VERTICA

1 Node, Version 5.0

November 2014

Vertica Essentials  
Lab Manual



© Copyright 2014 Hewlett-Packard Development Company, L.P.   
The information contained herein is subject to change without notice.

Table of Contents

[VM System Requirements and Setup 5](#_Toc402775427)

[Essentials Lab Overview 6](#_Toc402775428)

[Start and Connect to an Existing Database 7](#_Toc402775429)

[Loading Sample Data 9](#_Toc402775430)

[Database Designer 14](#_Toc402775431)

[Running the Database Designer 15](#_Toc402775432)

[Data Management 29](#_Toc402775433)

[Bulk loading data directly to ROS via VSQL 30](#_Toc402775434)

[Trickle loading data (WOS) 32](#_Toc402775435)

[Creating Partitions 35](#_Toc402775436)

[Data Loading Questions 37](#_Toc402775437)

[Removing Data 38](#_Toc402775438)

[Dropping Partitions 39](#_Toc402775439)

[Delete Rows 40](#_Toc402775440)

[Purge Deleted Records 42](#_Toc402775441)

[Delete Data Through Partitioning 43](#_Toc402775442)

[Removing, Deleting, and Purging Data Question 46](#_Toc402775443)

[Logical Design and Security 47](#_Toc402775444)

[Creating Users 48](#_Toc402775445)

[Grant Access to Schema 49](#_Toc402775446)

[Create Roles and Grant Privileges 50](#_Toc402775447)

[Assign Users to Roles 52](#_Toc402775448)

[Assign Roles to Other Roles 53](#_Toc402775449)

[Validating: New Users, Roles and Privileges 55](#_Toc402775450)

[Resource Management 56](#_Toc402775451)

[Creating Resource Pools 56](#_Toc402775452)

[Associate a Resource Pool with a User 58](#_Toc402775453)

[Grant Object Level Privileges to a User 59](#_Toc402775454)

[Resource Management Question 60](#_Toc402775455)

[Appendix 61](#_Toc402775456)

[Installation and Setup 61](#_Toc402775457)

[Unpacking the Vertica Installation Package and Installing Vertica 62](#_Toc402775458)

[Running the install\_vertica Script 62](#_Toc402775459)

[Installing the VMart Example Data 64](#_Toc402775460)

[Creating the Sample Database 65](#_Toc402775461)

[Defining Schemas and Tables 68](#_Toc402775462)

[Optional Lab Exercises 71](#_Toc402775463)

[Exercise 1 - Backup and Restore 71](#_Toc402775464)

[Answer Key 80](#_Toc402775465)

[Data Loading Questions 80](#_Toc402775466)

[Removing, Deleting, and Purging Data Question 80](#_Toc402775467)

[Resource Management Question 80](#_Toc402775468)

# VM System Requirements and Setup

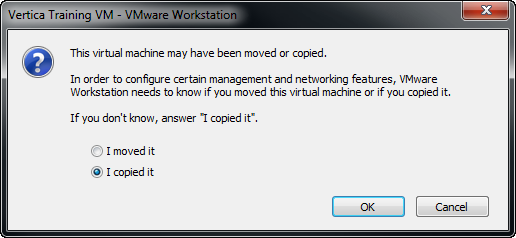
VM System Requirements

To setup the lab virtual environment on your PC, you will need:

* VMware Player 3.1.4 or higher installed OR sufficient privileges to install VMware Player
* A host system that fulfills VMware Player's installation requirements if installation is required
  + 2GHz CPU recommended, minimum 2GB RAM
  + 150 MB of free disk space
* At least 2G RAM free (to run guest Linux OS)

VM Setup Procedure

1. If VMware Player is not installed on your system, please install the provided executable. After the VMware Player installation is complete, Windows will require a reboot.
2. To start the VM, double-click the VerticaTrainingVM.vmx file inside the VerticaTrainingVM directory.
3. The following message will appear after starting up the VM, select the option “I copied it” and click the OK button:



Note: You can use copy and paste to copy commands from the lab manual into the virtual machine. Do not copy the command or vsql prompt. If you use keyboard commands for copy and paste (CTRL-C and CTRL-V respectively), they work slightly differently in the virtual machine. You can still copy with CTRL-C, but paste is SHIFT-CTR-V.

# Essentials Lab Overview

These lab exercises assume you have an environment with Vertica installed, a database created, and a schema defined. Vertica provided lab environments have these steps completed for you. Please refer to the Appendix if you wish to see instructions for installing Vertica, creating a database, and defining the schema.

Learning Objectives

In this lab, students will be able to:

* Start and connect to an existing database
* Load data into an existing schema
* Run the Database Designer in comprehensive mode
* Manage your data
* Remove data
* Create Resource Pools



Trouble with keyboard input? Countries have slight variations in their keyboards. To sync this with Vertica, type the command **system-config-keyboard** at a Linux prompt (it will prompt you for the root password if you are not logged in as root) and select the appropriate keyboard from the drop down list.

# Start and Connect to an Existing Database

In this step, you will start the existing database and connect to it.

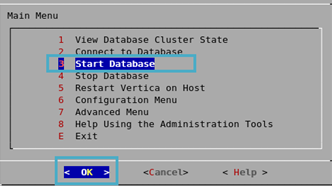
1. Access a terminal window by clicking on the  icon at the top of the window and navigate to the VMart examples directory:

$ cd /opt/vertica/examples/VMart\_Schema

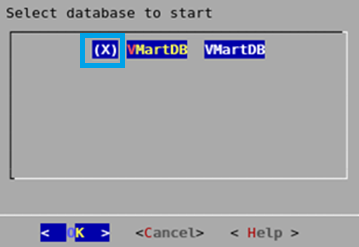
1. Run Administration Tools:

$ adminTools

1. Select "Start Database" and click "OK":

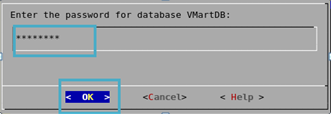


1. Select "VMartDB" and click "OK":

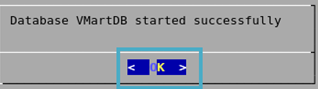


Start and Connect to an Existing Database (Continued)

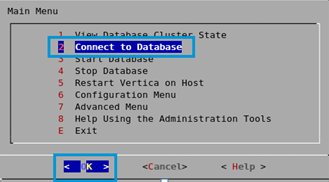
1. Enter the database password: *password* and click "OK":



1. Click "OK" once the database has started successfully:



1. Select "Connect to Database" and Click "OK":



# Loading Sample Data

Learning Objectives:

* Load data into an existing schema

Load Sample Data

1. Load data into the VMart database using the vmart\_load\_data.sql script. This process will take a few moments to complete. (See output example below)

VMartDB=> \i vmart\_load\_data.sql

Rows Loaded

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

1826

(1 row)

Rows Loaded

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

60000

(1 row)

Rows Loaded

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

250

(1 row)

Loading Sample Data (Continued)

1. Execute the following query to examine the superprojections that were created by default during the initial data load:

VMartDB=> SELECT projection\_schema, projection\_name, anchor\_table\_name FROM projections;

projection\_schema| projection \_name |anchor\_table\_name

-----------------+-------------------------+-----------------

public | date\_dimension\_b0 |date\_dimension

public | date\_dimension\_b1 |date\_dimension

public | product\_dimension\_b0 |product\_dimension

public | product\_dimension\_b1 |product\_dimension

store | store\_dimension\_b0 |store\_dimension

store | store\_dimension\_b1 |store\_dimension

public | promotion\_dimension\_b0 |promotion\_dimension

public | promotion\_dimension\_b1 |promotion\_dimension

public | vendor\_dimension\_b0 |vendor\_dimension

public | vendor\_dimension\_b1 |vendor\_dimension

public | customer\_dimension\_b0 |customer\_dimension

public | customer\_dimension\_b1 |customer\_dimension

public | employee\_dimension\_b0 |employee\_dimension

public | employee\_dimension\_b1 |employee\_dimension

public | warehouse\_dimension\_b0 |warehouse\_dimension

public | warehouse\_dimension\_b1 |warehouse\_dimension

public | shipping\_dimension\_b0 |shipping\_dimension

public | shipping\_dimension\_b1 |shipping\_dimension

online\_sales | online\_page\_dimension\_b0|online\_page\_dimension

online\_sales | online\_page\_dimension\_b1|online\_page\_dimension

online\_sales | call\_center\_dimension\_b0|call\_center\_dimension

online\_sales | call\_center\_dimension\_b1|call\_center\_dimension

store | store\_sales\_fact\_b0 |store\_sales\_fact

store | store\_sales\_fact\_b1 |store\_sales\_fact

store | store\_orders\_fact\_b0 |store\_orders\_fact

store | store\_orders\_fact\_b1 |store\_orders\_fact

online\_sales | online\_sales\_fact\_b0 |online\_sales\_fact

online\_sales | online\_sales\_fact\_b1 |online\_sales\_fact

public | inventory\_fact\_b0 |inventory\_fact

public | inventory\_fact\_b1 |inventory\_fact

Loading Sample Data (Continued)

1. Turn on timing and execute the following query:

VMartDB=> \timing

Timing is on.

VMartDB=> SELECT DISTINCT s.product\_key, p.product\_description

FROM store.store\_sales\_fact s, public.product\_dimension p

WHERE s.product\_key = p.product\_key

AND s.product\_version = p.product\_version

AND s.store\_key IN (

SELECT store\_key

FROM store.store\_dimension

WHERE store\_state = 'MA')

ORDER BY s.product\_key

LIMIT 10;

1. After the query completes, you will see results, as well as how long it took to execute the query.

product\_key | product\_description

--------------+-----------------------------------

1 | Brand #1 butter

1 | Brand #2 bagels

2 | Brand #3 lamb

2 | Brand #4 brandy

2 | Brand #5 golf clubs

2 | Brand #6 chicken noodle soup

3 | Brand #10 ground beef

3 | Brand #11 vanilla ice cream

3 | Brand #7 canned chicken broth

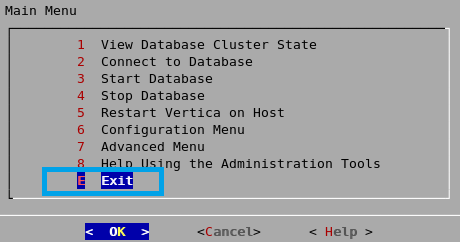
3 | Brand #8 halibut

1. Record how long your query took to execute:\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Exit from vsql.

VMartDB=> \q

Loading Sample Data (Continued)

1. Exit from adminTools



1. The user will now be at a linux prompt as shown below.

[dbadmin@node1 VMart\_Schema]$



# Database Designer

In this module, you will use the demonstration data to explore projections, the fundamental vehicle for physical storage in Vertica.

Learning Objectives:

* Run Database Designer in comprehensive mode.

## Running the Database Designer

1. Change to the directory /opt/vertica/examples/VMart\_Schema, and then launch the adminTools interface.

$ cd /opt/vertica/examples/VMart\_Schema

$ adminTools

1. In order to run the Database Designer with our 5.1 VM, we will disable the Vertica Resource Manager. The Resource Manager provides options and controls to ensure that every query gets serviced in a concurrent environment and that true system limits are respected at all times.

It is not recommended that you disable the Resource Manager in a standard environment.

Connect to the database, and enter the password: *password*.  
To disable Resource Manager, enter:

VMart\_Schema=> select set\_config\_parameter('EnableResourceManager','0');

set\_config\_parameter

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

Parameter set successfully

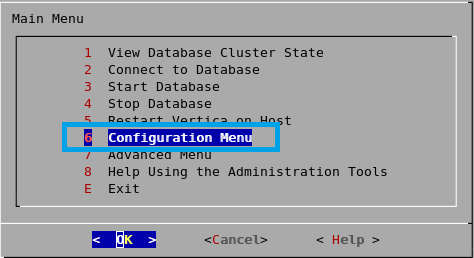
(1 row)

1. Exit from vsql

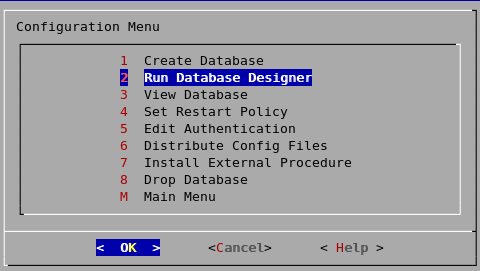
VMartDB=> \q

Database Designer (Continued)

1. From the Main Menu, select Configuration Menu and click the OK button.

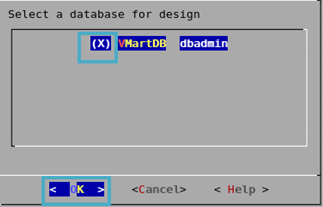


1. Select Run Database Designer and then click the OK button.

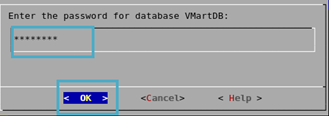


Database Designer (Continued)

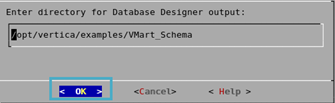
1. Select VMartDB in the Select a database dialog box and then click the OK button.



1. Enter the database password: *password*



1. Click the OK button to accept the default pathname in the Enter directory field dialog:

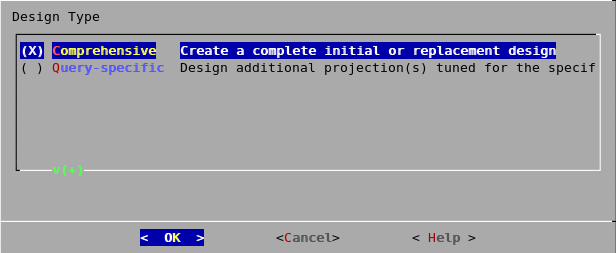


1. Name the design VMart\_Design in the Design Configuration dialog box, and then click the OK button.

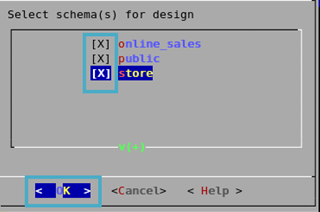


Database Designer (Continued)

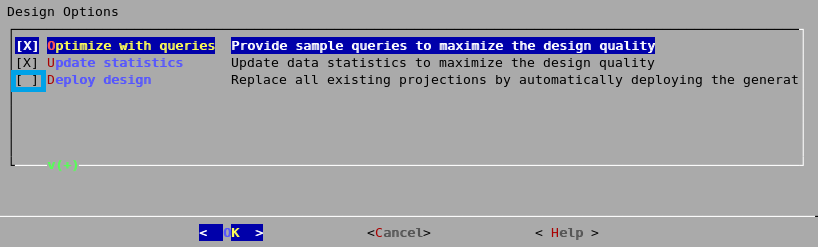
1. In the Design Type window, select Comprehensive to create a complete initial design, and then click the OK button.



1. Select all three schemas for your design, and then click the OK button.

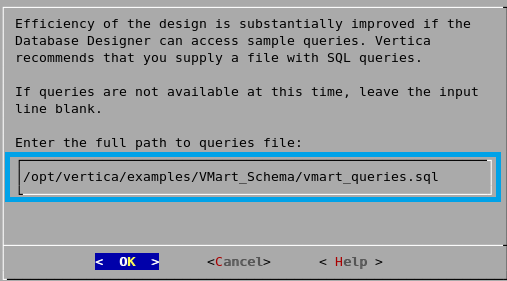


1. In the Design Options window, unselect the option Deploy Design and click OK.

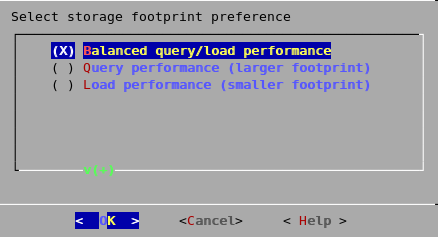


Database Designer (Continued)

1. Type the full path to the queries file (/opt/vertica/examples/VMart\_Schema/vmart\_queries.sql), and then click the OK button:



1. In the Select storage footprint preference window, select Balanced query/load performance and click the OK button:



Database Designer (Continued)

1. When the informational message displays, click the Proceed button.

The Database Designer is ready to generate a new comprehensive design - a complete set of projections for the tables in the selected schema(s). Please review the options you selected:

The database statistics will be updated. Accurate statistics assure the best design quality, however updating statistics takes time and resources. If the current statistics are up-to-date, this step may be unnecessary.

The new design will not be automatically deployed. For manual deployment procedures consult Vertica Administrator's Guide.

The generated deployment script will be saved to /opt/vertica/examples/VMart\_Schema/VMartDesign\_deploy.sql

For large databases a design session could take a long time; allow it to complete uninterrupted. Use Ctrl+C if you must cancel the session.

To change any of the options press <cancel> to return to the Design Options menu.

<Proceed> <Cancel>

Note: The Database Designer performs the following tasks: (See Example Output on the next page)

* Examines table data
* Analyzes statistics
* Loads queries from the vmart\_queries.sql you provided
* Recommends an optimized design based upon the representative data and sample queries

Database Designer (Continued)

Database Designer started.

For large databases a design session may take a long time, yet it is best to allow this process to complete un-interrupted.

If the session must be canceled, use Ctrl+C.

Setting up design session...

Examining table data...

Loading queries from '/opt/vertica/examples/VMart\_Schema/vmart\_queries.sql'.

Processed 9 SQL statement(s), all accepted and considered in the design.

Saving existing projections to /opt/vertica/examples/VMart\_Schema/VMartDesign\_projection\_backup\_0307124244.sql ...

[100%] Saving existing projection definitions... 15 of 15

Creating design...

[ 73%] Analyzing data statistics... Completed 11 of 15 tables. Analyzin [100%] Analyzing data statistics... Completed 15 of 15 tables.

0%] Optimizing for query performance... Completed 0 of 9 queries. En [100%] Optimizing for query performance... Completed 9 of 9 queries.

0%] Optimizing storage footprint... Completed 0 of 15 tables. Optimi [ 33%] Optimizing storage footprint... Completed 5 of 15 tables. Optimi [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [ 66%] Optimizing storage footprint... Completed 10 of 15 tables. Optim [100%] Optimizing storage footprint... Completed 15 of 15 tables.

Database Designer (Continued)

Query optimization results...ress...

Query 1 optimization ratio or status is 1

Query 2 optimization ratio or status is 1

Query 3 optimization ratio or status is 1

Query 4 optimization ratio or status is 1

Query 5 optimization ratio or status is 1

Query 6 optimization ratio or status is 1

Query 7 optimization ratio or status is 1

Query 8 optimization ratio or status is 1

Query 9 optimization ratio or status is 1

Generating deployment script...

Generated deployment script successfully

Design script is located in /opt/vertica/examples/VMart\_Schema/VMartDesign\_design.sql

Deployment script is located in /opt/vertica/examples/VMart\_Schema/VMartDesign\_deploy.sql

The design will not be deployed.

The new design was not automatically deployed.

For manual deployment procedures consult Vertica Administrator's Guide

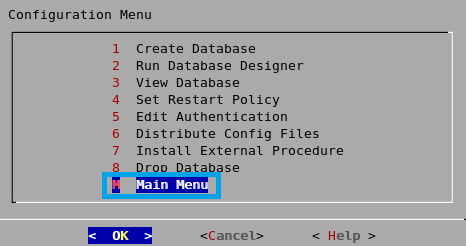
Database Designer finished.

Press <Enter> to return to the Administration Tools menu.

1. When Database Designer completes, press the Enter key to return to the adminTools menu.

Database Designer (Continued)

1. Return to the Main Menu and select Connect to Database to bring up a vsql session.



1. First, we will enable the Resource Manager:

VMart\_Schema=> select set\_config\_parameter('EnableResourceManager','1');

set\_config\_parameter

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

Parameter set successfully

(1 row)

1. Execute the following query to examine the superprojections that were created by default during the initial data load:

VMartDB=> SELECT projection\_schema, projection\_name, anchor\_table\_name FROM projections;

projection\_schema| projection\_name | anchor\_table\_name

-----------------+-----------------------------+----------------------

store | store\_sales\_fact\_super | store\_sales\_fact

store | store\_orders\_fact\_super | store\_orders\_fact

online\_sales | online\_sales\_fact\_super | online\_sales\_fact

public | inventory\_fact\_super | inventory\_fact

public | date\_dimension\_super | date\_dimension

public | product\_dimension\_super | product\_dimension

store | store\_dimension\_super | store\_dimension

public | promotion\_dimension\_super | promotion\_dimension

public | vendor\_dimension\_super | vendor\_dimension

public | customer\_dimension\_super | customer\_dimension

public | employee\_dimension\_super | employee\_dimension

public | warehouse\_dimension\_super | warehouse\_dimension

public | shipping\_dimension\_super | shipping\_dimension

online\_sales | online\_page\_dimension\_super | online\_page\_dimension

online\_sales | call\_center\_dimension\_super | call\_center\_dimension

Database Designer (Continued)

1. Use cat to view the deployment script generated by the database designer. Note the new name given to each projection, then the projection definition, followed by a select refresh command to populate the new projections with data. Once the refresh completes, the previous version of the projections are dropped.

VMartDB=> \! cat VMartDesign\_deploy.sql

CREATE PROJECTION store\_sales\_fact\_DBD\_5\_rep\_VMartDesign\_VMartDesign (

date\_key ENCODING DELTAVAL,

product\_key ENCODING RLE, …)

AS

SELECT date\_key,

product\_key, …

FROM store.store\_sales\_fact

ORDER BY product\_key,

product\_version, …

UNSEGMENTED ALL NODES;

select refresh('store.store\_sales\_fact');

select make\_ahm\_now();

1. Execute the deployment script in vsql.

VMartDB=> \i VMartDesign\_deploy.sql

Refresh completed with the following outcomes:

Projection Name: [Anchor Table] [Status] [Refresh Method] [Error Count] [Duration (sec)]

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

"public"."inventory\_fact\_DBD\_10\_rep\_VMartDesign\_node0001": [inventory\_fact] [refreshed] [scratch] [0] [11]

"store"."store\_dimension\_DBD\_11\_rep\_VMartDesign\_node0001": [store\_dimension] [refreshed] [scratch] [0] [6]

"store"."store\_sales\_fact\_DBD\_12\_rep\_VMartDesign\_node0001": [store\_sales\_fact] [refreshed] [scratch] [0] [82]

"store"."store\_orders\_fact\_DBD\_14\_rep\_VMartDesign\_node0001": [store\_orders\_fact] [refreshed] [scratch] [0] [14]

"store"."store\_orders\_fact\_DBD\_13\_rep\_VMartDesign\_node0001": [store\_orders\_fact] [refreshed] [scratch] [0] [14]

(1 row)

make\_ahm\_now

AHM set (New AHM Epoch: 1182)

(1 row)

Database Designer (Continued)

1. Once the script completes, view the newly deployed projections. Note the new names granted to each projection.

VMartDB=> SELECT projection\_schema, projection\_name, anchor\_table\_name FROM projections;

projection\_schema|projection\_name

|anchor\_table\_name

----------------+-----------------------------+--------------------

public | customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001 | customer\_dimension

public | product\_dimension\_DBD\_2\_rep\_VMartDesign\_node0001 | product\_dimension

public | promotion\_dimension\_DBD\_3\_rep\_VMartDesign\_node0001 | promotion\_dimension

public | date\_dimension\_DBD\_4\_rep\_VMartDesign\_node0001 | date\_dimension

public | vendor\_dimension\_DBD\_5\_rep\_VMartDesign\_node0001 | vendor\_dimension

public | vendor\_dimension\_DBD\_6\_rep\_VMartDesign\_node0001 | vendor\_dimension

public | employee\_dimension\_DBD\_7\_rep\_VMartDesign\_node0001 | employee\_dimension

public | shipping\_dimension\_DBD\_8\_rep\_VMartDesign\_node0001 | shipping\_dimension

public | warehouse\_dimension\_DBD\_9\_rep\_VMartDesign\_node0001 | warehouse\_dimension

public | inventory\_fact\_DBD\_10\_rep\_VMartDesign\_node0001 | inventory\_fact

store | store\_dimension\_DBD\_11\_rep\_VMartDesign\_node0001 | store\_dimension

store | store\_sales\_fact\_DBD\_12\_rep\_VMartDesign\_node0001 | store\_sales\_fact

store | store\_orders\_fact\_DBD\_13\_rep\_VMartDesign\_node0001 | store\_orders\_fact

store | store\_orders\_fact\_DBD\_14\_rep\_VMartDesign\_node0001 | store\_orders\_fact

online\_sales |online\_page\_dimension\_DBD\_15\_rep\_VMartDesign\_node0001 | online\_page\_dimension

online\_sales |call\_center\_dimension\_DBD\_16\_rep\_VMartDesign\_node0001 | call\_center\_dimension

online\_sales | online\_sales\_fact\_DBD\_17\_rep\_VMartDesign\_node0001 | online\_sales\_fact

online\_sales | online\_sales\_fact\_DBD\_18\_rep\_VMartDesign\_node0001 | online\_sales\_fact

Database Designer (Continued)

1. Turn on query timing using \timing and run the same query from Lab Module 1. Compare how long the query took to execute against the value recorded on page 11.

VMartDB=> \timing

Timing is on.

VMartDB=> SELECT DISTINCT s.product\_key, p.product\_description

FROM store.store\_sales\_fact s, public.product\_dimension p

WHERE s.product\_key = p.product\_key

AND s.product\_version = p.product\_version

AND s.store\_key IN (

SELECT store\_key

FROM store.store\_dimension

WHERE store\_state = 'MA')

ORDER BY s.product\_key

LIMIT 10;

product\_key | product\_description

-------------+-------------------------------

1 | Brand #1 butter

1 | Brand #2 bagels

2 | Brand #3 lamb

2 | Brand #4 brandy

2 | Brand #5 golf clubs

2 | Brand #6 chicken noodle soup

3 | Brand #10 ground beef

3 | Brand #11 vanilla ice cream

3 | Brand #7 canned chicken broth

3 | Brand #8 halibut

(10 rows)

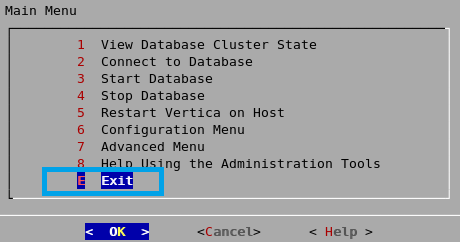
Time: First fetch (10 rows): 1487.352 ms. All rows formatted: 1487.451 ms

Database Designer (Continued)

1. Exit from vsql

VMartDB=> \q

1. Exit from the adminTools interface.





# Data Management

In this module, you will explore various methods to load data into Vertica, as well as cover designing partitioning schemes as a best practice to managing your data.

Learning Objectives:

* Load data into ROS
* Load data into WOS
* Describe the hybrid storage model with Tuple Mover
* Monitor the loading process
* Partition data and describe the benefits that partitioning provides

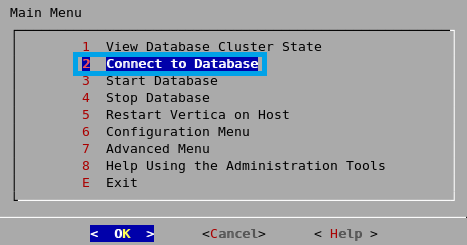
## Bulk loading data directly to ROS via VSQL

1. Change to the directory /opt/vertica/examples/VMart\_Schema, and then launch the adminTools interface.

$ cd /opt/vertica/examples/VMart\_Schema

$ adminTools

2. Select Connect to Database to launch a vsql session. (Note: If the database asked you for a password, the password is *password)*



Data Management (Continued)

1. Run the script vmart\_load\_data.sql to load the example data:

VMartDB=> \i vmart\_load\_data.sql

Rows Loaded

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

1826

(1 row)

Rows Loaded

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

60000

(1 row)

Rows Loaded

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

250

(1 row)

1. Examine the contents of the load\_streams table to see information on load progress:

VMartDB=> SELECT \* FROM load\_streams;

session\_id | transaction\_id | statement\_id | stream\_name |schema\_name | table\_id | table\_name

|load\_start | load\_duration\_ms | is\_executing

|accepted\_row\_count | rejected\_row\_count | read\_bytes

|input\_file\_size\_bytes | parse\_complete\_percent

|unsorted\_row\_count | sorted\_row\_count | sort\_complete\_percent

-------------------+-------------------+--------------+--------

node1-3255:0xae | 45035996273717198 | 1 | |public | 45035996273733204 | inventory\_fact | 2014-03-04 08:53:55.983536-05 |2939 | f | 300000 | 0 | 5593160 | 5593160 | 100 | 1200000 | 1200000 | 100

node1-3255:0xae | 45035996273717192 | 1 |

|online\_sales | 45035996273733248 | online\_sales\_fact | 2014-03-04 08:53:08.78367-05 47192 | f | 5000000 | 0 | 379423758 | 379423758 |

100 | 20000000 | 20000000 | 100

…

## Trickle loading data (WOS)

1. Monitor the data in WOS and ROS by querying the column\_storage system table:

VMartDB=> SELECT wos\_row\_count, ros\_row\_count, anchor\_table\_name FROM column\_storage WHERE anchor\_table\_name = 'date\_dimension';

wos\_row\_count | ros\_row\_count | anchor\_table\_name

---------------+---------------+-------------------

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

0 | 3652 | date\_dimension

Notice that the WOS row count for this anchor table is currently empty.

1. From the command prompt, load data from the Date\_Dimension.tbl file into WOS by removing the “DIRECT” keyword:

VMartDB=> COPY Date\_Dimension FROM '/opt/vertica/examples/VMart\_Schema/Date\_Dimension.tbl' DELIMITER '|' NULL '';

Rows Loaded

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

1826

Trickle loading data (WOS) (Continued)

1. Launch vsql from adminTools. Check the column\_storage system table for the Date Dimension table:

VMartDB=> SELECT wos\_row\_count, ros\_row\_count, anchor\_table\_name FROM projection\_storage WHERE anchor\_table\_name = 'date\_dimension';

wos\_row\_count | ros\_row\_count | anchor\_table\_name

---------------+---------------+-------------------

1826 | 3652 | date\_dimension

Without the “DIRECT” keyword, the data is loaded into WOS.

Note: It is possible that no data will be remaining in WOS if a moveout has occurred since the load in the previous exercise.

1. Manually invoke the moveout procedure to move the data from the WOS to the ROS:

VMartDB=> SELECT do\_tm\_task('moveout');

do\_tm\_task

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

Task: moveout

(Table: public.customer\_dimension) (Projection: public.customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001)

(Table: public.product\_dimension) (Projection: public.product\_dimension\_DBD\_2\_rep\_VMartDesign\_node0001)

(Table: public.promotion\_dimension) (Projection: public.promotion\_dimension\_DBD\_3\_rep\_VMartDesign\_node0001)

(Table: public.date\_dimension) (Projection: public.date\_dimension\_DBD\_4\_rep\_VMartDesign\_node0001)

(Table: public.vendor\_dimension) (Projection: public.vendor\_dimension\_DBD\_5\_rep\_VMartDesign\_node0001)

(Table: public.vendor\_dimension) (Projection: public.vendor\_dimension\_DBD\_6\_rep\_VMartDesign\_node0001)

(Table: public.employee\_dimension) (Projection:

…

Note: Moveout runs automatically in the background at a default interval of 5 minutes.

Trickle loading data (WOS) (Continued)

1. Check the column\_storage system table for the Date Dimension table again:

VMartDB=> SELECT wos\_row\_count, ros\_row\_count, anchor\_table\_name FROM projection\_storage WHERE anchor\_table\_name = 'date\_dimension';

wos\_row\_count | ros\_row\_count | anchor\_table\_name

---------------+---------------+-------------------

0| 5478 | date\_dimension

The tuple mover has moved the data out of the WOS and has written it on disk to ROS.

## Creating Partitions

1. From vsql, view the current partition information:

VMartDB=> SELECT \* FROM partitions;

partition\_key | projection\_id | table\_schema | projection\_name | ros\_id | ros\_size\_bytes | ros\_row\_count | node\_name | deleted\_row\_count | location\_label

---------------+---------------+--------------+-----------------+--------+----------------+---------------+-----------+-------------------+----------------

(0 rows)

1. The VMart schema does not partition tables. Create a new table for store\_sales\_fact that is partitioned:

VMartDB=> CREATE TABLE store.store\_sales\_fact\_part

( product\_key integer not null,

store\_key integer not null,

customer\_key integer not null,

pos\_transaction\_number integer not null,

sales\_dollar\_amount integer,

transaction\_time time not null

) PARTITION BY date\_part('hour',transaction\_time);

WARNING 6100: Using PARTITION expression that returns a Numeric value

HINT: This PARTITION expression may cause too many data partitions. Use of an expression that returns a more accurate value, such as a regular VARCHAR or INT, is encouraged

CREATE TABLE

Note the addition of the “not null” clause to the transaction\_time column (as compared to the original table definition for store\_sales\_fact). Partitioning requires non-null values in the column enforced in the table definition.

If the original definition of store\_sales\_fact had constrained the transaction\_time column to not null, partitioning could be done using ALTER TABLE store\_sales\_fact PARTITION BY date\_part('hour',transaction\_time) REORGANIZE;

Creating Partitions (Continued)

1. Vertica automatically partitions data when data is written to disk. Load the data from the store\_sales\_fact table into the newly created partitioned table:

VMartDB=> INSERT /\*+ DIRECT \*/ INTO store.store\_sales\_fact\_part SELECT product\_key, store\_key, customer\_key, pos\_transaction\_number, sales\_dollar\_amount, transaction\_time FROM store.store\_sales\_fact;

OUTPUT

----------

10000000

1. Commit the load and check the partition keys:

VMartDB=> commit;

COMMIT

VMartDB=> SELECT \* FROM partitions;

Verify that there were 24 unique partitions created, 1 for each hour of the day.

partition\_key | projection\_id | table\_schema | projection\_name | ros\_id | ros\_size\_bytes | ros\_row\_count | node\_name | deleted\_row\_count | location\_label

---------------+-------------------+--------------+-------------------

23 | 45035996273864340 | store | store\_sales\_fact\_part\_super | 45035996273864371 | 10191432 | 1036990 | v\_vmartdb\_node0001 | 0 |

…

## Data Loading Questions

Is it possible to load data into a Vertica system without using WOS? What are the advantages and disadvantages of such a decision?

What is the difference between partitioning and segmentation?

The data in WOS exists only in RAM. What are some possible issues associated with that?

Click [here](#_Data_loading) to go to answer key



# Removing Data

The Vertica Analytic Database is optimized to perform optimized insert and append operations and does not perform update or deletes in place. This module builds on the best practices for deletes and updates detailed in the user documentation on how to remove data.

Learning Objectives

* Optimize projections for deletes
* Purge deleted records
* Remove data through partitioning

## Dropping Partitions

Suppose you wanted to remove data from 00:00:00 to 03:00:00 so that you can load new data.

In the previous exercise, you created hourly partitions from the table store.store\_sales\_fact based on the transaction\_time. You now want to drop the partitions for the first three hours. The partitions must be dropped one at a time.

1. Drop the '00','01' and '02' partitions:

VMartDB=> SELECT drop\_partition('store.store\_sales\_fact\_part', '00');

drop\_partition

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

Partition dropped

VMartDB=> SELECT drop\_partition('store.store\_sales\_fact\_part', '01');

drop\_partition

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

Partition dropped

VMartDB=> SELECT drop\_partition('store.store\_sales\_fact\_part', '02');

drop\_partition

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

Partition dropped

1. Verify that partitions '00', '01', and '02' have been removed.

VMartDB=> SELECT DISTINCT partition\_key FROM partitions;

partition\_key

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

11

3

4

5

6

7

8

9

13

14

12

10

20

21

22

…

## Delete Rows

* + 1. Verify that there are no deleted rows currently in the storage\_containers table:

VMartDB=> SELECT \* FROM storage\_containers WHERE deleted\_row\_count > 0;

node\_name | schema\_name | projection\_name | storage\_type | storage\_oid | total\_row\_count | deleted\_row\_count | used\_bytes | start\_epoch | end\_epoch | grouping

-----------+-------------+-----------------+--------------+-------------+-----------------+-------------------+------------+-------------+-----------+----------

(0 rows)

* + 1. Delete rows from the customer\_dimension table:

VMartDB=> DELETE FROM public.customer\_dimension WHERE customer\_age = 100 AND date\_part('year', last\_deal\_update) = 2007;

OUTPUT

----------

8

* + 1. Commit the deletion

VMartDB=> commit;

COMMIT

Removing Data (Continued)

* + 1. A record of the deleted rows is stored in the storage\_containers.deleted\_row\_count. Show a list of containers that have had records removed from them.

VMartDB=> SELECT \* FROM storage\_containers WHERE deleted\_row\_count > 0;

## 

node\_name | schema\_name | projection\_id | projection\_name | storage\_type | storage\_oid | total\_row\_count | deleted\_row\_count | used\_bytes | start\_epoch | end\_epoch | grouping | segment\_lower\_bound | segment\_upper\_bound | is\_sorted | location\_label | delete\_vector\_count

--------------------+-------------+-------------------+---------------------------------------------------+--------------+------------------

v\_vmartdb\_node0001 | public | 45035996273860720 | customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001 | ROS | 45035996273861259 | 50000 | 4 | 1674457 | 18 | 18 | PROJECTION | | | t | | 1

v\_vmartdb\_node0001 | public | 45035996273860720 | customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001 | ROS | 45035996273862951 | 50000 | 4 | 1674457 | 1192 | 1192 | PROJECTION | | | t | | 1

## Purge Deleted Records

1. Run the purge\_table command:

VMartDB=> SELECT purge\_table('customer\_dimension');

purge\_table

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

Task: purge operation

(Table: public.customer\_dimension) (Projection: public.customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001)

(1 row)

1. Monitor the deleted row count after purging records:

VMartDB=> SELECT \* FROM storage\_containers WHERE deleted\_row\_count > 0;

node\_name | schema\_name | projection\_id | projection\_name | storage\_type | storage\_oid | **total\_row\_count | deleted\_row\_count** | used\_bytes | start\_epoch | end\_epoch | grouping | segment\_lower\_bound | segment\_upper\_bound | is\_sorted | location\_label | **delete\_vector\_count**

--------------------+-------------+-------------------+----------------------------------------------------+--------------+-------------------+-----------------+-------------------+------------+-------------+-----------+------------+---------------------+---------------------+-----------+----------------+---------------------

v\_vmartdb\_node0001 | public | 45035996273832680 | customer\_dimension\_DBD\_1\_rep\_VMartDesign\_node0001 | ROS | 45035996273837379 | 100000 | 8 | 2308186 | 18 | 1190 | PROJECTION | | | t | | 1

It is possible that the delete vectors will not be purged after the running of purge\_table. This is due to a configuration parameter called AdvanceAHMInterval. By default, this parameter is set to 180 seconds; it determines how frequently the AHM is moved based on the current epoch. Since you are performing these tasks in quick succession, you may run the purge prior to the AHM being advanced. Running purge\_table a second time will allow the AHM to advance as specified by this configuration parameter. In practice, this is normally moot.

## Delete Data Through Partitioning

During the database design phase, a delete use case would normally be identified. If the deleted records are logically contained within a single or a small numbers of partitions, a logical, more efficient approach to deletes is to partition the table and follow a few steps to ultimately remove only a portion of the data.

1. Create a partitioned version of the customer\_dimension table, partitioned by age, in increments of 10:

VMartDB=> CREATE TABLE public.customer\_dimension\_partitioned

( customer\_key int NOT NULL,

customer\_type varchar(16),

customer\_name varchar(256),

customer\_gender varchar(8),

title varchar(8),

household\_id int,

customer\_address varchar(256),

customer\_city varchar(64),

customer\_state char(2),

customer\_region varchar(64),

marital\_status varchar(32),

customer\_age int NOT NULL,

number\_of\_children int,

annual\_income int,

occupation varchar(64),

largest\_bill\_amount int,

store\_membership\_card int,

customer\_since date,

deal\_stage varchar(32),

deal\_size int,

last\_deal\_update date

) PARTITION BY customer\_age %10;

CREATE TABLE

Delete Data Through Partitioning (Continued)

1. From the Linux prompt, navigate to the directory /opt/vertica/examples/VMart\_Schema and load data into the temporary projection for this table:

VMartDB=> COPY public.customer\_dimension\_partitioned from '/opt/vertica/examples/VMart\_Schema/Customer\_Dimension.tbl' DIRECT;

Rows Loaded

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

50000

1. Count the records in the customer\_dimension\_partitioned table for customers with an age of 100 and an income above $7,000,000; it is assumed those records are erroneous.

VMartDB=> SELECT count(\*) FROM public.customer\_dimension\_partitioned WHERE customer\_age = 100 AND annual\_income > 7000000;

count

-------

16

1. Create a new table to store information on customers where customer\_age%10 = 0 and customer\_age <> 100, or customer\_age=100 and annual\_income < 7000000:

VMartDB=> CREATE TABLE customer\_dimension\_partitioned\_temp AS SELECT \* FROM customer\_dimension\_partitioned WHERE (customer\_age%10=0 AND customer\_age<>100) OR (customer\_age = 100 AND annual\_income < 7000000);

CREATE TABLE

1. Drop the original partition containing these records, along with the ones assumed to be in error. Based on the partitioning, the affected partition has a partition\_key=0:

VMartDB=> SELECT drop\_partition('customer\_dimension\_partitioned','0');

drop\_partition

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

Partition dropped

1. Insert the records from customer\_dimension\_partitioned\_temp (those not affected by the delete) into public.customer\_dimension\_partitioned:

VMartDB=> INSERT /\*+DIRECT\*/ INTO public.customer\_dimension\_partitioned SELECT \* FROM customer\_dimension\_partitioned\_temp;

Output

-------

5559

Delete Data Through Partitioning (Continued)

1. Commit the output

VMartDB=> COMMIT;

COMMIT

1. Drop the temp table:

VMartDB=> DROP TABLE customer\_dimension\_partitioned\_temp CASCADE;

DROP TABLE

## Removing, Deleting, and Purging Data Question

What are the advantages and disadvantages of creating delete vectors to identify deleted records?

Click [here](#_Removing,_deleting,_and) to go to answer key



# Logical Design and Security

In this module you will work with creating users and roles and granting privileges. You will create users and assign them to the Public schema. You will create roles and grant those roles various privileges, assign a user to a role and a role to another role. These exercises will use existing schemas and tables.

Learning Objectives:

* Creating users
* Assign users to the Public schema
* Create roles
* Grant roles privileges
* Assign a user to a role
* Assign a role to another role

## Creating Users

In this exercise, you will create users named Fred and Jane and assign the users specific passwords.

1. From vsql, create a new user named Fred and give him the password *password1*.

VMartDB=> CREATE USER Fred IDENTIFIED BY 'password1';

CREATE USER

1. From vsql, create a new user named Jane and give her the password *password2*.

VMartDB=> CREATE USER Jane IDENTIFIED BY 'password2';

CREATE USER

Verify that you have created the users Fred and Jane.

1. From vsql, display all users using \du

VMartDB=> \du

List of users

User name | Is Superuser

-----------+--------------

Fred | f

Jane | f

dbadmin | t

## Grant Access to Schema

In this exercise, you will grant users named Fred and Jane access to the Public schema.

1. From vsql, grant user Fred access to the Public schema.

VMartDB=> GRANT USAGE ON SCHEMA Public TO Fred;

GRANT PRIVILEGE

1. From vsql, grant user Jane access to the Public schema.

VMartDB=> GRANT USAGE ON SCHEMA Public TO Jane;

GRANT PRIVILEGE

Verify that users Fred and Jane have been granted access privileges to Schema Public.

1. From vsql, display all access privileges using \dp

VMartDB=> \dp

Access privileges for database "VMartDB"

Grantee | Grantor | Privileges | Schema | Name

---------+---------+------------+--------+----------------------------

Fred | dbadmin | USAGE | | general

Jane | dbadmin | USAGE | | general

public | dbadmin | USAGE | | public

Fred | dbadmin | USAGE | | public

Jane | dbadmin | USAGE | | public

## Create Roles and Grant Privileges

In this exercise, you will create three roles and will grant them various privileges. This exercise will use the existing table inventory\_fact.

1. From vsql, create a new role called sales\_general\_role.

VMartDB=> CREATE ROLE sales\_general\_role;

CREATE ROLE

1. From vsql, grant SELECT on the table called inventory\_fact to the role called sales\_general\_role.

VMartDB=> GRANT SELECT ON inventory\_fact TO sales\_general\_role;

CREATE PRIVILEGE

1. From vsql, create a new role called sales\_role.

VMartDB=> CREATE ROLE sales\_role;

CREATE ROLE

1. From vsql, grant INSERT on the table called inventory\_fact to the role called sales\_role.

VMartDB=> GRANT INSERT ON inventory\_fact TO sales\_role;

CREATE PRIVILEGE

1. From vsql, create a new role called sales\_admin\_role.

VMartDB=> CREATE ROLE sales\_admin\_role;

CREATE ROLE

Create Roles and Grant Privileges (Continued)

1. From vsql, grant DELETE on the table called inventory\_fact to the role called sales\_admin\_role.

VMartDB=> GRANT DELETE ON inventory\_fact TO sales\_admin\_role;

GRANT PRIVILEGE

1. Verify the creation of the following roles and that privileges were assigned to them.

* sales\_general\_role - SELECT on inventory\_fact
* sales\_admin\_role - DELETE on inventory\_fact
* sales\_role - INSERT on inventory\_fact

1. From vsql, verify role creation and privileges using \z

VMartDB=> \z

Access privileges for database "VMartDB"

Grantee | Grantor | Privileges | Schema | Name

-------------------+---------+-------------+--------+------

sales\_general\_role | dbadmin | SELECT | public | inventory\_fact

sales\_role | dbadmin | INSERT | public | inventory\_fact

sales\_admin\_role | dbadmin | DELETE | public | inventory\_fact

## Assign Users to Roles

In this exercise, you will give a user privileges to a role, show and set the role within the database.

1. From vsql, grant the user Jane priveleges to the role called sales\_admin\_role.

VMartDB=> GRANT sales\_admin\_role TO Jane;

GRANT ROLE

1. Change the access to the database to Jane by typing the following command. The password is password2.

VMartDB=> \c VMartDB Jane;

You are now connected to database "VMartDB" as user "Jane".

1. Show enabled roles command to show that no role have been enabled for Jane.

VMartDB=> show enabled roles;

name | setting

---------------+---------

enabled roles |

1. Set role sales\_admin\_role to enable the role in the database.

VMartDB=> set role sales\_admin\_role;

SET

1. Show enabled roles command to show that the sales\_admin\_role has been enabled.

VMartDB=> show enabled roles

name | setting

---------------+-----------------

enabled roles | sales\_admin\_role

1. Change the access to the database back to DBAdmin by typing the following command. The password is *password*.

VMartDB=> \c VMartDB DBAdmin;

You are now connected to database "VMartDB" as user "DBadmin".

## Assign Roles to Other Roles

In this exercise, you will give a role the same privileges as another role.

1. From vsql, grant the role called sales\_admin\_role to have the same privileges as the role called sales\_general\_role and the role called sales\_role.

VMartDB=> GRANT sales\_general\_role, sales\_role TO sales\_admin\_role;

GRANT ROLE

,

1. Verify the sales\_admin\_role has been granted to sales\_general\_role and sales\_role
2. From vsql, verify the privileges of sales\_general\_role and sales\_role using \dp

VMartDB=> \dp

Access privileges for database "VMartDB"

Grantee | Grantor |Privileges | Schema | Name

-------------------+---------+-----------+--------+------------------

sales\_admin\_role | dbadmin | | |sales\_general\_role

sales\_admin\_role | dbadmin | | |sales\_role

## Validating: New Users, Roles and Privileges

After completing this exercise, the following should have taken place within your training environment.

1. **Users** Two new users have been set up. Fred with a password identified by password and Jane identified with a password of *password2*
   * Jane has been granted sales\_admin\_role access.
2. **Access to Schemas** Fred and Jane both have access to the public schema
3. **Roles and Privileges-**
   * New roles created - sales\_general\_role, sales\_role, sales\_admin\_role
   * New privileges created -
     1. SELECT ON Inventory\_fact to sales\_general\_role
     2. INSERT ON Inventory\_fact to sales\_role
     3. DELETE ON inventory\_fact to sales\_admin\_role
4. **Assign Roles to other Roles-**
   * Sales Admin Role has been granted access to sales\_general\_role and sales\_admin\_role

Based on the above criteria. Answer the following question.

1. Type the following command to access the VMartDB as Jane. (The password is *password2*)

VMartDB=> \c VMartDB 'Jane'

You are now connected to database "VMartDB" as user "jane".

1. We want to verify that user Jane, can run a select statement on the database. Type in the following command.

VMartDB=> select count(\*) from inventory\_fact

count

--------

600000



# Resource Management

In this module you will work with resource pools. You will create a resource pool, create a user, and then assign that user to the pool. You will then run a query against that pool, have that query fail and find the failure in resource\_rejections table.

Learning Objectives:

* Create a resource pool
* Create a user
* Make a query abort from lack of resources
* Monitor the rejection in the resource\_rejections system table

## Creating Resource Pools

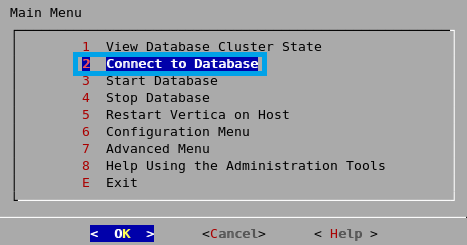
New resource pools can be created, associated to users, and monitored.

1. Change to the directory /opt/vertica/examples/VMart\_Schema, and then launch the adminTools interface.

$ cd /opt/vertica/examples/VMart\_Schema

$ adminTools

Select Connect to Database to launch a vsql session. (Note: If the database asks you for a password, the password is *password.)*



Resource Management (Continued)

1. From vsql, create a new resource pool, called low\_memory.

VMartDB=> CREATE RESOURCE POOL low\_memory memorysize '5M' maxmemorysize '10M';

CREATE RESOURCE POOL

1. Turn on expanded display and query the resource\_pools system table to see the new entry.

VMartDB=> \x

Expanded display is on

1. Select the low memory resource pool you just created.

VMartDB=> SELECT \* FROM resource\_pools WHERE name = 'low\_memory';

-[ RECORD 1 ]------------+------------------

pool\_id | 45035996273896072

name | low\_memory

is\_internal | f

memorysize | 5M

maxmemorysize | 10M

executionparallelism | AUTO

priority | 0

runtimepriority | MEDIUM

runtimeprioritythreshold | 2

queuetimeout | 300

plannedconcurrency | AUTO

maxconcurrency |

runtimecap |

singleinitiator | f

cpuaffinityset |

cpuaffinitymode | ANY

Note the null values where options were not specified. These values are treated as default values.

Resource Management (Continued)

## Associate a Resource Pool with a User

You will create a new user. In the same CREATE USER command you will also assign the user a default resource pool from which query resources will come.

1. Create a new user called test with *password3* as the password, and associate the test user to the low\_memory resource pool.

VMartDB=> CREATE USER Carlos IDENTIFIED BY 'password3' RESOURCE POOL low\_memory;

CREATE USER

1. Query the user's system table to see the new entry.

VMartDB=> SELECT \* from users WHERE user\_name = 'Carlos';

-[ RECORD 1 ]-----+--------------------------------------------------

user\_id | 45035996273896110

user\_name | Carlos

is\_super\_user | f

profile\_name | default

is\_locked | f

lock\_time |

resource\_pool | low\_memory

memory\_cap\_kb | unlimited

temp\_space\_cap\_kb | unlimited

run\_time\_cap | unlimited

all\_roles |

default\_roles |

search\_path | "$user", public, v\_catalog, v\_monitor, v\_internal

Resource Management (Continued)

## Grant Object Level Privileges to a User

You have assigned a certain level of resources to Carlos by associating him with a resource pool. But, you also must assign privileges to Carlos for the transactions that we will need to run with those resources.

1. Grant the user usage permissions on the appropriate schemas. These permissions are part of Vertica user setup.

VMartDB=> GRANT USAGE ON SCHEMA public to Carlos;

GRANT PRIVILEGE

VMartDB=> GRANT USAGE ON SCHEMA online\_sales to Carlos;

GRANT PRIVILEGE

1. Grant the user select permissions on the appropriate tables to enable the user to query these tables. These permissions are part of Vertica user setup.

VMartDB=> GRANT SELECT ON online\_sales.online\_sales\_fact to Carlos;

GRANT PRIVILEGE

VMartDB=> GRANT SELECT ON public.customer\_dimension to Carlos;

GRANT PRIVILEGE

1. Log into the VMartDB as Carlos with the following command. The password is *password3*

VMartDB=> \c VMartDB Carlos

You are now connected to database "VMartDB" as user "Carlos".

Resource Management (Continued)

1. The query below requires more than the current maximum memory for users associated to the low\_memory resource pool. Run the query:

VMartDB=> SELECT CD.\*,OSF.sale\_date\_key FROM online\_sales.online\_sales\_fact OSF INNER

JOIN customer\_dimension CD ON CD.customer\_key = OSF.customer\_key AND CD.customer\_key % 5= 0 LIMIT 1000;

ERROR 3587: Insufficient resources to execute plan on pool low\_memory [Request Too Large:Memory(KB) Exceeded: Requested = 36452, Free = 10240 (Limit = 10240, Used = 0)]

1. Switch back to dbadmin user with the following command. The password is *password*.

VMartDB=> \c VMartDB DBAdmin

You are now connected to database "VMartDB" as user "Carlos".

1. Query the resource\_rejections system table to see the entry for the rejected request.

VMartDB=> SELECT \* FROM resource\_rejections;

node\_name | pool\_id | pool \_name | reason |resource\_type | rejection\_count | first \_rejected\_timestamp |last\_rejected\_timestamp | last\_rejected\_value

v\_vmart\_db\_node001 | 45035996273819852 |low\_memory | Request exceeded high limit | Memory (KB) | 3 | 2014-01-09 16:58:43.690609.05 |2014-01-09 16:59:43.897684.05 | 102400

v\_vmart\_db\_node002 | 45035996273819852 |low\_memory | Request exceeded high limit | Memory (KB) | 3 | 2014-01-09 16:58:43.690609.05 |2014-01-09 16:59:43.897684.05 | 48898

v\_vmart\_db\_node003 | 45035996273819852 |low\_memory | Request exceeded high limit | Memory (KB) | 3 | 2014-01-09 16:58:43.690609.05 |2014-01-09 16:59:43.897684.05 |48898

## Resource Management Question

* How can we resolve the insufficient resource error related to this query?

Click [**here**](#ResourceManagementAK) to go to the answer key



# Appendix

# Installation and Setup

In this module, you will install Vertica and build the VMart sample database.

Learning Objectives:

* Install Vertica
* Install the VMart example
* Create a new database
* Define schemas and tables

## Unpacking the Vertica Installation Package and Installing Vertica

1. Access a terminal window by clicking on the  icon at the top of the window and log in as root:

$ su -

password: <root-password>

#

In this environment:

* The password for the user root is *vertica*
* The password for the user dbadmin is *dbadmin*

1. Run the rpm package installer:

# rpm -Uvh /class\_files/vertica-5.0.4-0.i386.RHEL5.rpm

Preparing... ########################################### [100%]

1:vertica ########################################### [100%]

Vertica Analytic Database V5.0.4-0 successfully installed on host vertica

Important Information

If you are upgrading from a previous version, you must backup your database before

continuing with this install. After restarting your database, you will be unable

to revert to a previous version of the software.

To download the latest Vertica documentation in zip or tar format:

http://www.vertica.com/v-zone/product\_documentation

To complete installation and configuration of the cluster,

run: /opt/vertica/sbin/install\_vertica

## Running the install\_vertica Script

The install\_vertica script installs the application on the machines identified in the command line. The script sets some Linux configuration parameters and checks the connectivity and bandwidth characteristics of the communication links among the cluster.

1. Run the install\_vertica script:

# /opt/vertica/sbin/install\_vertica --hosts 127.0.0.1 --rpmr /class\_files/ vertica-5.0.4-0.i386.RHEL5.rpm

Vertica Analytic Database 5.0.4-0 Installation Tool

Upgrading admintools meta data format..

scanning /opt/vertica/config/users

Starting installation tasks...

Getting system information for cluster (this may take a while)....

Checking/fixing OS parameters.....

Setting vm.min\_free\_kbytes to 4096 ...

Creating/Checking Vertica DBA group

Creating/Checking Vertica DBA user

Setting /etc/security/limits.conf nproc to 2011 ...

Creating Vertica Data Directory...

Updating spread configuration...

Verifying spread configuration on whole cluster.

Creating node node0001 definition for host 127.0.0.1

... Done

Error Monitor 0 errors 0 warnings

Installation complete.

To create a database:

1. Logout and login as dbadmin.\*\*

2. Run /opt/vertica/bin/adminTools as dbadmin

3. Select Create Database from the Configuration Menu

\*\* The installation modified the group privileges for dbadmin.

If you used sudo to install vertica as dbadmin, you will

need to logout and login again before the privileges are

applied.

## Installing the VMart Example Data

1. Switch to the dbadmin user account. If this account did not already exist, it would have been created by the install\_vertica script:

# su - dbadmin

1. Navigate to the VMart examples directory:

$ cd /opt/vertica/examples/VMart\_Schema

1. Run the sample data generator program:

$ ./vmart\_gen

The program will take some time to run, and may seem to stop.

Using default parameters

datadirectory = ./

numfiles = 1

seed = 20177

null = ''

timefile = Time.txt

numfactsalesrows = 5000000

numfactorderrows = 300000

numprodkeys = 60000

numstorekeys = 250

numpromokeys = 1000

numvendkeys = 50

numcustkeys = 50000

numempkeys = 10000

numwarehousekeys = 100

numshippingkeys = 100

numonlinepagekeys = 1000

numcallcenterkeys = 200

numfactonlinesalesrows = 5000000

numinventoryfactrows = 300000

gen\_load\_script = false

Data Generated successfully!

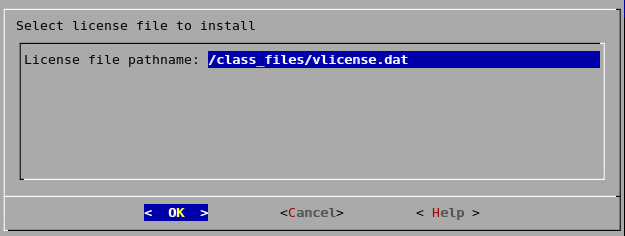
1. Navigate to the appropriate directory

$ /opt/vertica/bin/adminTools

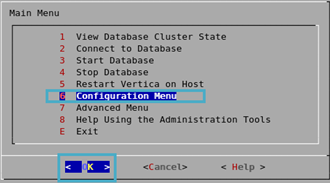
Entering the path information before the call to adminTools directs you to the proper subdirectory when you connect to the database from within adminTools.

## Creating the Sample Database

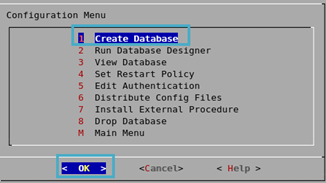
1. Accept the license agreement.
2. Specify the location of your license key file (/class\_files/vlicense.dat) and click OK.



1. From the Main Menu, select Configuration Menu and then click the OK button.

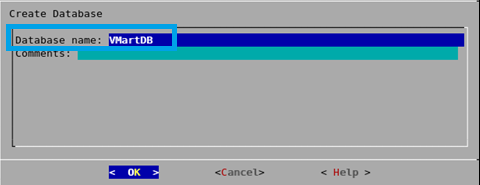


1. Select Create Database and then click the OK button.

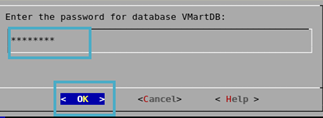


Creating the Sample Database (Continued)

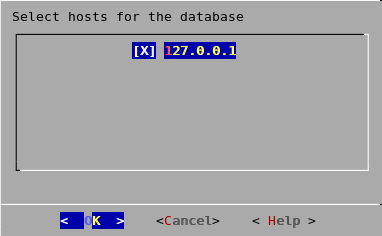
1. Name your database VMartDB, and then click the OK button.



1. Enter and confirm the database password *password* for the new database, and then click the OK button.

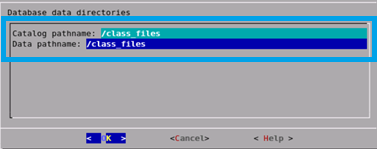


1. Select the host IP address(es) to add node(s) to the cluster, and then click the OK button.



Creating the Sample Database (Continued)

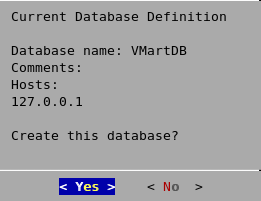
1. Change the pathnames for the data and catalog directories to /class\_files then click the OK button.



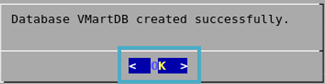
1. Click the OK button to confirm that the database is not going to be k-safe.



1. Click the Yes button to create the database.

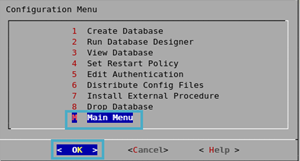


1. Click the OK button to confirm that the database has been created successfully.

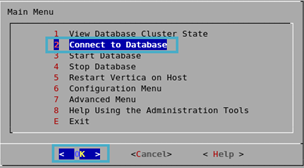


## Defining Schemas and Tables

1. From the adminTools interface, select Main Menu and then click the OK button.



1. Click Connect to Database and then click the OK button. You will see the following prompt.



1. You will be placed in vsql:

Welcome to vsql, the Vertica Analytic Database interactive terminal.

Type: \h or \? for help with vsql commands

\g or terminate with semicolon to execute query

\q to quit

VMartDB=>

Define Schemas and Tables (Continued)

1. Execute the SQL schema definition script using the \i meta-command in vsql:

VMartDB=> \i vmart\_define\_schema.sql

A series of CREATE TABLE and ALTER TABLE statement scrolls on the terminal window:

CREATE SCHEMA

CREATE SCHEMA

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

ALTER TABLE

CREATE TABLE

CREATE TABLE

ALTER TABLE

CREATE TABLE

ALTER TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

ALTER TABLE

The CREATE TABLE command creates the tables and referential integrity constraints that make up the logical schema.

Define Schemas and Tables (Continued)

1. You can check the tables that were created using \dt meta-command in vsql:

VMartDB=> \dt

List of tables

Schema | Name | Kind | Owner | Comment

--------------+-----------------------+-------+---------+---------

online\_sales | call\_center\_dimension | table | dbadmin |

online\_sales | online\_page\_dimension | table | dbadmin |

online\_sales | online\_sales\_fact | table | dbadmin |

public | customer\_dimension | table | dbadmin |

public | date\_dimension | table | dbadmin |

public | employee\_dimension | table | dbadmin |

public | inventory\_fact | table | dbadmin |

public | product\_dimension | table | dbadmin |

public | promotion\_dimension | table | dbadmin |

public | shipping\_dimension | table | dbadmin |

public | vendor\_dimension | table | dbadmin |

public | warehouse\_dimension | table | dbadmin |

store | store\_dimension | table | dbadmin |

store | store\_orders\_fact | table | dbadmin |

store | store\_sales\_fact | table | dbadmin |

# Optional Lab Exercises

## Exercise 1 - Backup and Restore

In this exercise, students will

* Setup the backup environment
* Create the backup configuration file
* Validate the existence of the configuration file and designated backup location on node 1.
* Run a backup
* Truncate data from a table to demonstrate the restore
* Restore the database and validate that the truncated table data was restored

**Setup the backup environment**

1. From a linux prompt, create a backup directory within the tmp directory of node 1.

[dbadmin@verticaTraining32bit ~]$ mkdir /tmp/vbackup

**Create the backup configuration file**

1. From the linux prompt run the vbr.py script to use a wizard to setup a configuration file.

[dbadmin@verticaTraining32bit ~]$ vbr.py --setupconfig

Exercise 1 Backup and Restore (Continued)

1. Enter the information for the backup configuration file as follows [press Enter after each line]:

Snapshot name: **vertica\_backup**

Temp directory: **/tmp**

Do you want to change default vertica binary directory (/opt/vertica/bin)? [y/n] **n**

Do you want to backup vertica configurations? [y/n] **n**

Do you want to change default retry behaviour? [y/n] **n**

Database name: **VMartDB**

Vertica Username: **dbadmin**

Do you want to save password instead of providing password everytime? [y/n] **y**

Vertica password: **password**

Do you want to specify a database port? [y/n] **n**

Do you want to specify a database initiator that vbr connects to? [y/n] **n**

Do you want to encrypt transmission? [y/n] **n**

Do you want to use checksum instead of time&size to check difference? [y/n] **n**

Rsync daemon listening port: 50000

Rsync bandwidth limit(in KBPS, 0 for unlimited): 0

Do you want to hardlink files on same host instead of copying them? [y/n] **n**

Number of cluster nodes: 1

Node #0

Node name: v\_vmartdb\_node0001

Backup host name: 127.0.0.1

Backup directory: /tmp/vbackup

Config file name: /tmp/vertica\_backup.ini

Saved vbr configuration to /tmp/vertica\_backup.ini.

[dbadmin@verticaTraining32bit tmp]$

Exercise 1 Backup and Restore (Continued)

**Validate the existence of the configuration file and designated backup location**

1. Validate that the configuration file (vertica\_backup.ini) and the backup location directory (vbackup) exist in the tmp directory.

[dbadmin@verticaTraining32bit ~]$ ls /tmp

4803 ssh-XvkEPU3082

gconfd-dbadmin **vbackup**

hsperfdata\_root **vertica\_backup.ini**

keyring-a9nEW8 virtual-dbadmin.DngRPR

keyring-izqTtf virtual-dbadmin.fu0VMH

keyring-VNdJQT virtual-dbadmin.uVw9fz

mapping-dbadmin vmware-dbadmin

mapping-root VMwareDnD

vmware-root orbit-dbadmin

[dbadmin@verticaTraining32bit ~]$

Note: The contents of your tmp directory may vary from the results shown above. The key is to identify the presence of the backup configuration file that you just created (vertica\_backup.ini) and the location of the backup directory location specified in the configuration file (vbackup).

Exercise 1 Backup and Restore (Continued)

**Run the backup**

1. Run a backup task using the configuration file that you just created.  Node1 will do all the work of running the backup.

[dbadmin@verticaTraining32bit ~]$ vbr.py --task backup --config-file /tmp/vertica\_backup.ini

Copying...

303896742 out of 303896742, 100%

All child processes terminated successfully.

Committing changes on all backup sites...

backup done!

[dbadmin@verticaTraining32bit ~]$

Notes:   
  
1. When the copy goes to 100%, your backup is complete.

2. If you see the following security note, enter *yes* then press the Enter key.

The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.

RSA key fingerprint is 59:e7:e2:08:aa:7a:5a:93:a6:d7:0d:f4:e4:77:85:fa.

Are you sure you want to continue connecting (yes/no)?

Exercise 1 Backup and Restore (Continued)

**Truncate data from a table (to demonstrate restore in next exercise)**

1. From the command line, connect to the database. (The Vertica dbadmin password is *password*.)

[dbadmin@verticaTraining32bit ~]$ vsql -w password

Welcome to vsql, the Vertica Analytic Database v5.0.4-0 interactive terminal.

Type: \h for help with SQL commands

\? for help with vsql commands

\g or terminate with semicolon to execute query

\q to quit

dbadmin=>

1. Truncate the contents of a table to remove its data.

dbadmin=> truncate table store.store\_orders\_fact;

TRUNCATE TABLE

dbadmin=>

Exercise 1 Backup and Restore (Continued)

1. Check the data in the truncated table.

dbadmin=> select \* from store.store\_orders\_fact;

product\_key | product\_version | store\_key | vendor\_key |   
employee\_key | order\_number | date\_ordered | date\_shipped |   
expected\_delivery\_date | date\_delivered | quantity\_ordered | quantity\_delivered | shipper\_name | unit\_price | shipping\_cost | total\_order\_cost | quantity\_in\_stock | reorder\_level |   
overstock\_ceiling

-------------+-----------------+-----------+------------+--------------+--------------+--------------+--------------+------------------------+----------------+------------------+--------------------+--------------+------------+---------------+------------------+-------------------+---------------+-------------------

**(0 rows)**

dbadmin=>

Note: There is no data in this table because it was truncated in the previous step.

1. Exit out of vsql back to the command line.

dbadmin=> \q

[dbadmin@verticaTraining32bit ~]$

Note: In the next exercise, you will use vbr.py to restore the database. Then, you will check to see whether the data was restored.

Exercise 1 Backup and Restore (Continued)

**Restore the database**

1. Close all running sessions.

[dbadmin@verticaTraining32bit ~]$ vsql -w password -c "select close\_all\_sessions();"

Note: To perform a restore, it is necessary to stop the database. The database cannot shut down while there are sessions running. This command ensures that there are no sessions running so that you can stop the database.

close\_all\_sessions

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

Close all sessions command sent. Check v\_monitor.sessions for progress.

(1 row)

[dbadmin@verticaTraining32bit ~]$

Exercise 1 Backup and Restore (Continued)

1. Stop the database using the command line.

[dbadmin@verticaTraining32bit ~]$ admintools -t stop\_db -p password -d VMartDB

Note: You can also use adminTools interface to stop the database. But, here we are using adminTools at the command line.   
  
For more information about adminTools command line functions, refer to the page titled “Writing Administration Tools Scripts” in the HP Vertica Administrators Guide.

Issuing shutdown command to database

Database VMartDB stopped successfully

[dbadmin@verticaTraining32bit ~]$

1. From the Linux prompt, restore the tables by running a restore task that references the same configuration file that was used to backup the tables.   
     
   Note: The Vertica dbadmin password is *password*.

[dbadmin@verticaTraining32bit ~]$ vbr.py --task restore --config-file /tmp/vertica\_backup.ini

Enter vertica password for user dbadmin:

Copying...

9836725 out of 9836725, 100%

All child processes terminated successfully.

restore done!

[dbadmin@verticaTraining32bit ~]$

Exercise 1 Backup and Restore (Continued)

**Validate the restore**

1. Restart the database from the command line.

[dbadmin@verticaTraining32bit ~]$ admintools -t start\_db -p password -d VMartDB

Node Status: v\_vmartdb\_node0001: (DOWN)

Node Status: v\_vmartdb\_node0001: (INITIALIZING)

Node Status: v\_vmartdb\_node0001: (INITIALIZING)

Node Status: v\_vmartdb\_node0001: (UP)

Database VMartDB started successfully

[dbadmin@verticaTraining32bit ~]$

1. From the command line, run a query that counts the number of rows in the table that you previously truncated.

[dbadmin@verticaTraining32bit ~]$

vsql -w password -c "select count(\*) from store.store\_orders\_fact;"

count

--------

300000

(1 row)

[dbadmin@verticaTraining32bit ~]$

Note: There is now data in the truncated table, demonstrating that the data has been restored.

# Answer Key

## Data Loading Questions

1. Is it possible to load data into a Vertica system without using WOS? What are the advantages and disadvantages of such a decision?

* **Yes, all data is loaded directly into ROS, using the DIRECT keyword. This is used in organizations that have regular bulk loads and little trickle.**
* **You can set the size of the WOSDATA pool to 0, but it is recommended that it be kept around 100M so that delete vectors can be initially written to WOS.**

1. What is the difference between partitioning and segmentation?

* **Segmentation is used for data distribution across nodes; partitioning is used for data distribution on a node, allowing for easy deletion by removal of a partition.**
* **Segmentation is a projection property; partitioning is a table property**.

1. The data in WOS exists only in RAM. What are some possible issues associated with that?

* **Data loss in the event of power outage or other catastrophic problems**

Click [**here**](#SectionII) to go back to exercise

## Removing, Deleting, and Purging Data Question

1. What are the advantages and disadvantages of creating delete vectors to identify deleted records?
   * + - 1. **Advantages - Allows deletion to run faster**
         2. **Disadvantages - Potential delete vectors collecting which slows down querying and Recovery. (You do not regain disk space of deletes.**

Click [**here**](#SectionIII) to go to back to the exercise

## Resource Management Question

1. How can we resolve the insufficient resource error related to this query?

**Use the alter resource pool command.**

**VMartDB=> alter resource pool low\_memory maxmemorysize '100M';**

Click [**here**](#ResourceManagement) to go to back to the exercise