データインポートとエクスポート

March, 2017

原文はこちら

MICRO FOCUS

目次

Vertica には、2 つの Vertica クラスタ間でデータを移動できるインポートおよびエクスポート機能があります。クラスタ間でのデータのインポートとエクスポートは、STDIN を介したデータのストリーミングや、本書で説明されている理由での vsql 接続の使用よりも高速です。

インポートとエクスポートの概要

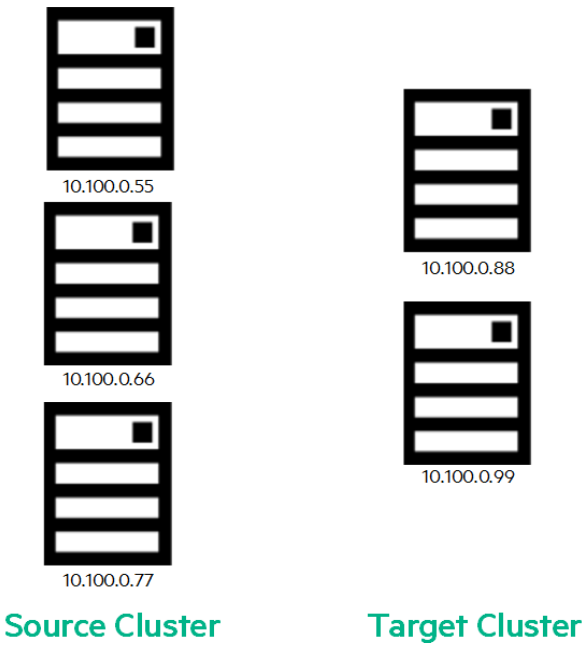
インポートおよびエクスポートプロセスは、イメージをミラーリングするように動作します。データをエクスポートするクラスタは、SELECT 文に相当するものを実行します。データをインポートするクラスタは、COPY 文に相当するものを実行します。インポート/エクスポート操作の仕組みは、INSERT ... SELECT ...クエリに似ています。

このドキュメントでは、データエクスポートするクラスタをソースクラスタ（source cluster）、データインポートするクラスタをターゲットクラスタ（target cluster）と表します。データは、ソースクラスタ内のテーブルから、ターゲットクラスタ内の同様のテーブルに移動します。これらのクラスタとテーブルは、次のような異なるトポロジとその他の特性を持つことがあります。

- ノードの数
- データベースのバージョン
- プロジェクションの数
- 列のデータ型
- セグメンテーション
- ユーザー
- DB パラメータ設定

インポートとエクスポートの動作を説明するために、このドキュメントの例では 2 つの Vertica クラスタを使用しています。

- ソースクラスタ：3 ノード (10.100.0.55, 10.100.0.66, 10.100.0.77)
- ターゲットクラスタ：2 ノード (10.100.0.88, 10.100.0.99)



エクスポートはどのように動作するのか？

ソースクラスタでは、通常、エクスポート操作は次のコマンドとして実行されます。まず、ソースデータベースがターゲットデータベースに接続します。次に、ソースは EXPORT TO VERTICA クエリを実行します。このクエリは、ターゲットにエクスポートするデータを取得する SELECT 文を指定します。

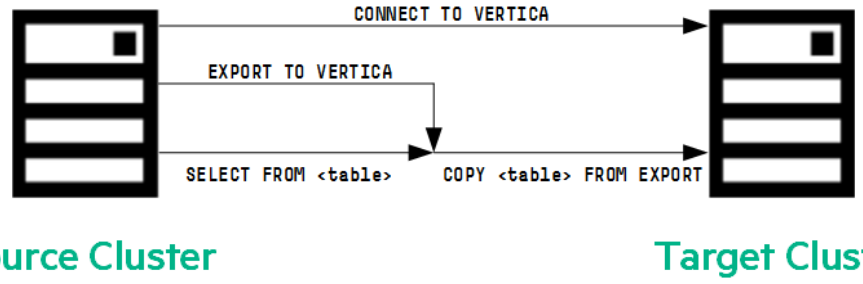
```
source=> CONNECT TO VERTICA VerticaDBTarget USER dbadmin PASSWORD '' ON
'10.100.0.88',5433;
source=> EXPORT TO VERTICA VerticaDBTarget.tgt_table (n,a,b) AS
   SELECT n AS col1, a as col2 , b as col3 from src_table;
```

ソースは上記の EXPORT 文で SELECT 文を実行して、エクスポートするデータを取得します。ターゲットは、ソースに接続する COPY ステートメントを作成し、SELECT ステートメントの出力を受け取ります。

この COPY 文には、ターゲットクラスタがソースクラスタに接続してデータをストリーミングするために必要な情報が含まれています。この情報には、IP アドレス、ポート番号、データ型と列名、およびエンコードおよび圧縮情報が含まれます。

COPY 文は、ソースで実行される SELECT 文の出力をコピーし、そのデータをターゲットに格納します。

次の図は、エクスポートのプロセスを示しています。



ターゲット・データベースで実行される正確な COPY ステートメントを表示するには、SESSIONS システムテーブルを照会します。複数の並行 TCP ストリームがデータをロードしているため、この COPY 文は vsql を使用するか STDIN からデータをコピーするよりも高速に実行されます。

```
target=> SELECT user_name, node_name, current_statement FROM sessions;
 user_name |          node_name          |
-----------+-----------------------------+-------------------------------
--------------------
 dbadmin   | v_VerticaDBTarget_node0001 | COPY tgt FROM EXPORT
':SendExport explainBits:0 ...
(1 rows)
```

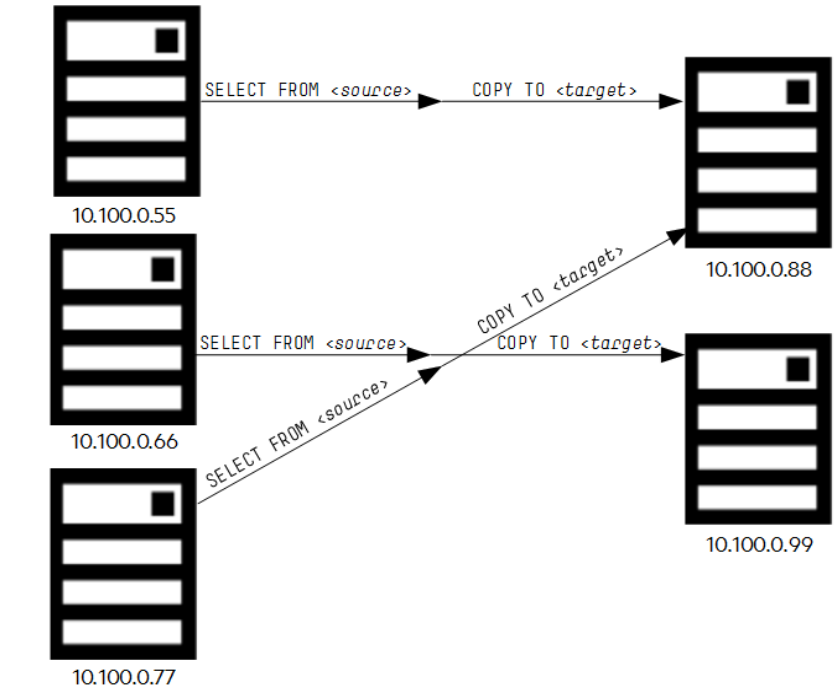
ソースノード IP:5434 ポートへの初期接続が行われ、ソース側の一時ポートに切り替えられます。

例:

- v_VerticaDBTarget_node0001 は、10.100.0.55（送信元ノード 1）と 10.100.0.77（送信元ノード 3）の両方からデータを受信しています。
- v_VerticaDBTarget_node0002 が、10.100.0.66（送信元ノード 2）からデータを受信しています。

v_VerticaDBTarget_node0001 は、v_VerticaDBTarget_node0002 よりも 2 倍の作業をしています。ソースノードとターゲットノードが一致していないと、インポート/エクスポート操作に時間がかかることがあります。

```
COPY tgt ( n, a, b )
FROM EXPORT
   ':SendExport explainBits:0 planNumber:45035996273709640 tag:1000
status:
   :DataPort ip_source:2 oid:45035996273704982
      name:v_VerticaDBTarget_node0001 ip:10.100.0.55 address_family:0
port:5434 .
    ports: { :DataPort ip_source:2 oid:45035996273704982
                name:v_VerticaDBTarget_node0001 ip:10.100.0.55
address_family:0 port:5434 .
             :DataPort ip_source:2 oid:45035996273721216
                name:v_VerticaDBTarget_node0002 ip:10.100.0.66
address_family:0 port:5434 .
             :DataPort ip_source:2 oid:45035996273721220
                name:v_VerticaDBTarget_node0003 ip:10.100.0.77
address_family:0 port:5434 . }
    db:VerticaDBSource
    table:tgt colnames: { :string _:n . :string _:a . :string _:b . }
              columns:  { :DataType oid:9 type:6 len:3 typmod:7 .
                          :DataType oid:9 type:6 len:4997 typmod:5001 .
                          :DataType oid:9 type:6 len:4997 typmod:5001 . }
    rle: { :vbool _:0 . :vbool _:0 . :vbool _:0 . }
    isCompressed:0 . '
```



netstat コマンドは、ターゲットシステム 10.100.0.88 と 10.100.0.99 上の受信キューが積極的にデータを受信していることを示しています。10.100.0.88 は 10.100.0.55 と 10.100.0.77 からデータを受信し、10.100.0.99 は 10.100.0.66 からデータを受信しています。

```
[ dbadmin@ip-10-100-0-88 ~]$ netstat -tna -p $(pgrep vertica)  | grep
ESTABLISHED | grep ":5434"
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp    476767      0 10.100.0.88:5434        10.100.0.55:46756
ESTABLISHED 6834/vertica
tcp    457636      0 10.100.0.88:5434        10.100.0.77:59568
ESTABLISHED 6834/vertica
[ dbadmin@ip-10-100-0-99 ~]$ netstat -tna -p $(pgrep vertica)  | grep
ESTABLISHED | grep ":5434"
tcp    480539      0 10.100.0.99:5434        10.100.0.66:59074
ESTABLISHED 6359/vertica
```

Vertica では、ソースとターゲットクラスタのノード数が同じであることを推奨しています。そうすることで、ロードのバランスをとって、単一のノードがエクスポート操作を保留できないようにします。ノード数も同じであれば、データ転送速度を向上させることができます。

ソースとターゲットクラスタに異なる数のノードがある場合、複数のソースが 1 つのターゲットにデータを送信しているか、1 つのソースが複数のターゲットにデータを送信している可能性があります。

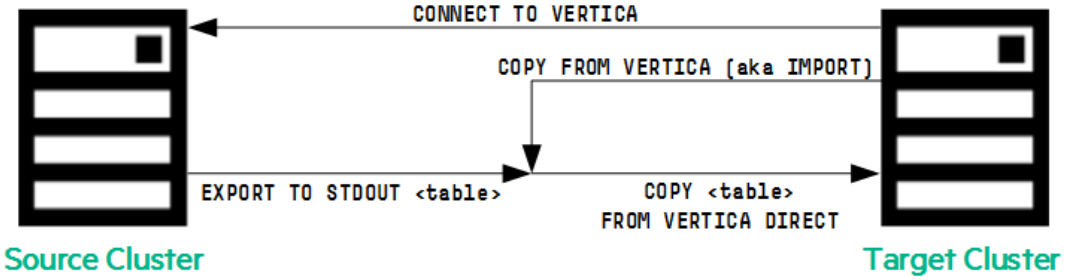
インポートはどのように動作するのか？

ターゲットクラスタでは、通常、インポート操作は次のコマンドとして実行されます。

```
target=> CONNECT TO VERTICA VerticaDBSource USER dbadmin PASSWORD '' ON
'10.100.0.55',5433;
CONNECT
target=> COPY tgt(n,a,b) FROM VERTICA VerticaDBSource.src(n,a,b) DIRECT;
```

CONNECT コマンドは、ターゲット・クラスターからソース・クラスターに接続します。COPY コマンドは、インポートするデータを指定します。

次の図は、インポートのプロセスを示しています。



ソース・データベースで実行される EXPORT 文を表示するには、SESSIONS システムテーブルを問い合せます。

```
source=> SELECT node_name, current_statement, last_statement FROM
sessions;
      node_name       |      current_statement                  |
last_statement
----------------------+-----------------------------------------+--------
--------
 v_verticadb_node0001 | export to STDOUT FROM src ( n , a , b )
```

ネットワーク上のデータ圧縮

通常、ネットワークはインポートまたはエクスポート操作のボトルネックではありません。ただし、低速または低帯域幅のネットワークでは、ソースクラスタとターゲットクラスタの両方でネットワーク上のデータ圧縮を有効にすることで、ソースクラスタとターゲットクラスタ間のデータ転送を高速化できます。

ネットワーク設定が同じでない場合は、次のエラーが表示されます。

```
target=> COPY tgt(n,a,b) FROM  VERTICA verticadb.src(n,a,b)  DIRECT;
ERROR 5520:  verticadb compresses network traffic. verticadb2
does NOT compress network traffic. Please change the configuration
to be consistent
HINT:  Configuration can be changed using set_config_parameter() function
```

Vertica は、ソースノードとターゲットノードの両方でネットワーク上のデータ圧縮を有効にすることを推奨しています。これを行うには、DB パラメーター「CompressNetworkData」を 1 に設定します。

```
=> SELECT SET_CONFIG_PARAMETER('CompressNetworkData',1);
SET_CONFIG_PARAMETER
----------------------------
Parameter set successfully
(1 row)
```

インポートおよびエクスポート操作の進捗状況の確認

インポート/エクスポート操作の進行状況は、ターゲット上の LOAD_STREAMS システム・テーブルを使用して監視できます。parse_complete_percent フィールドは空になっています。つまり、ターゲットクラスタでデータを解析する必要はありません。

```
target=> SELECT
read_bytes,parse_complete_percent,unsorted_row_count,sorted_row_count
         FROM load_streams WHERE is_executing;
 read bytes | parse complete percent | unsorted row count |
sorted_row_count
------------+-----------------------+--------------------+--------------
----
          0 |                       |           39645696 |
19559826
(1 row)
```

ファイルを解析する必要がある COPY 文を使用して CSV ファイルをロードするのとは異なり、IMPORT 操作は、Vertica がターゲット・クラスタで解析する必要のない COPY を意味します。ソースはデータをタプルとしてターゲットにストリームします。

具体的には、ターゲット上のカウンタで受け取った行を調べて進捗状況を監視します。

```
target=> SELECT node_name, counter_name, counter_value, operator_name
         from execution_engine_profiles WHERE is_executing='t'
         AND counter_name IN ('rows received') AND Operator_name IN
('Import');
          node_name          | counter_name  | counter_value |
operator_name
-----------------------------+---------------+---------------+------------
 v_VerticaDBTarget_node0001 | rows received |     124257398 | Import
 v_VerticaDBTarget_node0001 | rows received |     123197558 | Import
 v_VerticaDBTarget_node0002 | rows received |     237063416 | Import
(3 rows)
```

ソース上のプロセスを監視するには、カウンタを送信した行を確認します。

```
source=> SELECT node_name, counter_name, counter_value,operator_name
         FROM execution_engine_profiles WHERE is_executing='t' a
         AND counter_name in ('rows sent') AND Operator_name IN
('Export');
          node_name          | counter_name | counter_value |
operator_name
-----------------------------+--------------+---------------+-------------
 v_VerticaDBSource_node0001 | rows sent    |      78890185 | Export
 v_VerticaDBSource_node0002 | rows sent    |     126071726 | Export
 v_VerticaDBSource_node0003 | rows sent    |      78889767 | Export
(3 rows)
```

増分エクスポート

ソースクラスタ内のテーブルにデータを継続的に追加し、ターゲットクラスタ内のターゲットテーブルにデータをコピーする場合は、増分エクスポートを実行できます。増分エクスポートでは、エポック述部または日付述部などの述部を含むクエリを使用して、最近追加されたデータのみをターゲットにプッシュします。

次の例では、39 は最後の成功したエクスポートが行われたエポックです。次の例の EXPORT ステートメントは、エポック 39 の後にソースにロードされたデータのみをエクスポートします。インポート操作では、同様の構文はサポートされていません。

```
source=> COPY cluster1_table1 FROM STDIN DIRECT;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself.
>> 100
>> 101
>> 102
>> 103
>> \.
source=> SELECT epoch, * FROM cluster1_table1;
 epoch |  i
-------+-----
    39 |   2
    39 |   3
    39 |   5
    40 | 101
    40 | 102
    39 |   1
    39 |   4
```

```
    40 | 100
    40 | 103
(9 rows)
source=> CONNECT TO VERTICA VerticaDBTarget USER dbadmin PASSWORD '' ON
'10.100.0.88',5433;
 CONNECT
source=> EXPORT TO VERTICA VerticaDBTarget.cluster2_table2 AS SELECT *
FROM public.cluster1_table1
         WHERE epoch > 39;
 Rows Exported
---------------
             4
(1 row)
```

並列エクスポート

大量のデータをエクスポートする場合は、複数のエクスポートを並行して実行することで、エクスポート操作の速度を向上させることができます。

```
=> EXPORT TO VERTICA VerticaDBTarget.tgt (n,a,b) AS
   SELECT n as col1, a as col2 , b as col3 FROM src
   WHERE epoch > 0 AND epoch <= 1;

=> EXPORT TO VERTICA VerticaDBTarget.tgt (n,a,b) AS
   SELECT n as col1, a as col2 , b as col3 FROM src
   WHERE epoch > 1 AND epoch <= 2;

=> EXPORT TO VERTICA VerticaDBTarget.tgt (n,a,b) AS
   SELECT n as col1, a as col2 , b as col3 FROM src
   WHERE epoch > 2 AND epoch <= 3;

=> EXPORT TO VERTICA VerticaDBTarget.tgt (n,a,b) AS
   SELECT n as col1, a as col2 , b as col3 FROM src
   WHERE epoch > 3 AND epoch <= 4;
```

エポック範囲（または日付述部などの他の述部）を選択して、行数がデータサイズとほぼ同じになるようにします。

インポートとエクスポートのためのネットワーク接続の設定

Vertica は、EXPORT TO VERTICA ステートメントと COPY FROM VERTICA ステートメントを使用して、1 つの Vertica クラスターから別の Vertica クラスターにデータをインポートし、プライベートネットワークを介してエク

スポートします。デフォルトでは、クラスタはデータのインポートとエクスポートにプライベートネットワークを使用します。

パブリックネットワークを使用するには、エクスポートアドレスを変更してシステムを設定する必要があります。各サーバーには、1 つのネットワーク構成しか持てません。次の方法でパブリックネットワークを使用するようにシステムを設定します。

- パブリックネットワーク上のノードまたはクラスタの IP アドレスを識別する。
- インポートまたはエクスポートのためのデータベースまたは個々のノード情報を設定する。

構成手順の詳細については、Configuring Network to Import and Export Data をご参照ください。

シーケンス、アイデンティティ、列のデフォルト値について

エクスポート操作は、データを INSERT する操作に似ています。ただし、ソースから直接データをエクスポートする場合、Vertica はターゲット上でシーケンスと列のデフォルト値を生成しません。

test_seq という名前のテーブルに次のシーケンスがあるとします。

```
source=> CREATE SEQUENCE seqinc START 101 MAXVALUE 1000 CACHE 7 CYCLE;
source=> CREATE TABLE test_seq (col_seq INT DEFAULT
NEXTVAL('seqinc'),data VARCHAR(100)) ;
source=> COPY test_seq (data) FROM STDIN;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself.
>> AAAA
>> BBB
>> CCCC
>> AAAAA
>> BBBBB
>> CCCCC
>> DDDDD
>> EEEEE
>> FFFFF
>> \.
source=> SELECT * FROM source;
col_seq | data
---------+-------
     101 | AAAA
     102 | BBB
     103 | CCCC
     104 | AAAAA
```

```
     105 | BBBBB
     106 | CCCCC
     107 | DDDDD
     108 | EEEEE
     109 | FFFFF
(9 rows)
```

このデータをターゲットにエクスポートするには、ターゲット上で次のコマンドを実行します。

```
target=> CREATE SEQUENCE seqinc START 10000
         MAXVALUE 100000 CACHE 9 CYCLE;
target=> CREATE TABLE tgt(col_seq INT DEFAULT
         NEXTVAL('seqinc'), data VARCHAR(100)) ;
```

ソース上のデータをインポートするには、ソース上で次のコマンドを実行します。

```
source=> CONNECT TO VERTICA targetdb USER dbadmin
         PASSWORD '' ON '10.100.0.77',5433;
source=> EXPORT TO VERTICA targetdb.tgt AS SELECT * from test_seq;
 Rows Exported
---------------
             9
(1 row)
```

SELECT * を指定したため、ターゲットはソースが生成したシーケンス値を取得します。

```
target=> SELECT * FROM tgt;
 col_seq | data
---------+-------
     101 | AAAA
     102 | BBB
     103 | CCCC
     104 | AAAAA
     105 | BBBBB
     106 | CCCCC
     107 | DDDDD
     108 | EEEEE
     109 | FFFFF
(9 rows)
```

ターゲット表でデフォルト値（シーケンス、アイデンティティなど）を有効にする場合は、列データの値のみを選択するようにエクスポートを構成します。シーケンス列を省略すると、ターゲットはシーケンスとアイデンティティ値を生成します。

```
source=> CONNECT TO VERTICA targetdb USER dbadmin
         PASSWORD '' ON '10.100.0.77',5433;
source=> EXPORT TO VERTICA test1.tgt(data) AS
         SELECT data FROM test_seq;
 Rows Exported
---------------
             9
(1 row)
```

ターゲット上のシーケンスは必ずしもソースのシーケンス順序に従わないことがわかります。

```
target=> SELECT * FROM tgt;
 col_seq | data
---------+-------
   10012 | BBBBB
   10013 | CCCCC
   10015 | EEEEE
   10000 | AAAA
   10009 | BBB
   10010 | CCCC
   10011 | AAAAA
   10014 | DDDDD
   10016 | FFFFF
(9 rows)
```

この方法を使用すると、シーケンスとアイデンティティの値をソースからターゲットにエクスポートできます。あるいは、EXPORT / IMPORT ステートメントから列を省略して、ターゲットにシーケンス列とアイデンティティ列を生成させることもできます。

オブジェクトレベルのバックアップ・リストア

Vertica 7.2.x 以降のバージョンでは、オブジェクトレベルの新しいバックアップ/リストア機能を提供します。この機能を使用すると、バックアップ全体をリストアせずに、これらのオブジェクトを含む任意のバックアップから個々のテーブルまたはスキーマをリストアすることができます。

ソースとターゲットの一致するノード数と Vertica サーバーのバージョンが同じ場合、オブジェクトレベルのバックアップ/リストアは、実行およびインポート/エクスポート操作よりもはるかに高速です。ただし、ソースノードの数がターゲットノードの数と等しくない場合、vbr.py スクリプトは使用できません。

Vertica は、ソースノードとターゲットノードの数が一致する場合に、オブジェクトレベルのバックアップとリストアを使用することを推奨しています。

詳細情報

インポートおよびエクスポートの詳細については、Vertica ドキュメントの次のトピックを参照してください。

- Exporting Data
- Copying Data