Vertica のエポックについて

March, 2017

原文はこちら

MICRO FOCUS

目次

エポックの概要

エポックは、Vertica のデータの論理タイムスタンプを表す 64 ビットの数値です。すべての行には、コミットされたエポックを記録する暗黙的に格納された列があります。

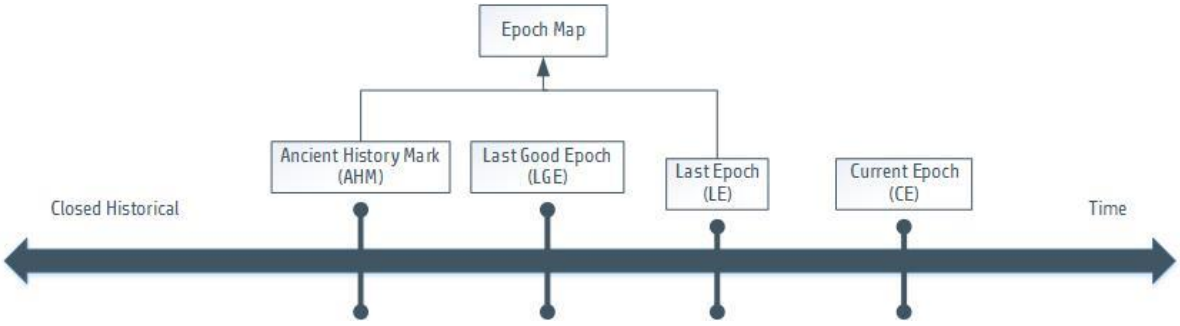
```
=> CREATE TABLE testdata (a INT, b INT);
=> INSERT INTO testdata VALUES (1,2);
=> INSERT INTO testdata VALUES (3,4);
=> COMMIT;
=> INSERT INTO testdata VALUES (5,6);
COMMIT;
=> SELECT a,b,epoch FROM testdata;

OUTPUT
--------
      1
(1 row)

 OUTPUT
--------
      1
(1 row)

COMMIT

 OUTPUT
--------
      1
(1 row)

COMMIT
 a | b | epoch
---+---+-------
 1 | 2 |  1898
 3 | 4 |  1898
 5 | 6 |  1899
(3 rows)
```

このエポックは、システムの論理状態が変更されたとき、またはデータが DML 操作（INSERT、UPDATE、MERGE、COPY、または DELETE）でコミットされたときに進みます。EPOCHS システムテーブルは、クローズされた各エポックの日時およびクローズされたエポックに対応するエポック番号を含みます。この情報により、どの期間がどのエポックに関係しているかを判断できます。

```
=> SELECT * FROM epochs;
      epoch_close_time          | epoch_number
--------------------------------+--------------
 2015-11-09 17:47:21.013641+00 |         1350
 2015-11-09 18:03:30.454707+00 |         1351
 2015-11-09 18:04:48.329494+00 |         1371
 2015-11-09 18:18:42.796702+00 |         1372
 2015-11-09 18:19:52.738018+00 |         1392
 2015-11-09 18:26:43.655577+00 |         1393
 2015-11-11 15:21:47.918074+00 |         1394
 2015-11-11 15:23:16.80952+00  |         1897
 2015-11-20 18:13:20.324436+00 |         1898
 2015-11-20 18:13:20.372756+00 |         1899
(10 rows)
```

エポックの種類

エポックマップは、AHM と LE の間にあるエポックのリストです。エポックマップは、データベースカタログに格納されます。



さまざまな種類のエポックは次のとおりです。

Current Epoch (CE)

Current epoch（現在のエポック）は、COMMIT 操作後の最後のエポック（LE）になるオープン状態のエポックです。COMMIT 時の current_epoch は、その DML のエポックです。

```
=> SELECT CURRENT_EPOCH FROM SYSTEM;
CURRENT_EPOCH
---------------
629415

(1 row)

=> INSERT INTO test VALUES (10,20); COMMIT;
OUTPUT
```

```
--------
1
(1 row)

COMMIT
=> SELECT current_epoch FROM SYSTEM;
 current_epoch
---------------
        629416
(1 row)
```

Latest Epoch (LE)

Latest epoch（最新のエポック）は、一番最後に閉じたエポックです。COMMIT 操作後の current epoch（現在のエポック）は、latest epoch（最新のエポック）になります。

Checkpoint Epoch (CPE)

プロジェクションごとのチェックポイントエポックは、WOS にデータがない状態の最新のエポックです。プロジェクションがリカバリできるポイントです。Tuple Mover の moveout 処理は、データを WOS から ROS に移動させながらプロジェクションの CPE を進めます。プロジェクションのチェックポイントのエポックは、PROJECTION_CHECKPOINT_EPOCHS システムテーブルで確認できます。

```
=> SELECT node_name, projection_schema schama, projection_name p_name,
is_up_to_date UTD, checkpoint_epoch CPE, would_recover WR, is_behind_ahm
BAHM from projection_checkpoint_epochs;

       node_name        |  schema | p_name                    |   UTD |
CPE    | WR    | BAHM
------------------------+---------+---------------------------+-------+---
--------+-------+-----
 v_test_israel_node0001 | public  |   product_dimension_b1    |   t   |
1347   |   f   |   f
 v_test_israel_node0003 | public  |   product_dimension_b1    |   t   |
1347   |   f   |   f
 v_test_israel_node0002 | public  |   product_dimension_b1    |   t   |
1347   |   f   |   f
 v_test_israel_node0002 | store   |   store_dimension_b1      |   t   |
1347   |   f   |   f
 v_test_israel_node0003 | store   |   store_dimension_b1      |   t   |
1347   |   f   |   f
 v_test_israel_node0001 | store   |   store_dimension_b1      |   t   |
1347   |   f   |   f
 v_test_israel_node0002 | public  |   promotion_dimension_b1 |   t   |
1347   |   f   |   f
```

```
 v_test_israel_node0001 | public  |   promotion_dimension_b1 |   t   |
1347   |   f   |   f
 v_test_israel_node0003 | public  |   promotion_dimension_b1 |   t   |
1347   |   f   |   f
 v_test_israel_node0001 | public  |   warehouse_dimension_b1 |   t   |
1347   |   f   |   f
 v_test_israel_node0002 | public  |   warehouse_dimension_b1 |   t   |
1347   |   f   |   f
 v_test_israel_node0003 | public  |   warehouse_dimension_b1 |   t   |
1347   |   f   |   f
```

Last Good Epoch (LGE)

すべてのノードで最小のチェックポイントエポックは、last good epoch（最後の正常なエポック）として知られています。Last good epoch とは、手動リカバリで回復できる最新のエポックを指します。LGE は、ディスク上のすべてのデータのスナップショットで構成されています。クラスタが異常終了した場合、LGE の後のデータは失われます。Tuple Mover は CPE を進め、新しい LGE を設定します。Tuple Mover が失敗した場合、データは WOS から ROS に移動しません。したがって、データは CPE および LGE を進めません。各ノードには、ノード上のプロジェクションの最小チェックポイントエポックである LGE があります。Vertica は、ノードごとの LGE からクラスターの LGE を計算し、クラスターの LGE が K-safety を利用して最大限可能な LGE を提示するようにします。Last good epoch を確認するには、次のコマンドを使用します。

```
=> SELECT GET_LAST_GOOD_EPOCH();
 GET_LAST_GOOD_EPOCH
---------------------
                1347
(1 row)
```

リカバリエポックをチェックすることによって、各ノードの LGE を確認できます。

```
=> SELECT GET_EXPECTED_RECOVERY_EPOCH();
INFO 4544:  Recovery Epoch Computation:
Node Dependencies:
011 - cnt: 253
101 - cnt: 253
110 - cnt: 253
111 - cnt: 239
Nodes certainly in the cluster:
        Node 0(v_test_israel_node0001), epoch 1347
        Node 1(v_test_israel_node0002), epoch 1347
Filling more nodes to satisfy node dependencies:
Data dependencies fulfilled, remaining nodes LGEs don't matter:
        Node 2(v_test_israel_node0003), epoch 1347
--
 GET_EXPECTED_RECOVERY_EPOCH
```

```
----------------------------
                        1347
(1 row)
```

Ancient History Mark (AHM)

大きなエポックマップは、カタログのサイズを大きくする可能性があります。Ancient history mark は、それより前の履歴データを物理ストレージから purge することができるエポックです。AHM の前に履歴クエリを実行することはできません。デフォルトでは、Vertica は AHM を LGE と同じになるように 5 分間隔で進めます。クラスター内のノードで、リフレッシュされていないプロジェクションが存在したり、ノードが停止している場合、AHM は進みません。AHM は LGE を決して上回ることはありません。AHM を確認するには、次のコマンドを使用します。

```
=> SELECT GET_AHM_EPOCH();

GET_AHM_EPOCH

---------------

          1347

(1 row)
```

エポックの仕組み

DML トランザクション（INSERT、UPDATE、MERGE、COPY、および DELETE）の COMMIT を使用すると、CE と LE の両方が進みます。Current epoch が 1 だけ動くと、LE も 1 だけ動きます。Current epoch（現在のエポック）は latest epoch（最新のエポック）になります。Vertica は、DML トランザクションのタイプに応じて、次の処理を行います。

- データが INSERT 文または COPY 文によってロードされる場合、各行は、行がコミットされた時間を表すエポック値を持ちます。
- DELETE 文を使用してデータを削除すると、Vertica はエポックを格納するデリートベクターを作成します。デリートベクターは、削除用にマークされた ROS コンテナ内の位置情報を格納します。

次のコマンドは、Vertica が新しいエポックを作成するタイミングを示しています。

```
=> CREATE TABLE test_epochs ( c1 int);
CREATE TABLE

=> SELECT CURRENT_EPOCH, AHM_EPOCH, LAST_GOOD_EPOCH FROM SYSTEM;
 CURRENT_EPOCH | AHM_EPOCH | LAST_GOOD_EPOCH
---------------+-----------+-----------------
          1348 |      1347 |            1347
(1 row)
```

```
-- Insert new data
=> INSERT INTO test_epochs VALUES (1);
 OUTPUT
--------
      1
(1 row)

-- Because the INSERT was not committed, no epoch was recorded
=> SELECT epoch, * FROM test_epochs;
 epoch | c1
-------+----
       |  1
(1 row)

=> COMMIT;
COMMIT

-- The COMMIT transaction uses last current epoch and current epoch
becomes the latest epoch.

=> SELECT epoch, * FROM test_epochs;
 epoch | c1
-------+----
  1348 |  1
(1 row)

-- Current epoch advances because of the last COMMIT.
=> SELECT CURRENT_EPOCH, AHM_EPOCH, LAST_GOOD_EPOCH FROM SYSTEM;
 CURRENT_EPOCH | AHM_EPOCH | LAST_GOOD_EPOCH
---------------+-----------+-----------------
          1349 |      1347 |            1347
(1 row)

-- In Vertica, UPDATE is DELETE + INSERT.

=> UPDATE test_epochs SET c1 = 2 WHERE c1 = 1;
 OUTPUT
--------
      1
(1 row)
```

```
- Because transaction has not been committed, the epoch column is empty.
=> SELECT epoch, * FROM test_epochs;
 epoch | c1
-------+----
       |  2
(1 row)

=> COMMIT;
COMMIT

- After the COMMIT, the last epoch becomes the last current epoch.
=> SELECT epoch, * FROM test_epochs;
 epoch | c1
-------+----
  1349 |  2
(1 row)

=> SELECT CURRENT_EPOCH, AHM_EPOCH, LAST_GOOD_EPOCH FROM SYSTEM;
 CURRENT_EPOCH | AHM_EPOCH | LAST_GOOD_EPOCH
---------------+-----------+-----------------
          1350 |      1348 |            1348
(1 row)
```

SELECT ステートメントを実行すると、Vertica は latest epoch（最新のエポック）でデータを取得します。Vertica は同じトランザクション内のコミットされていないデータも取得します。

AHM よりも前に削除されたデータは、物理削除の対象となります。データの削除の詳細については、Best Practices for Deleting Data を参照してください。

エポックに関する問題のトラブルシューティング

次の情報は、エポックに関する問題のトラブルシューティング方法に関するヒントを提供します。

Last Good Epoch が進まない

Last good epoch が進まない状況があります。LGE が進むと、次の結果が表示されます。Tuple Mover がデータを WOS から ROS に移動すると、LGE は次のように進みます。

```
=>SELECT CURRENT_EPOCH, LAST_GOOD_EPOCH FROM SYSTEM ;
 CURRENT_EPOCH   |   LAST_GOOD_EPOCH
---------------+-----------------
        630384   |        620380
```

- LGE が進んだことを確認できない場合、WOS にデータがあるかどうかを確認します。

```
=>SELECT sum(wos_used_bytes) from projection_storage ;
  sum
-------
 49152
```

- WOS にデータがある場合、強制的に moveout 処理を実行します。

```
=> SELECT do_tm_task('moveout');
```

- 静的なデータがある場合、Tuple Mover をチェックします。Tuple Mover の moveout 処理は、以下の理由により失敗します。
    - Tuple Mover がテーブルの T ロックを取得できない場合
    - Tuple Mover が ROS コンテナの最大数に達した場合、ROS プッシュバックが発生します。ROS プッシュバックは、ROS コンテナの最大数に達したために moveout 処理で新しいコンテナを作成できない場合に発生します。
    - WOS に 1024 以上のパーティションがある場合

Tuple Mover が失敗する理由を調べるには、vertica.log ファイルを確認してください。

```
$ FIND <catalog path> -name vertica.log   |   xargs grep 'TM Moveout.*
<ERROR>'
```

- Replay delete により、Tuple Mover の処理が遅延します。Tuple Mover の動きを確認するには、次のコマンドを使用します。

```
=> SELECT * from tuple_mover_operations where is_executing ;
```

Ancient History Mark が進まない

Ancient history mark が進まない時があります。AHM は次のシナリオでは進みません。

- リフレッシュされていないプロジェクションがある場合。これを解決するには、DDL をエクスポートし、プロジェクションをリフレッシュします。リフレッシュされていないプロジェクションを見つけるためには、次のコマンドを使用してください。

```
=> SELECT * FROM projections where is_up_to_date = 'f';
```

- リフレッシュするプロジェクションが大きなテーブルの場合。Best Practices for Refreshing Large Projections に記載されているワークアラウンドを試してください。このワークアラウンドは、プロジェクションのリフレッシュ処理が終了するまで Vertica が AHM を保持しないようにします。

```
-- When try to advance the AHM, you get an error

=> SELECT MAKE_AHM_NOW();
ERROR 2154:  AHM must lag behind the create epoch of unrefreshed
projection public.test_2_b0 (Create epoch: 1372)

-- Export DDL to save if want to deploy later
```

```
=> SELECT export_objects('','test_epochs');
....

-- Drop projection

=> drop projection test_2;
DROP PROJECTION

-- Advance the AHM
=> SELECT MAKE_AHM_NOW();
         MAKE_AHM_NOW
------------------------------
 AHM set (New AHM Epoch: 1393)
(1 row)
```

- クラスター内のすべてのノードが UP でない場合。次のコマンドを使用して、ノードの状態を確認します。

```
=> SELECT * FROM nodes where node_state != 'UP';
```

Vertica のエポック関数

Vertica は以下の Epoch 関数を提供します。

関数名	説明
GET_AHM_EPOCH	AHM が配置されているエポックの番号を返します。
GET_CURRENT_EPOCH	COPY、INSERT、UPDATE、および DELETE 操作によってデータが書き込まれる current epoch（現在のエポック）の番号を返します。
GET_LAST_GOOD_EPOCH	Last good epoch（最後の正常なエポック）の番号を返します。Last good epoch は、手動リカバリを使用して回復できる最近のエポックを指します。
SET_AHM_EPOCH	AHM を指定されたエポックに設定します。SET_AHM_EPOCH は通常、テスト目的で使用されます。

詳細については、Vertica ドキュメントの Epochs を参照してください。