行します。

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

これらのステートメントは、それぞれエポック 105 とエポック 106 でコミットされます。DELETE 文は新しい DVWOS コンテナを作成し、direct ヒントを使用する DELETE 文は DVROS コンテナを作成します。



手順 4：Tuple Mover は、ROS コンテナを単一のソートされた ROS コンテナにマージする mergeout 処理で応答します。Mergeout 処理は、AHM の前に削除されたすべてのレコードを purge します。次の図は、エポック 105 および 106 で削除されたレコードを purge できなかったことを示しています。DELETE 文のコミットエポックが AHM エポックより大きいため、purge は行われませんでした。



プロジェクション設計の考慮事項

ベストな delete と replay delete のパフォーマンスを得るためにはプロジェクションをどのように設計すべきでしょうか？

DELETE または UPDATE を含むテーブルについては、次のプロジェクション設計の考慮事項を検討してください。プロジェクション設計がこれらの考慮事項を満たしていない場合、Vertica は新しいプロジェクションを作成し、リフレッシュを実行し、古いプロジェクションを削除することを推奨します。

注意

データベースデザイナーを使用する場合、データベースデザイナーに渡されたクエリの一覧に DELETE 文を含めることによって、最適化されたプロジェクションを作成可能です。

Delete のパフォーマンス

DELETE 文のパフォーマンスは、DELETE 文で使用される述語の列がテーブルにひもづくすべてのプロジェクションに存在する場合に最適な状態であるといえます。Vertica は、テーブルにひもづくすべてのプロジェクションにクエリの述語の列がある場合、最適化された削除アルゴリズムを選択します。最適化された削除アルゴリズムでは、Vertica は、テーブルにひもづく各プロジェクションのレコードの位置と削除マークがついたレコードを選択します。アルゴリズムは、位置を使用して、それぞれのプロジェクションのためのデリートベクターを作成します。このアルゴリズムは高速で、DELETE 文の述語の列を持たないプロジェクションがない限り、デフォルトの動作です。

最適化されていない削除アルゴリズムを使用すると、Vertica はテーブルにひもづけられた 1 つのプロジェクションから削除マークが付いたレコードを選択します。次に、データベースは、すべてのプロジェクションを検索して、デリートベクターを構築するために必要な削除されたレコードの位置情報を見つけます。最適化されていないアルゴリズムを使用すると、最適化されたアルゴリズムを使用した場合よりも処理が遅くなります。

Replay Delete のパフォーマンス

最適な replay delete のパフォーマンスを得るには、カーディナリティの最も高い列がプロジェクションのソート順の末尾に表示されます。Replay delete 処理では、削除されたレコードの位置が新しいストレージコンテナ内で Vertica によって検出される必要があります。Vertica は最初に、プロジェクションのソート順にある列を使用して、削除されたレコードを検索します。ソート順に高いカーディナリティ列がある場合、Vertica は新しい位置情報をより迅速に見つけます。

場合によっては、Vertica はソート順にある列を使用して、削除された行の正確な位置を特定できません。このような状況では、Vertica はすべてのプロジェクションの列と一致させるため、replay delete が遅くなります。

Replay delete のパフォーマンスの観点でプロジェクションを評価するにはどうすればよいですか？

次のステートメントを使用して、単一のプロジェクションまたはテーブルにひもづけられたすべてのプロジェクションの replay delete のパフォーマンスを評価します。

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

出力は次のいずれかです。

- No projection delete performance concerns found
  （該当のプロジェクションに delete のパフォーマンスの問題が見つかりません。）

または

- Projection exhibits delete performance concerns, followed by a list of concerns
  （該当のプロジェクションは delete のパフォーマンスの観点で懸念があるため、懸念点のリストを表示します。）

リカバリ中に特定のプロジェクションの replay delete 実行に非常に時間がかかるのを避けるにはどうすればよいですか？

ノードがダウンすると、AHM エポックは進みません。ノードがダウンしているとき、または、ノードがリカバリしているときに発生する delete 処理は、AHM エポックよりも大きいコミットエポックを持ちます。リカバリ処理の一環として、これらの delete はリカバリノードで再生される必要があります。ターゲットテーブルが大きく、ターゲットテーブルのプロジェクションが、最適な削除、あるいは、replay delete のパフォーマンスに適した状態で設計されていない場合、問題が発生する可能性があります。このような場合、リカバリ実行中に、特定のプロジェクションが replay delete のステータスでスタックしてしまうことがあります。

このような状況に遭遇した場合は、既存のプロジェクションを、最適なプロジェクション設計の推奨に従ったプロジェクションに置き換えてください。ソート順の最後にカーディナリティの高い列を追加することで実行できます。

次のコマンドを使用して、replay delete のパフォーマンスを評価できます。

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

多数の delete されたレコードの存在がクエリのパフォーマンスに影響する可能性はありますか？

レコードが 20%以上削除された数億のレコードを持つテーブルは、問合せのパフォーマンスに影響を与える可能性があります。クエリがフルテーブルスキャンを実行し、クエリ処理中に削除されたレコードがプルーニングされなかった場合、そのようなテーブルでパフォーマンスの問題が発生する可能性があります。

論理削除されたレコードの物理削除

Vertica は論理削除されたレコードをどのように物理削除（purge）しますか？

ストレージコンテナ内の論理削除（delete）されたレコードを物理削除（purge）するには、ストレージコンテナの書き換えが必要です。論理削除（delete）されたレコードを必要以上に物理削除（purge）すると、I/O が大幅に増加します。I/O が大幅に増加する可能性があるため、Vertica は、次のようなノードベースの操作によってストレージコンテナが書き換えられた場合にのみ、論理削除されたレコードを物理削除します。

- リカバリ
- 再編成
- リバランス
- リフレッシュ
- Tuple Mover mergeout

mergeout 処理は、アクティブパーティションとパーティションのないテーブルのストラータアルゴリズムに基づいて、mergeout する必要のある ROS コンテナをマージします。mergeout 中に、Tuple Mover は、AHM エポック以前に削除された対象となる ROS コンテナからレコードを purge します。

新しいパーティションが作成されると、現在のアクティブなパーティションは非アクティブになります。最新の非アクティブパーティションの ROS コンテナは、単一の ROS コンテナに統合されます。この処理中に AHM エポック以前に削除されたレコードは purge されます。

非アクティブパーティションからレコードを削除する場合、ROS コンテナ内の削除されたレコードの割合が 20%を超えると、mergeout アルゴリズムは削除されたレコードを ROS コンテナから purge します。PurgeMergeoutPercent という構成パラメーターは、このパーセンテージを制御します。

非アクティブなパーティションの mergeout 処理は、アクティブなパーティションとパーティションのないテーブルの両方の mergeout 処理よりも優先順位が低くなります。

この purge ポリシーは、手動 purge を実行しない限り、削除されたレコードを持つ一部の ROS コンテナが purge されないことを意味します。

手動 purge をいつどのように実行する必要がありますか？

手動 purge 処理では、ストレージコンテナに削除されたレコードが 1 つであっても、ストレージコンテナが書き換えられます。何億行ものレコード数で 20%以上の削除された大きなテーブルを持っていない限り、purge を実行しないでください。次のクエリを実行すると、1 億を超えるレコードを持ち、かつ、20%以上削除されているすべてのテーブルを識別できます。

```
=>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;
```

Vertica では、AHM がスムーズに進んでいることと、プロジェクションが delete と replay delete に最適化されていることを確認することをお勧めします。

手動 purge を実行することを決定する前に、DELETE_VECTORS というシステムテーブルを使用して、purge したい削除されたレコードのコミットエポックをチェックしてください。コミットエポックが現在の AHM エポックよりも小さいことを確認してください。特定のパーティションから論理削除されたレコードを物理削除する場合は、特定の ROS コンテナを対象としているため、PURGE_PARTITION 関数を使用します。

注意

PURGE_PROJECTION 関数を使用すると、削除を含むすべての ROS コンテナがターゲットとなり、パフォーマンスが低下します。

次のクエリを実行すると、論理削除されたデータが 20%を超えるプロジェクションとパーティションを検出できます。次の例は、1 億のレコードを持つパーティションを示しています。

```
=>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;
```

1 つのテーブルから大量のデータを論理削除する場合のベストプラクティスは？

大量のデータをテーブルから削除するには、次の手順を実行します。

1. 次のコマンドを使用して、WOS から ROS にデータを移動します。

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

2. DIRECT ヒントを使用して DELETE または UPDATE 文を実行します。

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

3. 次のコマンドを使用して、AHM エポックの現在の値を取得します。

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

4. 次のコマンドを使用して、AHM を手動で進めます。

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

5. AHM が進んでいることを確認します。

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

6. DELETE 文が単一のパーティションまたはいくつかのパーティションに影響する場合は、PURGE_PARTITION 関数を実行します。それ以外の場合は、PURGE_TABLE 関数を実行します。クエリのバジェットが 4GB 以上のリソースプールを使用して PURGE コマンドを実行します。バジェットが高くなると、PURGE コマンドのパフォーマンスが向上します（特に、ワイドテーブルである場合）。RESOURCE_POOL_STATUS ステートメントを実行すると、リソースプールのクエリバジェットを確認することができます。

！重要！

テーブル内の大部分のデータを削除する場合は、次のコマンドを実行して新しいテーブルを作成できます。

```
=>CREATE TABLE <new_table_name> like <table_name>
```

新しいテーブルに保持する必要があるデータを挿入してから、テーブルをスワップすることができます。

詳細情報

Vertica での削除の詳細については、Vertica ナレッジベースでの Best Practices for Deleting Data を参照してください。