HP Vertica Analytics Platform 6.1.x

Best Practices for HP Vertica OEM Customers

Doc Revision 3

Date of Publication: Monday, September 23, 2013

Contents

Best Practices for HP Vertica OEM Customers

Welcome to Best Practices for HP Vertica OEM Customers. This guide contains best practices and tips for HP Vertica OEM customers who embed their HP Vertica database with their own application and deliver it as an on-the-premises solution. Typically, the end users of the application do not interact with the HP Vertica database directly but rather through the application.

The information in this guide describes technical issues for two types of tasks that OEM customers must perform to optimize the database in their application:

Developing and testing the HP Vertica database:

- Running Database Designer and designing projections
- Managing your system resources
- Managing data load
- Optimizing delete and update operations
- Dropping partitions to save disk space

Packaging the HP Vertica database with your application:

- Installing HP Vertica silently
- Configuring the HP Vertica database using parameters defined during the development phase
- Using the Administration Tools command-line interface to create, start, and connect to the HP Vertica database

Part 1: Developing and Testing the HP Vertica Database to Work with a Customer Application

During your application development, install an HP Vertica database and load it with sample data so that you can simulate and test the behavior of the database and application in a production-like environment.

The following sections describe the steps you need to take and the HP Vertica features that you need to focus on for the HP Vertica database to work as desired with your application at a customer site.

Running HP Vertica on Multiple Nodes

OEM deployments with small data sets or light workloads may be able to run the HP Vertica database along with your application on a single node. If you run the database and application on a single node, you cannot run with K-safety and you risk data loss because the database cannot be replicated.

If you have the resources to run your HP Vertica database with your application in a multi-node environment, your application should run on a different node than the   database.

If you want to run the database on a server shared with your application, refer to ***Managing Database Resources*** (page 6). That section describes the best practices for this scenario.   If you want to run the HP Vertica software on dedicated servers, general resource management best practices apply.

Installing the HP Vertica Database

To install the HP Vertica database on your development system, follow the instructions in Installing HP Vertica.

Identifying Typical Queries

Before you run Database Designer, identify common queries that your application uses. If you supply these queries to Database Designer before running it, Database Designer uses those queries to create projections that maximize performance throughput in your database.

Loading Representative Data

To effectively test the way the   database interacts with your application in a production environment, load data that is representative of the type of data that might be used in a production environment. This allows you to effectively test database functionality and performance.

For more information about loading data into an   database, see Bulk Loading Data.

Optimizing the Database with Database Designer

A HP Vertica database stores its data in a physical format called *projections*. For optimal performance, projections should be optimized based on data and workload characteristics.

Most HP Vertica OEM customers optimize the projection design during their development cycle and then standardize on that design for all deployments of their application.

This section describes the process of tuning projections using Database Designer.   ***Exporting the Optimized Database*** (page 5) contains more information on how to ensure that the optimized projections are deployed when the database is embedded with your application.

After you create your database for the first time and load the typical queries and sample data on your development system, run Database Designer using the Administration Tools, as described in one of the following sections of the   documentation:

- Creating a Comprehensive Design Using Database Designer
- Creating a Query-Specific Design Using Database Designer

What Does Database Designer Do?

Database Designer uses sophisticated strategies to create designs that provide excellent ad-hoc query performance while using disk space efficiently.

Database Designer can:

- Create a comprehensive design, which drops all existing projections and creates a new set of projections for all tables in your database.
- Create a query-specific design, which supplements your current design by creating new projections for all tables that are referenced in the queries that you supply to Database Designer. Database Designer accepts up to 100 sample queries in the query input file for a query-specific design.

Results of Running Database Designer

In most cases, the projections that Database Designer creates provide excellent query performance within physical constraints while using disk space efficiently. Database Designer:

- Recommends buddy projections with the same sort order, which can significantly improve load, recovery, and site node performance. By default, Database Designer recommends both super and non-super segmented projections with a buddy of the same sort order.
- Automatically rebalances data after you add or remove nodes.
- Produces higher quality designs by considering UPDATE and DELETE statements.

Database Designer creates two scripts:

- Design scripts—Contains all the DDL statements that are needed to create the design.
- Deployment scripts—Contains all the CREATE PROJECTION scripts that are needed to create the projections for the design. The deployment script is a subset of the design script.

Exporting the Optimized Database (page 5) describes how to configure these scripts so that all customers can be standardized on the optimal projections and to ensure that the scripts are portable to deployments with different numbers of nodes and K-safety levels.

Exporting the Optimized Database

Once you have optimized the database using Database Designer and other techniques, you need to ensure that the optimized projections are available for all customers in the deployed databases.

The following HP Vertica functions create scripts that create the database on the customer's cluster:

- `export_catalog`—Generates a SQL script that can recreate a physical schema design in its current state on a different cluster.
- `export_objects`—Generates a SQL script that can recreate catalog objects (only the non-virtual objects to which you have access) on a different cluster. Running the generated SQL script on another cluster creates all referenced objects before their dependent objects.

- `export_tables`—Generates a SQL script that can recreate a logical schema (schemas, tables, constraints, and views) on a different cluster.

Once these scripts have been created, modify the CREATE PROJECTION statements as follows:

- Modify the CREATE PROJECTION statements to include the appropriate segmentation clauses.
- Modify the CREATE PROJECTION statements to specify ALL NODES KSAFE <*value*> so that all unsegmented projections are replicated on all nodes, with the designated K-safety value. If you do not specify a K-safety value, HP Vertica uses the current K-safety value, or 1.

Managing Database Resources

For large sets of complex workloads, Hewlett-Packard recommends that you run the HP Vertica database software on dedicated servers.

If, however, you plan to have the database running on a shared server with your application, configure CPU and memory resources for both the database and your application to ensure that neither application can monopolize either CPU or memory from the other. Consider the following tips to manage resources between them.

Configuring CPU Resources

You can configure CPU resources in two ways:

- ***Pinning HP Vertica processes to certain CPUs*** (page 6)
- ***Setting run-time priorities for resource pools*** (page 7)

Pinning HP Vertica Processes to Certain CPUs

To pin an entire HP Vertica server process and its child processes to a specified set of CPUs in the cluster, use the following configuration parameters. These parameters do not affect external processes and applications:

- PINPROCESSORS: Number of CPUs for   processes to be pinned to.
- PINPROCESSORSOFFSET: Offset for the start of CPUs that the HP Vertica server will use.

For example, for a cluster where each node has 24 CPUs, numbered 0 through 23, the server process can be restricted to CPUs #16–23 by entering the following statements:

- `SELECT SET_CONFIGURATION_PARAMETER('PINPROCESSORS',8)`
- `SELECT SET_CONFIGURATION_PARAMETER('PINPROCESSORSOFFSET',16)`

Pin your application to the processors that are not allocated to the HP Vertica database. To make sure that other applications do not use the processors assigned to the HP Vertica server, use the Linux `cgroups` command.

Setting Run-Time Priorities for Resource Pools

For each resource pool, you can manage resources that are assigned to queries that are already running. Assign each resource pool a run-time priority of HIGH, MEDIUM, or LOW. These settings determine the amount of run-time resources (such as CPU and I/O bandwidth) assigned to queries in the resource pool when they run. Queries in a resource pool with a HIGH priority are assigned greater run-time resources than those in resource pools with MEDIUM or LOW run-time priorities.

Configuring Memory Resources

Set the MAXMEMORYSIZE parameter on the GENERAL pool to limit the amount of overall system memory that you allocate to the HP Vertica database.

To ensure that there is no swapping, make sure your application is limited to the memory that the database is not using. For example, if the MAXMEMORYSIZE for the GENERAL pool for the database is set to 32 GB on a system with 48 GB of RAM, your application should use no more than (48–32) = 16 GB of RAM.

Dropping Partitions Due to Space Constraints

If your hardware has fixed disk space, you may need to configure a regular process to roll out old data by dropping partitions.

For example, if you have only enough space to store data for a fixed number of days, configure HP Vertica to drop the partition with the oldest date. To do this, create a time-based job scheduler like cron to drop the partition on a regular basis during low-load periods.

If the ingest rate for data has peaks and valleys, you can use two techniques to manage how you drop partitions:

- Set up a process to check the disk space on a regular (daily) basis. If the percentage of used disk space exceeds a certain threshold, like 80%, drop the oldest partition.
- Add an artificial column in a partition that increments based on a metric like row count. For example, it increments each time that the row count increases by 100 rows. Set up a process to query that column on a regular (daily) basis. If the value in the new column exceeds a certain threshold, for example, 100, drop the oldest partition and set the column value back to 0.

For more information about partitions, see the following sections in the HP Vertica documentation:

- CREATE TABLE
- Working with Table Partitions

Loading Data into the Database

While your application is running, you need to load data into your HP Vertica database on a regular basis. You can collect large amounts of data to load into your database at once, as long as you can meet your latency service-level agreement (SLA).

Simulate your data-loading operations in a production environment, being sure to replicate the maximum expected rates of loading and spiking.

The following sections describe how to load data into your HP Vertica database and how you can tune the processes and monitor the system to prevent performance problems.

How Loading Works

When you load data into a database, HP Vertica first moves it into memory (WOS) by default. (You can override this default, and specify that HP Vertica move the data directly to disk (ROS). The Tuple Mover runs in the background and periodically

- Moves data from the WOS to the ROS
- Combines like-size ROS containers
- Deletes purges records

If you move too much data too frequently to the WOS, the data can spill to disk. No data is lost, but this can cause performance problems.

WOS and ROS

Write Optimized Store (WOS) is a memory-resident data structure for storing INSERT, UPDATE, DELETE, and COPY (without DIRECT hint) actions. To support very fast data load speeds, the WOS stores records without data compression or indexing. The WOS organizes data by epoch and holds both committed and uncommitted transaction data.

Read Optimized Store (ROS) is a highly optimized, read-oriented, disk storage structure. The ROS makes heavy use of compression and indexing. You can use the COPY...DIRECT and INSERT (with /*+direct*/ hint) statements to load data directly into the ROS.

The Tuple Mover (TM) is the HP Vertica database optimizer component that moves data from memory (WOS) to disk (ROS). The Tuple Mover runs in the background, performing some tasks automatically (ATM) at time intervals determined by its configuration parameters.

For information about the WOS, ROS, and Tuple Mover, see Hybrid Storage Model in the Concepts Guide.

Loading Data from a File

If the data to be loaded is stored in a file, use the COPY statement to load the data into your database. You can also use COPY to load data from a data stream:

In its basic form, use COPY as follows:

```
COPY to_table FROM data_source
```

For more information, see

- Using COPY and COPY LOCAL
- COPY
- COPY LOCAL

Loading Data from Your Application

If your application needs to push data into the HP Vertica database, use the driver associated with the programming language your application uses. For more information, see

- Loading Data through ODBC
- Loading Data through JDBC
- Loading Data through ADO.NET

Parallel Load Streams

When you have a large amount of data to load, use parallel load streams to distribute the load operations across the cluster. To do this, create threads from multiple nodes that are connected to the HP Vertica database and load the data. This approach lets you use vsql, ODBC, ADO.NET, or JDBC. You can load server-side files, or load client-side files using the COPY from LOCAL statement.

Best practices for parallel load streams are:

- Issue a single multi-node COPY command that loads different files from different nodes specifying the *nodename* option for each file.
- Issue a single multi-node COPY command that loads different files from any node, using the ON ANY NODE option.
- Use the COPY x WITH SOURCE PloadDelimitedSource option to parallel load using all cores on the server node on which the file resides.

For additional information, see Using Parallel Load Streams.

Trickle Loading

Trickle loading is the process of loading data into your database periodically. This ensures that the load process is sustainable and does not hit too many ROS containers.

HP Vertica uses the transaction isolation level of READ COMMITTED, which allows users to see the most recently committed data without holding any locks. This allows new data to be loaded while concurrent queries are running.

Tune the trickle-loading operations by:

- Adjusting batch size. Generally, larger load batches are more efficient.
- Altering the moveout/mergeout parameters:
  - MergeOutInterval
  - MoveOutInterval
  - MoveOutMaxAgeTime
  - MoveOutSizePct
- To limit the load process to a specific resource so that other resources are available to queries, create a resource pool specifically for the trickle-loading process.

Monitoring Load Operations

When you simulate the data load operations while developing your application, there are two system tables you should monitor to make sure that you have configured load operations in a way that does not negatively impact the database or application performance.

PROJECTION_STORAGE

Monitor the ROS count over time by querying the ROS_COUNT field in the PROJECTION_STORAGE system table and querying partitions that show the ROS_COUNT per partition. Make sure the ROS_COUNT field is not approaching the default maximum of 1024. If this is sustainable for the duration of a partition, and through the time for an old partition to merge, your loading techniques should work fine.

```
VMart=> SELECT projection_id, projection_name, ros_count
        FROM projection_storage;
   projection_id   |       projection_name       | ros_count
-------------------+-----------------------------+-----------
 45035996273719430 | promotion_dimension_super   |         1
 45035996273720042 | online_page_dimension_super |         1
 45035996273719788 | employee_dimension_super    |         1
 45035996273719276 | store_dimension_super       |         1
 45035996273719610 | customer_dimension_super    |         1
 45035996273722132 | T_super                     |         1
 45035996273719098 | product_dimension_super     |         1
 45035996273719942 | warehouse_dimension_super   |         1
 45035996273720000 | shipping_dimension_super    |         1
 45035996273720498 | online_sales_fact_super     |         1
 45035996273719536 | vendor_dimension_super      |         1
 45035996273720206 | store_sales_fact_super      |         1
 45035996273720652 | inventory_fact_super        |         1
 45035996273718920 | date_dimension_super        |         1
 45035996273720336 | store_orders_fact_super     |         1
 45035996273720100 | call_center_dimension_super |         1
 45035996273722194 | T2_super                    |         1
```

```
(17 rows)
```

PARTITIONS

Monitor the PARTITIONS system table to make sure that the number of ROS containers per partition is not reaching the maximum of 1024.

In addition, monitor the `ros_size_bytes` field to see if there are any small ROS containers you can merge.

```
=> SELECT projection_name, ros_id, ros_size_bytes
   FROM partitions;
 projection_name |      ros_id       | ros_size_bytes
-----------------+-------------------+----------------
 t_p_node0001    | 45035996273740461 |             90
 t_p_node0001    | 45035996273740477 |             99
 t_p_node0001    | 45035996273740493 |             99
```

Delete and Update Operations

Many OEMs have small configuration tables with more of an online transaction processing (OLTP) workload. Your HP Vertica database can handle this situation efficiently.

The following sections explain how to configure your database to ensure optimal performance during delete and update operations.

How HP Vertica Deletes and Updates Data

In HP Vertica, delete operations do not actually remove rows from physical storage. Instead, the HP Vertica DELETE statement marks rows as deleted. Update operations have the same effect; the UPDATE statement is actually a combined delete and insert operation. The deleted rows remain in physical storage until the Tuple Mover performs a purge operation to permanently remove deleted data so that the disk space can be reused.

Note: For more information about how deletes and updates work in HP Vertica, see Understanding the Tuple Mover.

Buffering Deletes and Updates

The most efficient way for HP Vertica to perform delete and update operations is to update many database records at a time using fewer statements. You accomplish this by first buffering database deletes and updates in your application before submitting them to HP Vertica. That way, you can manage the flow of the delete and update operations so that they do not interfere with normal database performance.

For more information, see Best Practices for DELETE and UPDATE.

Monitoring Deletes and Updates

To make sure that the frequency of submitting the deletes and updates to the database is sustainable and doesn't have a huge performance impact, monitor the delete and update operations in the database. To do this, regularly monitor the DELETE_VECTORS system table to view the number of deleted records and their corresponding delete vectors that are still physically stored in the system.:

```
VMart=> SELECT node_name, schema_name, projection_name,
           deleted_row_count, used_bytes FROM delete_vectors;
-[ RECORD 1 ]-----+-------------------------
node_name         | v_vmart_node0001
schema_name       | public
projection_name   | product_dimension_super
deleted_row_count | 32345
used_bytes        | 524288
-[ RECORD 2 ]-----+-------------------------
node_name         | v_vmart_node0001
schema_name       | public
projection_name   | employee_dimension_super
deleted_row_count | 943
used_bytes        | 54788
-[ RECORD 3 ]-----+-------------------------
node_name         | v_vmart_node0001
schema_name       | public
projection_name   | warehouse_dimension_super
deleted_row_count | 20
used_bytes        | 16384
```

Purging Deleted Data

Consider scheduling a nightly purge of small tables that receive frequent updates. This operation rewrites the tables while removing deleted records. Run the nightly purge against small tables that won't be expensive to rebuild and that have many deleted records.

For more information, see PURGE_TABLE.

Part 2: Packaging the HP Vertica Database with Your Application

Once you have installed your HP Vertica database, loading representative data into it, and tested the database operation in a simulated production environment, create a script that installs and configures the database with your application. This script should

Perform its operations "silently," in a way that is invisible to your end users.

Be portable so that it can install the database on a cluster with any number of nodes.

The following sections describe what functions the script should perform.

Installing HP Vertica at the Command Line

The packaging script should contain commands to execute the HP Vertica installation procedure.

Important: HP Vertica must be installed separately for each user. You cannot create a system image for subsequent installations because this will cause all users to use the same SSH keys, which can cause serious security issues.

Those commands should perform the following tasks:

1 Login as the root user.

2 Create a properties file that enables non-interactive setup by supplying the parameters you want HP Vertica to use. The properties file is described in Installing HP Vertica Silently.

3 Install the database by running the `install_vertica` script, as described in Installing HP Vertica Silently in the documentation.

1. Make sure to run the scripts with the `–y` parameter to avoid the script asking for EULA agreement.
2. On a single-node installation, consider using the host name or IP address instead of localhost or 127.0.0.1 because of the following limitations:

- The `–s` parameter is required for multi-node installations. On single-node installations the parameter is optional and the default is localhost. However, if you plan to expand to additional hosts later, you must use the `–s` parameter. Installations done without the `–s` parameter cannot be upgraded to a multiple-node deployment.
- On a single-node localhost installation, the installer does not set up a passwordless ssh. Because HP 's backup scripts require that the administrator be able to log into the node via ssh without a password, you must manually enable passwordless ssh logins for any single-node installation if you want to use the backup scripts.
- If either side of a connection is a single-node cluster installed to local house, or you did not specify a host name or IP address for your single-node cluster, importing and exporting data will fail.

4 To save any warnings that the `install_vertica` script returns, redirect any warning messages to a separate file by specifying the `redirect_output =` *filename* parameter in the properties file. After the installation complete, review the warnings and correct any problems that are critical.

5 The following steps are optional:

- Installing the HP Vertica Client Drivers
- Installing the vsql client application on non-cluster hosts

6 Disconnect from the Administration Host.

7 Unless you used the `–L` parameter in the properties file, install the license key you downloaded.

Note: To use data or catalog directories other than those that the `install_vertica` script creates, create those directories and make sure that dbadmin has ownership of those directories.

Creating the Database

To create the HP Vertica database that you plan to embed with your customer application, add the following command to the packaging script:

```
$ /opt/vertica/bin/admintools --tool create_db [ options ]
```

Command-Line Options:

Option	Function
`-h` `--help`	Show this help message and exit.
`-s` *NODES* `--hosts=`*NODES*	Comma-separated list of hosts to participate in database.
`-d` *DB* `--database=`*DB*	Name of database to be created.
`-c` *CATALOG* `--catalog_path=`*CATALOG*	[Optional] Path of catalog directory if not using compat21.
`-D` *DATA* `--data_path=`*DATA*	[Optional] Path of data directory if not using compat21.
`-p` *DBPASSWORD* `--password=`*DBPASSWORD*	[Optional] Database password in single quotes.
`-l` *LICENSEFILE* `--license=`*LICENSEFILE*	[Optional] Database license.
`-P` *POLICY* `--policy=`*POLICY*	[Optional] Database restart policy.
`--compat21`	Use the HP Vertica 2.1 method, using node names instead of host names.

The following example creates a database named `mydb`:

```
$ admintools -t create_db
    -s 10.20.100.66,10.20.100.67,10.20.100.68
    -d mydb -c /home/dbadmin/mydb/catalog
    -D /home/dbadmin/mydb/data -l /home/dbadmin/vlicense.dat
Info: no password specified, using none
Distributing changes to cluster.
10.20.100.66 OK [vertica][(6, 1, 2)][20130430][x86_64]
10.20.100.67 OK [vertica][(6, 1, 2)][20130430][x86_64]
10.20.100.68 OK [vertica][(6, 1, 2)][20130430][x86_64]
Checking full connectivity
Creating database mydb
Node Status: v_mydb_node0001: (DOWN)
Node Status: v_mydb_node0001: (INITIALIZING)
Node Status: v_mydb_node0001: (VALIDATING LICENSE)
Node Status: v_mydb_node0001: (UP)
Creating database nodes
Creating node v_mydb_node0002 (host 10.20.100.67)
Creating node v_mydb_node0003 (host 10.20.100.68)
Node Status: v_mydb_node0001: (UP) v_mydb_node0002: (UP)
    v_mydb_node0003: (DOWN)
Node Status: v_mydb_node0001: (UP) v_mydb_node0002: (UP)
    v_mydb_node0003: (UP)
Database mydb created successfully.
```

Configuring the Database

Execute the scripts that you create in ***Exporting the Optimized Database*** (page 5) to:

- Create the physical schema.
- Create the catalog objects.
- Create the logical schema.
- Create projections that have been configured for use with your application.

Starting the Database

To start your database, add a command to the packaging script that uses the `admintools` script with either the `start_db` or `restart_db` option, as described in this section.

Use the `restart_db` option to restart the database at the last good epoch, minimizing any loss of data.

start_db Option

```
$ /opt/vertica/bin/admintools --tool start_db [ options ]
```

Command-Line Options

Option	Purpose
`-h` `--help`	Show this help message and exit.
`-d` *DB* `--database=`*DB*	Name of database to be started.
`-p` *DBPASSWORD* `--password=`*DBPASSWORD*	Database password in single quotes.
`-i` `--noprompts`	Do not stop and wait for user input. (Default: False.)

restart_db Option

```
$ /opt/vertica/bin/admintools --tool restart_db [ options ]
```

Command-Line Options

Option	Purpose
`-h` `--help`	Show this help message and exit.

`-d` *`DB`*
`--database=`*`DB`*
	Name of database to be restarted.

`-e` *`EPOCH`*
`--epoch=`*`EPOCH`*
	Epoch at which the database is to be restarted. If you pass `'last'` as the argument, the database restarts from the last good epoch.

`-p` *`DBPASSWORD`*
`--password=`*`DBPASSWORD`*
	Database password in single quotes.

`-i` `--noprompts`	Do not stop and wait for user input. (Default: `'False'`.)

Connecting to the Database

To connect to the database that you have started, add the following command to the packaging script:

```
$ /opt/vertica/bin/admintools --tool connect_db [ options ]
```

Command-Line Options

Option	Purpose
`-h` `--help`	Show this help message and exit.

`-d` *`DB`*
`--database=`*`DB`*
	Name of database to connect to.

`-p` *`DBPASSWORD`*
`--password=`*`DBPASSWORD`*
	Database password in single quotes.

Backing Up the Database

Hewlett-Packard strongly recommends backing up your HP Vertica database on a regular basis. This is especially important in an single-node environment where K-safety = 0.

HP Vertica provides a comprehensive utility, the `vbr.py` Python script, that lets you back up, restore, list backups, and copy your database. You can create full and incremental database backups and snapshots of specific schemas or tables to use with a multi-tenant database. Using `vbr.py`, you can save your data to a variety of locations:

- A local directory on the nodes in the cluster
- One or more hosts outside of the cluster
- A different HP Vertica cluster (effectively cloning your database)

Create regular snapshots of your database by running `vbr.py` from a cron job or other task scheduler.

Syntax Conventions

The following are the syntax conventions used in the HP Vertica documentation.

Syntax Convention	Description
`Text without brackets/braces`	Indicates content you type, as shown.
`<` *`Text inside angle brackets`* `>`	Represents a placeholder for which you must supply a value. The variable is usually shown in italics. See *Placeholders* below.
`[ Text inside brackets ]`	Indicates optional items; for example, CREATE TABLE [*schema_name.*]*table_name* The brackets indicate that the *schema_name* is optional. Do not type the square brackets.
`{ Text inside braces }`	Indicates a set of options from which you choose one; for example: `QUOTES { ON | OFF }` indicates that exactly one of ON or OFF must be provided. You do not type the braces: `QUOTES ON`
Backslash `\`	Represents a continuation character used to indicate text that is too long to fit on a single line.
Ellipses `...`	Indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options. Showing ellipses in code examples might also mean that part of the text has been omitted for readability, such as in multi-row result sets.
`    Indentation`	Is an attempt to maximize readability; SQL is a free-form language.
`Placeholders`	Represent items that must be replaced with appropriate identifiers or expressions and are usually shown in italics.
Vertical bar `|`	Is a separator for mutually exclusive items. For example: `[ ASC | DESC ]` Choose one or neither. You do not type the square brackets.

Copyright Notice

Information on third-party software used in HP Vertica, including details on open-source software, is available in the guide Third-Party Software Acknowledgements.