Vertica でのパーティション：よくあるご質問

April, 2017

原文はこちら

MICRO FOCUS

目次

ドキュメントの概要

Vertica のパーティション化機能は、1 つまたは複数の列の値に基づいて、1 つの大きなテーブルをより小さなかたまりに分割します。パーティションはデータのライフサイクルマネージメントを容易にし、述語がパーティション式に含まれるクエリのパフォーマンスを向上させます。

このドキュメントでは、Vertica データベースのパーティションに関する最も重要な質問に対する答えが記載されています。これらの質問は次のカテゴリに分類されます。

パーティションの基本

パーティションはどのように機能しますか？

データの保存ポリシーを作成する必要があるとします。たとえば、5 年間だけのデータを保持する必要があるかもしれません。5 年以上経過したすべてのデータを削除したいとします。

Vertica のパーティショニング機能により、このデータを効率的に管理することが容易になります。これがどのように機能するか見てみましょう。

次のデータを含む trade という名前のテーブルがあるとします。

- Trade date (tdate)
- Ticker symbol (tsymbol)
- Trade time (ttime)

次の CREATE TABLE ステートメントを使用して、Vertica が取引が行われた年に基づいてデータを分割するように指定します。

```
=> CREATE TABLE trade (
   tdate DATE NOT NULL,
   tsymbol VARCHAR(8) NOT NULL,
   ttime TIME)
   PARTITION BY EXTRACT(year FROM tdate);
```

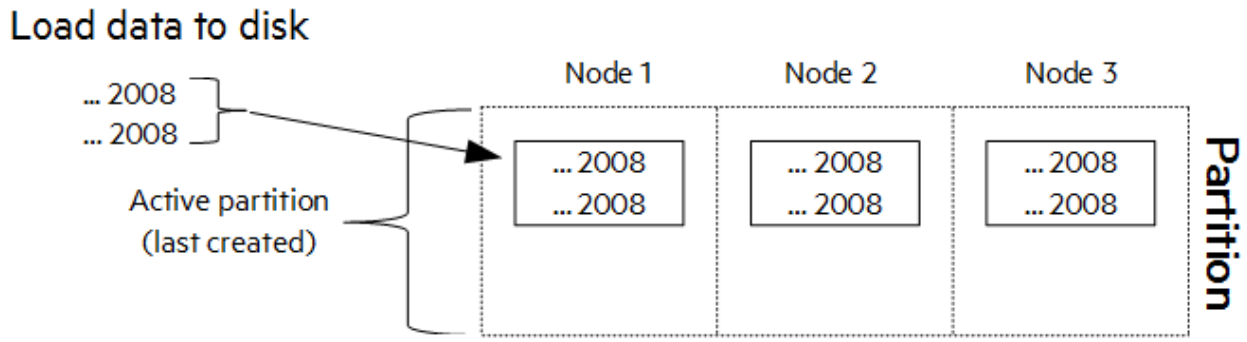
trade テーブルにデータをロードすると、Vertica はパーティション式（この例では暦年）に基づいてデータを分離します。

```
PARTITION BY EXTRACT(year FROM tdate)
```

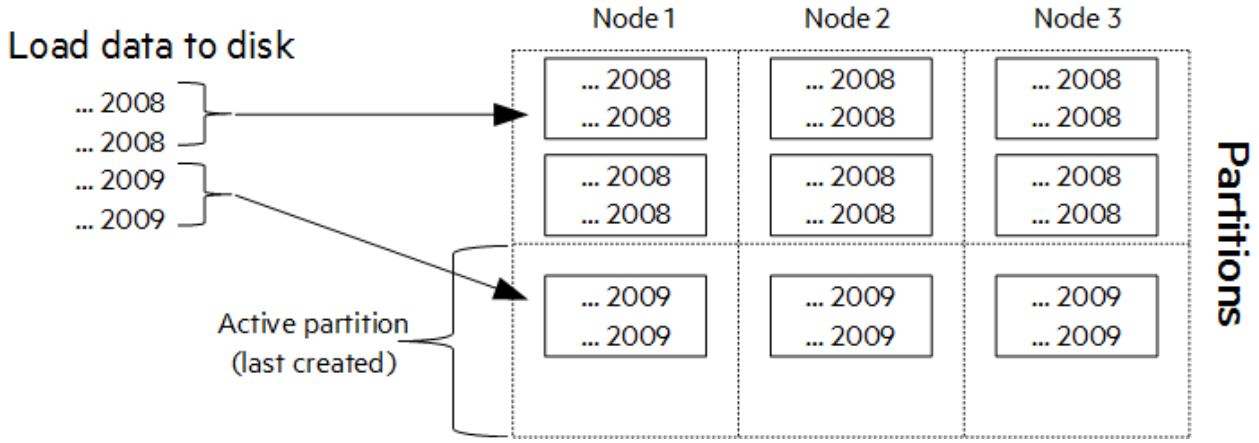
このパーティション化により、partition management functions（パーティションの管理関数）を使用して移動または削除する必要のあるデータのサブセットを簡単に管理できます。

trade テーブルの例では、2008 年のデータを最初に読み込むと仮定します.Verica はデータを ROS（read optimized store の略）コンテナに格納します。

Vertica は、2008 年のデータのすべての ROS コンテナを格納するパーティションを作成します。直近で作成されたパーティションを*アクティブパーティション*と呼びます。次の図は、パーティションにロードされた 2008 年のデータの 2 行を示しています。

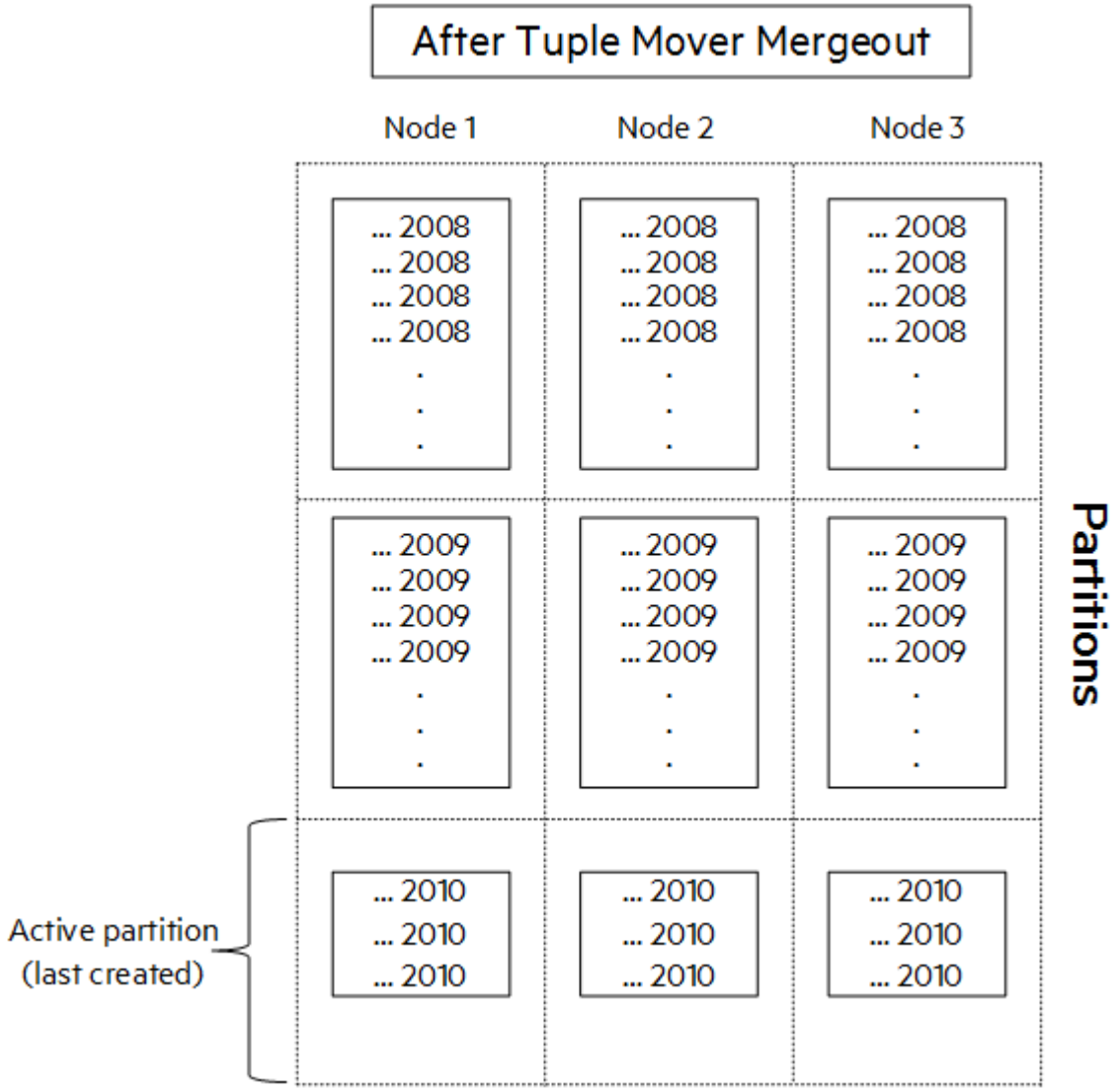


2009 年のデータを最初に読み込むと、Vertica はアクティブなパーティションになる新しいパーティションを作成します。次の図では、Vertica は 2 行の 2009 年のデータを新しいパーティションにロードします。非アクティブなパーティションには、頻繁にアクセスする必要のないデータが含まれています。



Mergeout を実行するとき、非アクティブなパーティションに対して、Tuple Mover はすべての ROS コンテナを 1 つの ROS コンテナに統合します。アクティブなパーティションでは、Tuple Mover は階層ベースのアルゴリズムを使用して ROS コンテナを統合します。

次の図は、2008 年と 2009 年のパーティションの ROS コンテナと、アクティブなパーティション内の新しくロードされたデータを示しています。



メモ

アクティブなパーティションと ROS コンテナの詳細については、この記事の後半の「パーティションと ROS ファイルと ROS コンテナ」を参照してください。

パーティショニングとセグメンテーションの違いは何ですか？

セグメンテーションは、クラスター内のノード間でデータを均等に分割し、MPP（超並列処理）アーキテクチャーを活用するのに役立ちます。

パーティショニングは、各ノード上のデータを異なるストレージコンテナに編成するのに役立ちます。パーティショニングによって I/O が削減され、クエリのパフォーマンスが向上します。

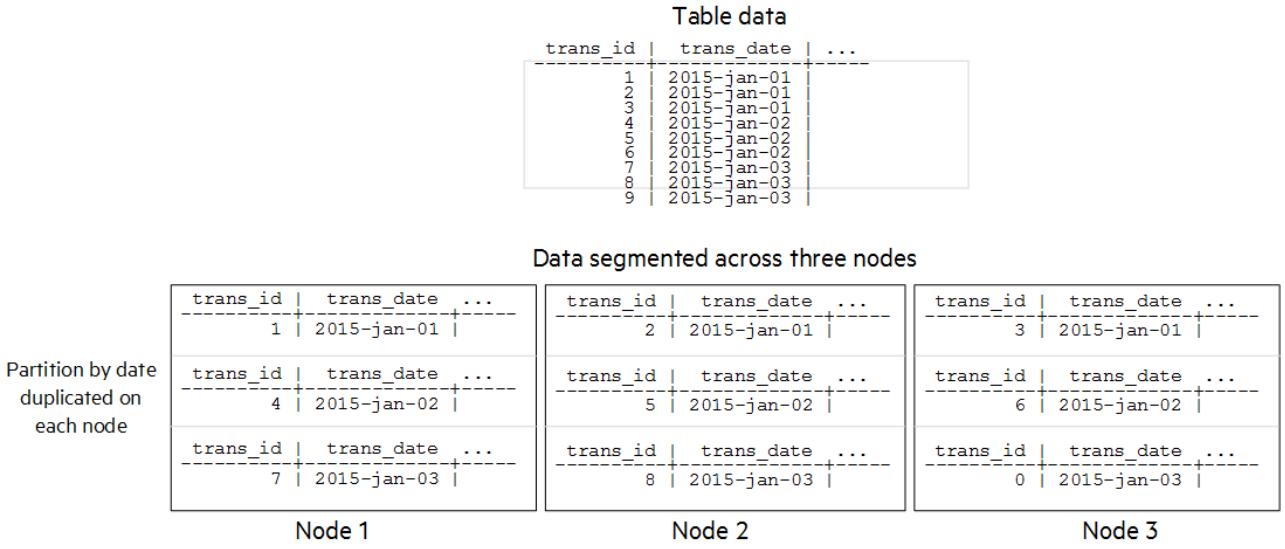
ファクトテーブルに少なくとも次の 2 つのカラムがあるとします。

- Transaction ID (trans_id)
- Transaction date (trans_date)

ファクトテーブルを作成する時に、以下を行う必要があります。

- HASH（trans_id）でテーブルのプロジェクションを分割して、すべてのノードにわたってデータを均等に分散させます。
- date でテーブルをパーティションします。

このテーブル定義では、各パーティションのデータがすべてのクラスターのノードに均等に分割されます。このセグメンテーションを使用すると、特定のパーティションを対象とするクエリを並行して実行できます。次の簡単な例は、データの配布方法を示しています。



アクティブなパーティションとは何ですか？

前述のように、テーブルの *アクティブなパーティション*は、最後に作成されたパーティションであり、最後に更新されたパーティション *ではありません*。アクティブパーティションには、頻繁にロードされているデータが含まれています。

構成パラメーターActivePartitionCount を変更することにより、テーブルごとのアクティブパーティション数を変更できます。デフォルト値は 1 です。

Tuple Mover は、非アクティブパーティションのストレージコンテナを単一の ROS コンテナに統合し、パーティションごとに 1 つの ROS コンテナを作成します。Tuple Mover は、データをマージする方法を決定するストラータアルゴリズムを使用して、アクティブなパーティションのストレージコンテナをマージします。

最後の 2 つのパーティションにデータを頻繁にロードする場合は、ActivePartitionCount を 2 に変更します。これはグローバルの構成パラメーターであり、すべてのテーブルに影響します。2 に設定すると、Tuple Mover は最後に作成された 2 つのパーティションにストラータアルゴリズムを適用します。

ActivePartitionCount を増やすと、Tuple Mover 処理の数が減ります。しかしながら、プロジェクションには、あまりに多くの ROS コンテナが残ってしまうかもしれません。

次のクエリを使用して、プロジェクションのアクティブなパーティションを検索します。

```
=> SELECT DISTINCT partition_key FROM strata
   WHERE projection_name ILIKE '%sktest%'
   AND schema_name ILIKEe '%public%';
 partition_key
---------------
             5
             8
             9
(3 rows)
```

どのテーブルをパーティション化する必要がありますか？

Vertica は、大きなファクトテーブルのみをパーティション化することを推奨します。小さなテーブルまたはディメンションテーブルをパーティション化しないでください。そうすることで、多数の ROS コンテナが作成されます。これにより、カタログのサイズが急速に増加し、クエリのパフォーマンスが低下します。

ほとんど使用されないデータをどのようにアーカイブできますか？

データを削除するのではなく、履歴の目的で長い間データを保存する必要があるかもしれません。

その場合、履歴用の古いパーティションのストレージポリシーを作成することを検討してください。古いパーティションをより低速で安価なストレージに格納することができます。このオプションを使用すると、高速で高価なストレージを解放でき、頻繁にアクセスされるパーティションをすばやく検索できます。

詳細については、Vertica ドキュメントの Creating Storage Policies（ストレージポリシーの作成）を参照してください。

テーブルのパーティションスキームを定義するときに考慮すべき点は何ですか？

テーブルのパーティション式を定義するときは、次の点を考慮してください。

- データ保持ポリシー

- 頻繁に使用されるクエリの述語
- ノード毎プロジェクション毎の ROS コンテナの数とノード毎の ROS ファイルの合計数に、パーティションの粒度が与える影響

パーティションとストレージプルーニング

パーティションはデータライフサイクル管理にどのように影響しますか？

Vertica データベースの大きなファクトテーブルをパーティション化すると、データライフサイクル管理が容易になり、クエリのパフォーマンスが向上します。

Vertica は次の機能を提供します。

- テーブルからパーティションを削除する。
- めったに使用していないパーティションをアーカイブテーブルに移動する。
- 使用頻度の低いパーティションを安価なストレージに移動する。
- アーカイブからパーティションを復元する。

Vertica では、パーティションと関連する SQL 機能を使用してデータ保持ポリシーを管理することを強く推奨します。

ストレージプルーニングとは何ですか？また、パーティション化にはどのように関連していますか？

パーティションのデータは、個々のストレージコンテナに分離されます。結果として、パーティション化されたカラムで述語を指定してクエリを実行すると、Vertica のストレージプルーニング機能のメリットを大幅に得ることができます。ストレージプルーニングがどのように機能するかについては、次を参照してください。

クエリプランニングフェーズでは、データベースのオプティマイザは、そのクエリで必要なデータを含まないストレージコンテナを識別します。オプティマイザは、この情報を各コンテナのパーティショニングカラムの最小値と最大値に基づいて得ます。クエリ処理中、Vertica 実行エンジンは、適用可能な値を格納しないストレージコンテナを省き、I/O を削減し、クエリパフォーマンスを向上させます。

月ごとにパーティション化された 3 年間のデータを持つテーブルを持っているとします。このテーブルには 36 個のデータのパーティションがあります。ここには、次の 4 つのシナリオがあるとします。

クエリ	ストレージコンテナプルーニング
クエリは、特定の週または月のデータを分析します。	Vertica は、残りの 35 のパーティションを含むストレージコンテナを読み飛ばします。
クエリは、3 か月間のデータを分析します。	Vertica は 3 つのパーティション（毎月 1 つ）に属するストレージコンテナを使用し、残りを読み飛ばします。
クエリは、パーティショニングに関係しないカラムを述語を指定しています。	このクエリでは、述語カラムのデータが複数のパーティションにわたって存在する可能性があるため、ストレージプルーニングが有効に使用されません。 **例**：テーブルは transaction_date によってパーティション化されていますが、クエリは zip_code カラムを述語に指定しています。

クエリは、パーティショニングカラムと相関関係があるカラムを述語に指定しています。	このクエリは、必要に応じてストレージプルーニングを利用します。 **例**：クエリは ship_date を述語に指定しています。通常、ship_date は、transaction_date と同じ月、またはその次の月になります。この場合、Vertica はその 2 ヶ月間の 2 つのパーティションを除くすべてのパーティションを読み飛ばすことができます。

クエリでパーティショニングとストレージプルーニングが使用されているかどうかを確認するにはどうすればよいですか？

クエリで読み飛ばされたストレージコンテナの数を確認するには、次の手順を実行します。

1. クエリをプロファイルして、transaction_id および statement_id を取得します。

```
=> PROFILE SELECT * FROM <table_name> WHERE <column_name> BETWEEN 5 AND
7;
NOTICE 4788:  Statement is being profiled
HINT:  Select * from v_monitor.execution_engine_profiles where
transaction_id=54043195528458555 and statement_id=1;
NOTICE 3557:  Initiator memory for query: [on pool general: 19543 KB,
minimum: 19543 KB]
NOTICE 5077:  Total memory required by query: [19543 KB]
 C1
----
 6
 7
 5
(3 rows)
```

2. システムテーブル QUERY_EVENTS を使ってそのトランザクションについて調べ、そのクエリが実行されたときにパーティションが処理されなくなっていることをクエリプランが示す場所を特定します。

```
=> SELECT node_name , event_details FROM query_events WHERE
   event_type = 'PARTITIONS_ELIMINATED' AND transaction_id =
54043195528458555
   AND statement_id=1;
    node_name     |                       event_details
------------------+-----------------------------------------------------
-------
v_vmart_node0003  | Using only 1 stores out of 3 for projection
public.tab_b0
v_vmart_node0002  | Using only 1 stores out of 5 for projection
public.tab_b0
v_vmart_node0001  | Using only 1 stores out of 2 for projection
public.tab_b0
(3 rows)
```

パーティションと ROS ファイルと ROS コンテナ

ROS ファイルと ROS コンテナの違いは何ですか？

ユーザーが COPY ステートメントを DIRECT で発行すると、Vertica はカラムごとに 1 つの ROS ファイルを作成します。ROS コンテナは、ROS ファイルの論理的なグループです。Vertica は、COPY DIRECT または Tuple Mover 処理の結果として ROS コンテナを作成します。

なぜ各ノードのプロジェクションあたりの ROS コンテナの数を考慮する必要がありますか？

Vertica は、パーティション化されたデータを異なるストレージコンテナに分離します。Vertica はパーティション間でデータをマージしないため、小さなパーティションが多すぎると、ストレージコンテナの数を ROS コンテナの最大数：1024 に近づく可能性が出てきます。

特定のプロジェクションに対してこの限界に達した場合、そのプロジェクションに新しいデータをロードしようとすると、「Too many ROS containers（ROS コンテナが多すぎる）」というエラーでロードが失敗します。そのエラーが発生した場合は、次のいずれかの操作を行います。

- ALTER TABLE を使用して、パーティション式を、より荒い粒度のパーティショニングスキームに変更する。
- MOVE_PARTITIONS_TO_TABLE を使用して古いパーティションをアーカイブテーブルに移動する。

ノードあたりの ROS コンテナの合計数は、どのようにデータベースに影響を与えますか？

ノードあたりの ROS ファイルの数は次のとおりです。

```
(# storage containers) x (# columns per projection)
```

ROS ファイルの数は、大きなカタログサイズの大きな要因となります。大規模なカタログを使用すると、システムメモリを消費し、システムテーブルへのクエリ、データベース起動、データベースバックアップ、およびスクラッチリカバリーなどの他のデータベース操作が遅くなります。

ノードあたり 100 万個の ROS ファイルを持つカタログがある場合、カタログのサイズは約 3-4GB です。ROS ファイルを管理する前に ROS ファイルの数が増えると、カタログメモリが解放されないことがあります。Vertica がリリースされたカタログメモリを再取得すると、そのメモリは将来の使用のために利用可能なリストになります。ただし、ノードを再起動するまで、ノードは実際にそのメモリを解放しません。

2 つのユースケースを見てみましょう。

ユースケース 1：ラージカタログ

下記のオブジェクトを持つデータベースであるとします。

- 1,000 テーブル
- テーブル毎に 2 つのプロジェクション
- テーブル毎に 50 カラム（平均で）
- ノード毎に 50 の ROS コンテナ
- ノード毎におよそ 500 万個の ROS ファイル

2000 個のプロジェクションと 5,000,000 個の ROS ファイルがあります。

```
(1000 tables) x (2 projections per table) = 2000 projections
(50 columns) x (2000 projections ) = 10,0000 x (50 ROS per projection) =
5,000,000 ROS files
```

日付ごとに 300 個のテーブルをパーティション化し、1 年間のデータを保持することができます。これにより、Vertica のパーティション管理機能を使用して、1 つの日付分のデータを管理することができます。しかし、それはカタログのサイズを 2 倍にして、ROS 数をさらに 1000 万個押し上げます。

```
(300 tables) x (2 projections) = 600 projections
(365 days) x 600 = 219,000 x (50 columns) = 10,900,000 ROS files
```

この場合、次の 2 つの方法があります。

- 日付でのパーティショニングでは、ノード毎のプロジェクション毎にに少なくとも 365 個の ROS コンテナが追加されるため、日付ごとにパーティション化されるテーブルの数を少なくします。小さなテーブルのパーティション化は避けてください。可能であれば、ほとんどのテーブルでは週または月ごとにパーティション化してください。

- 日付によるパーティショニングが必要な場合、同様のテーブルを単一のテーブルに結合して ROS コンテナの数を減らすことができる箇所を特定します。スキーマは、クエリパフォーマンスを向上させるために、テーブルを地理的な場所によって複数のテーブルに分割する従来の OLTP システムから継承されることがよくあります。Vertica では、これらのテーブルを単一のテーブルに結合することを推奨します。

ユースケース 2：プロジェクションあたりの ROS コンテナが多すぎる

下記のオブジェクトを持つデータベースであるとします。

- 100 テーブル
- テーブル毎に 2 つのプロジェクション
- テーブル毎に 50 カラム
- ノード毎に 50 の ROS コンテナ
- ノード毎におよそ 50 万個の ROS ファイル

200 個のプロジェクションと 50 万個の ROS ファイルがあります。

```
(100 tables) x (2 projections per table)      = 200 projections
(Average of 50 columns) x (200 projections)  =
 10,000 x (50 ROS containers per projection) =
 500,000 ROS files
```

10 個のテーブルを日付別にパーティション化し、3 年間データを保持したいとします。これにより、パーティション管理機能を使用して、単一の日付のデータを管理することができます。このシナリオでは、多数の ROS ファイルが蓄積されることはありません。しかしながら、3 年間経過する前に、データロード時に、ROS プッシュバック（1024 個の ROS コンテナの制限）にヒットしました。

```
1 table x (365 x 3) daily for 3 years = 1095  ROS containers for inactive
partitions
```

その結果、アクティブなパーティションにデータを受け取る余地はありません。

また、週ごとにパーティションを作成する場合は、下記のとおりとなります。

```
1 table x (52 x 3) weekly for 3 years = 156 ROS containers for inactive
partitions
```

この代替案では、ROS コンテナを大幅に少なくすることができます。この状況では、日付ではなく週ごとにデータをパーティショニングすることを検討してください。

Vertica は、将来データの拡張を計画する際に、これらの問題を考慮することを推奨します。他の分析アプリケーションのデータを Vertica に移行したり、るいは、データベースのストレージの最適化とパフォーマンスを確認した後、より多くのデータを保持することができるでしょう。

再パーティショニングと再編成

パーティション化されていないテーブルをパーティション化できますか？ テーブルのパーティション式を変更することはできますか？

既存のテーブルをパーティション化するか、次のようにしてテーブルのパーティション式を変更します。

```
=> ALTER TABLE <table_name> PARTITION BY <partition_expression>
```

このステートメントを実行すると、ストレージコンテナの既存のパーティションキー情報は直ちに消去されます。REORGANIZE キーワードを使用して、新しいパーティション式に従ってこの情報を再構築する必要があります。

注意

ノードが停止している際は、パーティショニングの変更は実行しないでください。

REORGANIZE キーワードを使用するとどうなりますか？

テーブルをパーティション化、または、再パーティション化するには、次のように、PARTITION BY と REORGANIZE キーワードを個別に使うか、あるいは、一緒に使用します。

```
=> ALTER TABLE <table_name> PARTITION BY <partition_expression>
REORGANIZE;
```

このステートメントを実行すると、Vertica は既存のパーティションキーをすべて削除し、テーブルを再パーティション化してテーブルを再編成します。

再編成処理は、バックグラウンドで実行される Tuple Mover 処理の一種です。再編成処理では、データベースのパフォーマンスに影響を与えないように、データをチャンク単位で読み取ります。次に、REORGANIZE は新しいパーティショニング方式に従って ROS コンテナにデータを書き込み、パーティションキーを ROS コンテナオブジェクトに追加します。

再編成を遅らせることはできますか？ 遅らせることによる影響は考えられますか？

実行中のデータベースのパフォーマンスを最小限に抑えるために、再編成は一度に 1 つの ROS コンテナのサブセットでのみ動作します。

再パーティション化後に再編成処理を遅らせると、次のような制限が生じます。

- パーティション式が変更されていても再編成されていないテーブルでは、パーティション関数を実行できない。
- パーティション化されたテーブルの場合、パーティションキーのない ROS コンテナは、Tuper Mover の mergeout に参加しません。これにより、ROS プッシュバックが発生する可能性がでてくる。

Vertica は、再パーティショニングを実行した後、できるだけ早く再編成することを推奨します。変更されたテーブルにひもづけられているすべてのプロジェクションに対して再編成が完了するまで、再編成処理の進行をモニターしてください。

どのように REORGANIZE 処理のステータスをモニタリングできますか？

次のシステムテーブルを使用して、再編成処理のバックグラウンドでの進行状況をモニタリングします。

- VS_TUPLE_MOVER_OPERATIONS
- PARTITION_STATUS
- PARTITION_REORGANIZE_ERRORS

特定のテーブルのパーティションの履歴を確認するには、システムテーブル CATALOG_EVENTS の内容を確認します。

テーブルのパーティショニングを削除するにはどうすればよいですか？

テーブルがパーティション化されなくなるように変更するには、次のステートメントを使用します。

```
=> ALTER TABLE <table_name> REMOVE PARTITIONING;
```

パーティショニングをテーブルから削除すると、Vertica はテーブルを他のパーティション化されていないテーブルと同様に扱います。Vertica は、ストラータアルゴリズムを使用して ROS コンテナをマージします。

パーティションの制限における考慮事項

テーブルあたりのパーティション数に制限はありますか？

テーブルあたりのパーティション数に制限はありません。ただし、パーティションのデータは ROS コンテナに分離されています。Vertica は、1 ノード毎のプロジェクション毎の ROS コンテナ数を 1024 に制限しているため、制限は本質的に 1 テーブルあたり 1024 パーティションです。

Vertica は、1 つの COPY DIRECT ステートメントが 1024 以上のパーティションをロードできないようにしています。

365 を超えるパーティション（制限の約 3 分の 1）がある場合、プロジェクションごとの ROS コンテナ数を監視し、Tuple Mover の mergeout 処理を監視します。365 を超えるパーティションでは、そのテーブルのパーティショニングスキームを再考するか、または、クエリが発行されていないパーティションをアーカイブテーブルに移動する必要があります。

メモリ内のデータベースカタログのサイズを確認するにはどうすればよいですか？

次のクエリは、データベースカタログのサイズを返します。

```
=> SELECT node_name, MAX (ts) AS ts, MAX(catalog_size_in_MB)
   AS catlog_size_in_MB
   FROM
   (SELECT node_name,
    TRUNC((dc_allocation_pool_statistics_by_second."time")::TIMESTAMP,
    'SS'::VARCHAR(2)) AS ts,
    SUM((dc_allocation_pool_statistics_by_second.total_memory_max_value
```

```
    -
dc_allocation_pool_statistics_by_second.free_memory_min_value))/(1024*102
4)
    AS catalog_size_in_MB from dc_allocation_pool_statistics_by_second
GROUP BY 1,
    TRUNC((dc_allocation_pool_statistics_by_second."time")::TIMESTAMP,
    'SS'::VARCHAR(2))
    )
    subquery_1 GROUP BY 1 ORDER BY 1 LIMIT 50;
```

パーティショニングすることにより、カタログサイズが大きくなっているかどうかを確認するにはどうすればよいですか？

データベースのカタログサイズが大きくなってしまう原因となる状況はいくつか考えられます。ここで、大きなカタログとは、1 ノードあたり 10 GB 以上のカタログを指します。

次のクエリは、ROS コンテナを最も多く持つ上位 50 個のプロジェクションを返します。結果として得られたテーブルがパーティション化されていて、パーティション数が多い場合は、Vertica がより少ないパーティションを作成できるようにパーティション化スキームを変更することを検討してください。

```
=> SELECT s.node_name, p.table_schema, s.projection_name,
   COUNT(DISTINCT s.storage_oid) storage_container_count,
   COUNT(DISTINCT partition_key) partition_count,
   COUNT(r.rosid) ros_file_count
   FROM storage_containers s LEFT OUTER JOIN PARTITIONS p
   ON s.storage_oid = p.ros_id JOIN vs_ros r
   ON r.delid = s.storage_oid
   GROUP BY 1,2,3 ORDER BY 4 DESC LIMIT 50;
```

詳細情報

パーティショニングの詳細については、Vertica ドキュメントの次のセクションを参照してください。

- Using Table Partitions
- Partition Management Functions