Getting Started Guide

HP Vertica Analytic Database

Software Version: 7.0.x

Document Release Date: 2/24/2014

Contents

Getting Started Guide Overview

The purpose of the Getting Started Guide is to show you how to set up an HP Vertica example database and run simple queries that perform common database tasks.

Who Should Use This Guide

The Getting Started Guide targets anyone who wants to learn how to create and run an HP Vertica database. This guide requires no special knowledge at this point, although a rudimentary knowledge of basic SQL commands is useful when you begin to run queries.

What You Need

The examples provided in this guide require that you:

- Installed HP Vertica on one host or a cluster of hosts. Hewlett-Packard recommends a minimum of three hosts in the cluster.

OR

- Obtained a Virtual Machine (VM) with HP Vertica installed on it.

For further instructions regarding installation, see the Installation Guide.

Accessing Your Database

You access your database either using an SSH client or through the terminal utility in your Linux Console, such as vsql. Throughout this guide you use the following user interfaces:

- The Linux command line (shell) interface
- The HP Vertica Administration Tools (See Running the Administration Tools in this guide for details.)
- The vsql client interface (See Using vsql in the Programmer’s Guide for details.)

Downloading and Starting the Virtual Machine

HP Vertica is available as a Virtual Machine (VM) that is pre-installed on a 64-bit CentOS image and comes with a license for 500 GB of data storage.

Important: HP Vertica virtual machines are not supported in a production environment. The VMs are provided for evaluation purposes.

The VM image is preconfigured with the following hardware settings:

- 1 CPU
- 1024 MB RAM
- 50 GB Hard Disk (SCSI, not preallocated, single file storage)
- Bridged Networking

Downloading a VM

The HP Vertica VM is available both as an OVF template (for VMWare vSphere 4.0) and as a VMDK file (for VMWare Server 2.0 and VMWare Workstation 7.0). Download and install the appropriate file for your VMWare deployment from the myVertica portal at http://www.vertica.com/documentation (registration required).

Starting the VM

1. Open the appropriate HP Vertica VM image file in VMWare. For example, open the VMX file if you are using VMWare Workstation, or the OVF template if you are using VMWare vSphere.
2. Navigate to the settings for the VM image and adjust the network settings so that they are compatible with your VM.
3. Start the VM. For example, in VMWare Workstation, select **VM > Power > Power On**.

Checking for HP Vertica Updates

The VM image might not include the latest available HP Vertica release. After you install and start your VM, verify the version of HP Vertica with the following command.

```
$ rpm -qa | grep vertica
```

The RPM package name that the command returns contains the version and build numbers. If there is a later version of HP Vertica, download it from the myVertica portal at https://my.vertica.com/downloads (registration required). Upgrade instructions are provided in the Installation Guide.

Types of Database Users

Every HP Vertica database has one or more users. When users connect to a database, they must log on with valid credentials (username and password) that a database administrator defines.

Database users own the objects they create in a database, such as tables, procedures, and storage locations. By default, all users have the right to create temporary tables in a database.

In an HP Vertica database, there are three types of users:

- Database administrator (dbadmin)
- Object owner
- Everyone else (PUBLIC)

dbadmin User

When you create a new database, a single database administrator account, dbadmin, is automatically created along with a PUBLIC role. The database administrator bypasses all permission checks and has the authority to perform all database operations, such as bypassing all GRANT/REVOKE authorizations and any user granted PSEUDOSUPERUSER role.

Note: Although the dbadmin user has the same name as the Linux database administrator account, do not confuse the concept of a database administrator with a Linux superuser (root) privilege; they are not the same. A database administrator cannot have Linux superuser privileges.

Object Owner

An object owner is the user who creates a particular database object; the owner can perform any operation on that object. By default, only an owner or a database administrator can act on a database object. In order to allow other users to use an object, the owner or database administrator must grant privileges to those users using one of the GRANT statements. Object owners are PUBLIC users for objects that other users own.

PUBLIC User

All non- administrator and non-object owners are PUBLIC users. Newly created users do not have access to schema PUBLIC by default. Make sure to GRANT USAGE ON SCHEMA PUBLIC to all users you create.

Logging in as dbadmin

The first time you boot the VM you are automatically logged in and a web page displays further instructions. To log back into the VM, use the following username and password.

- `Username: dbadmin`
- `Password: password`
- `Root Password: password`

Important: The dbadmin user has sudo privileges. Be sure to change the dbadmin and root passwords with the Linux `passwrd` command.

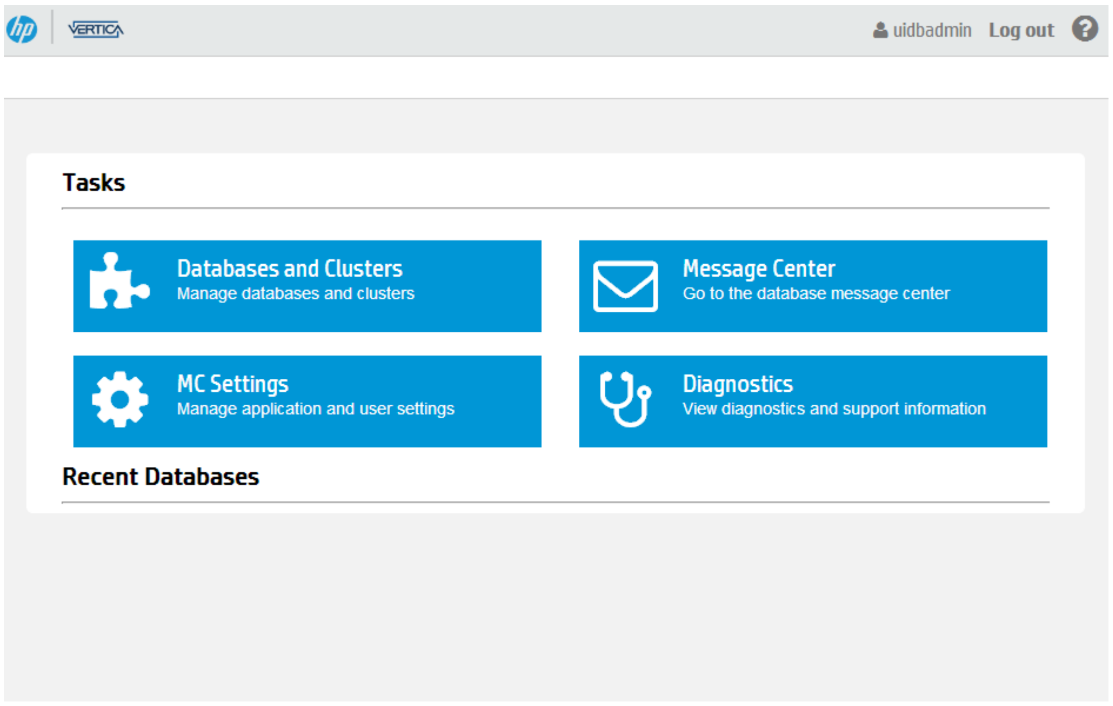
Using the HP Vertica Interfaces

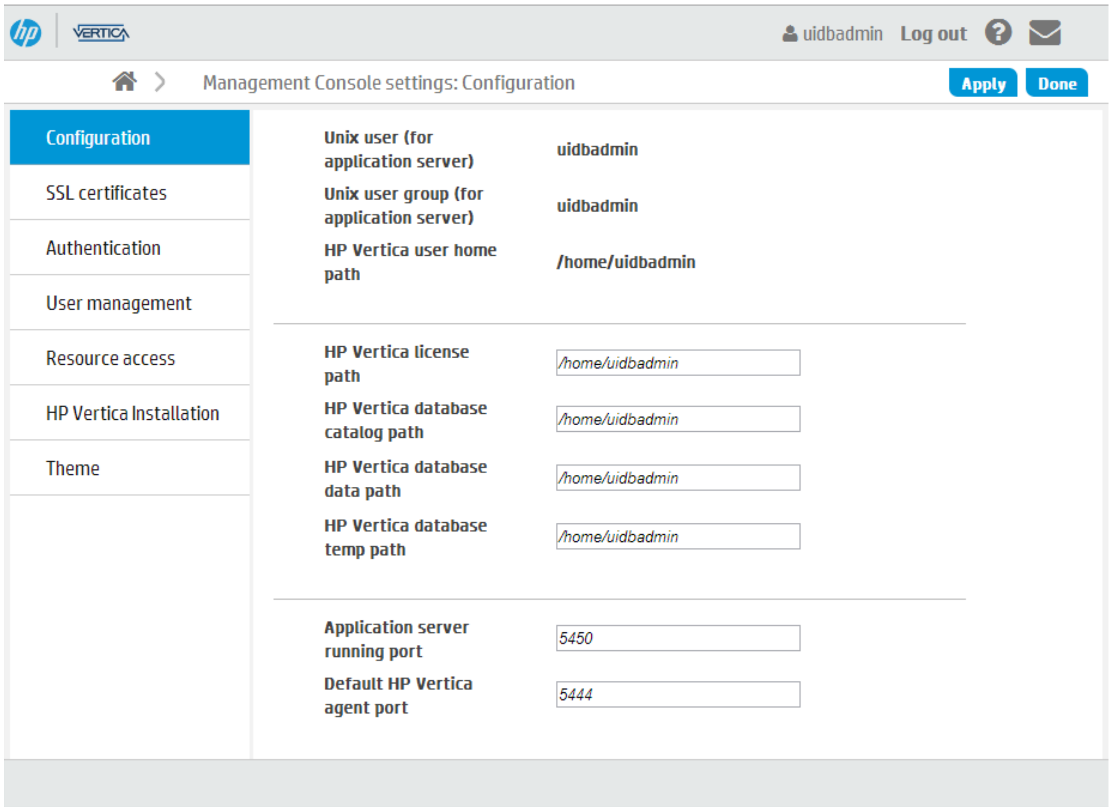
HP Vertica provides a set of tools that allows you to perform administrative tasks quickly and easily. The administration tasks in HP Vertica can be done using the Management Console (MC) or the Administration Tools. The MC provides a unified view of your HP Vertica cluster through a browser connection, while the Administration Tools are implemented using Dialog, a graphical user interface that works in terminal (character-cell) windows.

Using Management Console

The Management Console provides some, but not all, of the functionality that the Administration Tools provides. In addition, the MC provides extended functionality not available in the Administration Tools, such as a graphical view of your HP Vertica database and detailed monitoring charts and graphs.

Most of the information you need to use MC is available on the MC interface, as seen in the following two screenshots. For installation instructions, see Installing and Configuring Management Console in the Installation Guide. For an introduction to MC functionality, architecture, and security, see Management Console in the Concepts Guide.


uidbadmin
Log out
Tasks
Databases and Clusters
Manage databases and clusters
Message Center
Go to the database message center
MC Settings
Manage application and user settings
Diagnostics
View diagnostics and support information
Recent Databases



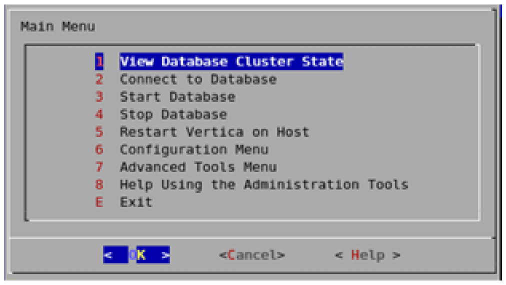
Unix user (for application server)	uidbadmin
Unix user group (for application server)	uidbadmin
HP Vertica user home path	/home/uidbadmin
HP Vertica license path	/home/uidbadmin
HP Vertica database catalog path	/home/uidbadmin
HP Vertica database data path	/home/uidbadmin
HP Vertica database temp path	/home/uidbadmin
Application server running port	5450
Default HP Vertica agent port	5444

Running the Administration Tools

A man page is available for convenient access to Administration Tools details. If you are running as the dbadmin user, simply type `man admintools`. If you are running as a different user, type `man -M /opt/vertica/man admintools`. If possible, always run the Administration Tools using the database administrator account (dbadmin) on the administration host.

The Administration Tools interface responds to mouse clicks in some terminal windows, particularly local Linux windows, but you might find that it responds only to keystrokes. For a quick reference to keystrokes, see Using Keystrokes in the Administration Tools Interface in this guide.

When you run Administration Tools, the **Main Menu** dialog box appears with a dark blue background and a title on top. The screen captures used in this documentation set are cropped down to the dialog box itself, as shown in the following screenshot.



```
Main Menu

   1  View Database Cluster State
   2  Connect to Database
   3  Start Database
   4  Stop Database
   5  Restart Vertica on Host
   6  Configuration Menu
   7  Advanced Tools Menu
   8  Help Using the Administration Tools
   E  Exit

        <  OK  >      <Cancel>     < Help >
```

First Time Only

The first time you log in as the database administrator and run the Administration Tools, complete the following steps.

1. In the EULA (end-user license agreement) window, type `accept` to proceed. A window displays, requesting the location of the license key file you downloaded from the HP Web site. The default path is `/tmp/vlicense.dat`.
2. Type the absolute path to your license key (for example, `/tmp/vlicense.dat`) and click **OK.**
3. To return to the command line, select **Exit** and click **OK**.

Using Keystrokes in the Administration Tools Interface

The following table is a quick reference to keystroke usage in the Administration Tools interface. See Using the Administration Tools in the Administrator's Guide for full details.

Return	Run selected command.
Tab	Cycle between **OK, Cancel, Help**, and menu.
Up/Down Arrow	Move cursor up and down in menu, window, or help file.
Space	Select item in list.
Character	Select corresponding command from menu.

Introducing the VMart Example Database

HP Vertica ships with a sample multi-schema database called the VMart Example Database, which represents a database that might be used by a large supermarket (VMart) to access information about its products, customers, employees, and online and physical stores. Using this example, you can create, run, optimize, and test a multi-schema database.

The VMart database contains the following schema:

- `public` (automatically created in any newly created HP Vertica database)
- `store`
- `online_Sales`

VMart Database Location and Scripts

If you installed HP Vertica from the RPM package, the VMart schema installed in the `/opt/vertica/examples/VMart_Schema` directory. This folder contains the following script files that you can use to get started quickly. Use the scripts as templates for your own applications.

Script/file name	Description
`vmart_count_data.sql`	SQL script that counts rows of all example database tables, which you can use to verify load.
`vmart_define_scehma.sql`	SQL script that defines the logical schema for each table and referential integrity constraints.
`vmart_gen.cpp`	Data generator source code (C++).
`vmart_gen`	Data generator executable file.
`vmart_load_data.sql`	SQL script that loads the generated sample data to the corresponding tables using COPY DIRECT.

`vmart_ queries.sql`	SQL script that contains concatenated sample queries for use as a training set for the Database Designer.
`vmart_query_##.sql`	SQL scripts that contain individual queries; for example, `vmart_query_01` through `vmart_query_09.sql`
`vmart_schema_drop.sql`	SQL script that drops all example database tables.

For more information about the schema, tables, and queries included with the VMart example database, see the Appendix.

Installing and Connecting to the VMart Example Database

Follow the steps in this section to create the fully functioning, multi-schema VMart example database that you'll use to run sample queries. The number of example databases you create within a single HP Vertica installation is limited only by the disk space available on your system; however, Hewlett-Packard strongly recommends that you start only one example database at a time to avoid unpredictable results.

HP Vertica provides two options to install the example database:

- A quick installation that lets you create the example database and start using it immediately. See Quick Installation Using a Script in this guide for details. Use this method to bypass the schema and table creation processes and start querying immediately.

- An advanced-but-simple example database installation using the Administration Tools interface. See Advanced Installation in this guide for details. Use this method to better understand the database creation process and practice creating schema and tables, and loading data.

Note: Both installation methods create a database named VMart. If you try both installation methods, you will either need to drop the VMart database you created (see Restoring the Status of Your Host in this guide) or create the subsequent database with a new name. However, Hewlett-Packard strongly recommends that you start only one example database at a time to avoid unpredictable results

This tutorial uses HP Vertica-provided queries, but you can follow the same set of procedures later, when you create your own design and use your own queries file.

After you install the VMart database, the database has started. Connect to it using the steps in Step 3: Connecting to the Database.

Quick Installation Using a Script

The script you need to perform a quick installation is located in `/opt/vertica/sbin` and is called `install_example`. This script creates a database on the default port (5433), generates data, creates the schema and a default superprojection, and loads the data. The folder also contains a `delete_example` script, which stops and drops the database.

1. In a terminal window, log in as the database administrator.

```
$ su dbadmin
```

```
Password: (your password)
```

2. Change to the `/examples` directory.

```
$ cd /opt/vertica/examples
```

3. Run the install script:

```
$ /opt/vertica/sbin/install_example VMart
```

After installation, you should see the following:

```
[dbadmin@localhost examples]$ /opt/vertica/sbin/install_examples VMart
Installing VMart example example database
Mon Jul 22 06:57:40 PDT 2013
Creating Database
Completed
Generating Data. This may take a few minutes.
Completed
Creating schema
Completed
Loading 5 million rows of data. Please stand by.
Completed
Removing generated data files
Example data
```

The example database log files, `ExampleInstall.txt` and `ExampleDelete.txt`, are written to `/opt/vertica/examples/log`.

To start using your database, continue to Connecting to the Database in this guide. To drop the example database, see Restoring the Status of Your Host in this guide.

Advanced Installation

To perform an advanced-but-simple installation, set up the VMart example database environment and then create the database using the Administration Tools or Management Console.

Note: If you installed the VMart database using the quick installation method, you cannot complete the following steps because the database has already been created.

To try the advanced installation, drop the example database (see Restoring the Status of Your Host on this guide) and perform the advanced Installation, or create a new example database with a different name. However, Hewlett-Packard strongly recommends that you install only one example database at a time to avoid unpredictable results.

The advanced installation requires the following steps:

- Step 1: Setting Up the Example Environment
- Step 2: Creating the Example Database
- Step 3: Connecting to the Database

Step 1: Setting Up the Example Environment

1. Stop all databases running on the same host on which you plan to install your example database.

   If you are unsure if other databases are running, run the Administration Tools and select **View Cluster State**. The State column should show DOWN values on pre-existing databases.

   If databases are running, click **Stop Database** in the **Main Menu** of the Administration Tools interface and click **OK**.

2. In a terminal window, log in as the database administrator:

   ```
   $ su dbadmin
   Password:
   ```

3. Change to the `/VMart_Schema` directory.

   ```
   $ cd /opt/vertica/examples/VMart_Schema
   ```

   Do not change directories while following this tutorial. Some steps depend on being in a specific directory.

4. Run the sample data generator.

   ```
   $ ./vmart_gen
   ```

5. Let the program run with the default parameters, which you can review in the README file.

   ```
   Using default parameters
   datadirectory = ./
   numfiles = 1
   seed = 2
   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 !

Using default parameters
datadirectory = ./
numfiles = 1
seed = 2
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 !
```

6. If the `vmart_gen` executable does not work correctly, recompile it as follows, and run the sample data generator script again.

```
$ g++ vmart_gen.cpp -o vmart_gen
$ chmod +x vmart_gen
$ ./vmart_gen
```

Step 2: Creating the Example Database

To create the example database: use the Administration Tools or Management Console, as described in this section.

Creating the Example Database Using the Administration Tools

In this procedure, you create the example database using the Administration Tools. To use the Management Console, go to the next section.

Note: If you have not used Administration Tools before, see Running the Administration Tools in this guide.

1. Run the Administration Tools.

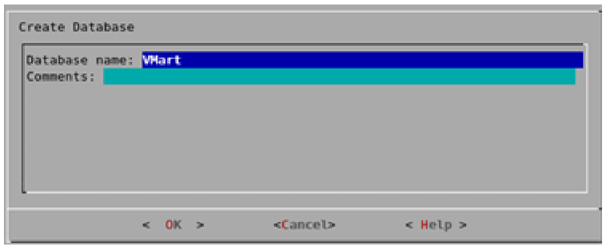
   ```
   $ /opt/vertica/bin/admintools
   ```

   or simply type `admintools`

2. From the Administration Tools **Main Menu**, click **Configuration Menu** and click **OK**.

3. Click **Create Database** and click **OK**.

4. Name the database `VMart` and click **OK**.



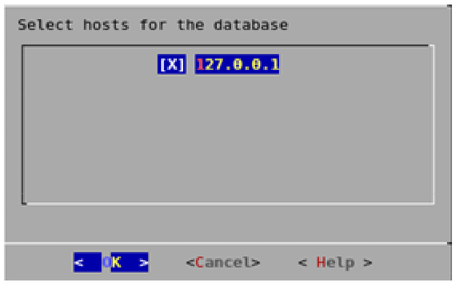
```
Create Database

Database name: VMart
Comments:
```

5. Click **OK** to bypass the password and click **Yes** to confirm.

   There is no need for a database administrator password in this tutorial. When you create a production database, however, always specify an administrator password. Otherwise, the database is permanently set to trust authentication (no passwords).

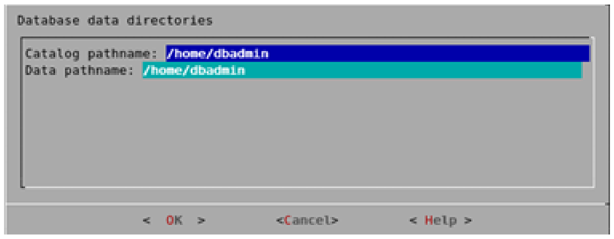
6. Select the hosts you want to include from your HP Vertica cluster and click **OK**.

   This example creates the database on a one-host cluster. Hewlett-Packard recommends a minimum of three hosts in the cluster. If you are using the HP Vertica Community Edition, you are limited to three nodes.



```
Select hosts for the database

[X] 127.0.0.1
```

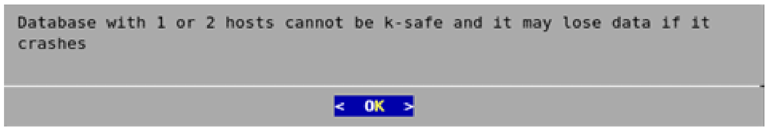
7. Click **OK** to select the default paths for the data and catalog directories.



```
Database data directories

Catalog pathname: /home/dbadmin
Data pathname: /home/dbadmin
```

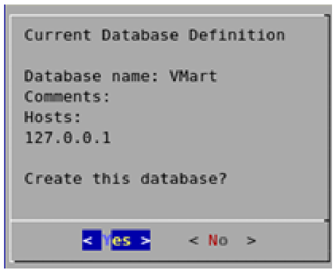
- Catalog and data paths must contain only alphanumeric characters and cannot have leading space characters. Failure to comply with these restrictions could result in database creation failure.
- When you create a production database, you'll likely specify other locations than the default. See Prepare Disk Storage Locations in the Administrator's Guide for more information.

8. Since this tutorial uses a one-host cluster, a K-safety warning appears. Click **OK**.



```
Database with 1 or 2 hosts cannot be k-safe and it may lose data if it
crashes
```

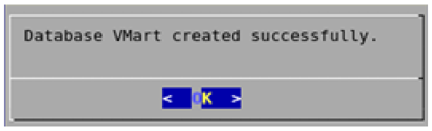
9. Click **Yes** to create the database.



```
Current Database Definition

Database name: VMart
Comments:
Hosts:
127.0.0.1

Create this database?
```

During database creation, HP Vertica automatically creates a set of node definitions based on the database name and the names of the hosts you selected and returns a success message.

10. Click **OK** to close the **Database VMart created successfully** message.



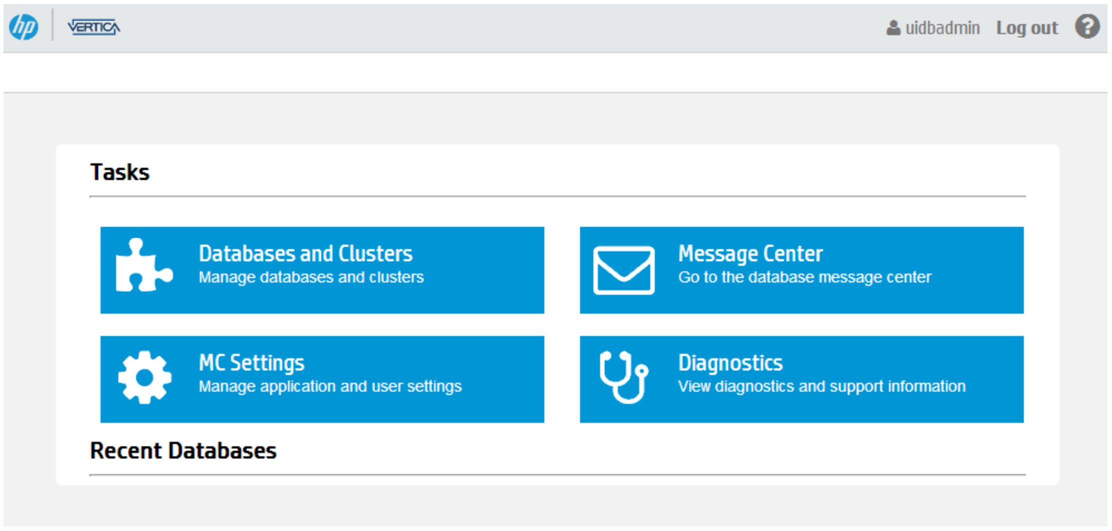
```
Database VMart created successfully.
```

Creating the Example Database Using the Management Console

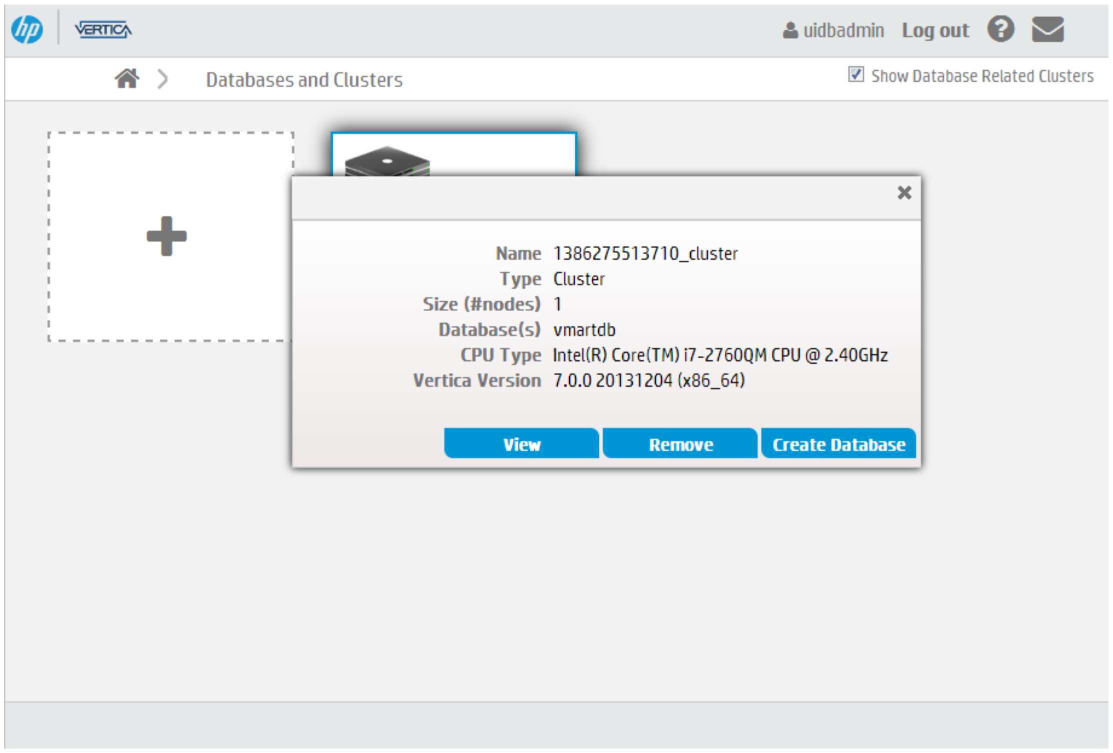
In this procedure, you create the example database using the Management Console. To use the Administration Tools, follow the steps in the preceding section.

> **Note:** To use Management Console, the console should already be installed and you should be familiar with its concepts and layout. See Using Management Console in this guide for a brief overview, or for detailed information, see Management Console in the Concepts Guide and Installing and Configuring Management Console in the Installation Guide.

1. Connect to Management Console and log in.
2. On the Home page, under **Tasks**, click **Database and Clusters**.


uidbadmin
Log out
Tasks
Databases and Clusters
Manage databases and clusters
Message Center
Go to the database message center
MC Settings
Manage application and user settings
Diagnostics
View diagnostics and support information
Recent Databases

3. Click to select the appropriate existing cluster and click **Create Database**.



Name	1386275513710_cluster
Type	Cluster
Size (#nodes)	1
Database(s)	vmartdb
CPU Type	Intel(R) Core(TM) i7-2760QM CPU @ 2.40GHz
Vertica Version	7.0.0 20131204 (x86_64)

4. Follow the on-screen wizard, which prompts you to provide the following information:
   - Database name, which must be between 3–25 characters, starting with a letter, and followed by any combination of letters, numbers, or underscores.
   - (Optional) database administrator password for the database you want to create and connect to.
   - IP address of a node in your database cluster, typically the IP address of the administration host.
5. Click **Next**.

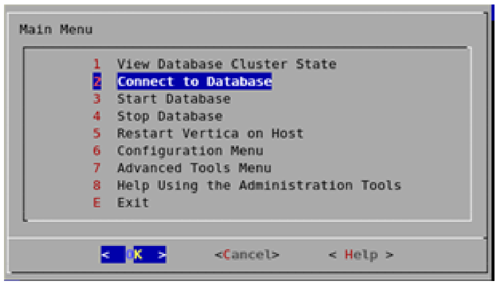
Step 3: Connecting to the Database

Regardless of the installation method you used, follow these steps to connect to the database.

1. As `dbadmin`, run the Administration Tools.

   ```
   $ /opt/vertica/bin/admintools
   ```

   or simply type `admintools`.
2. If you are already in the Administration Tools, navigate to the Main Menu page.
3. Select **Connect to Database**, click **OK**.



```
Main Menu

        1  View Database Cluster State
        2  Connect to Database
        3  Start Database
        4  Stop Database
        5  Restart Vertica on Host
        6  Configuration Menu
        7  Advanced Tools Menu
        8  Help Using the Administration Tools
        E  Exit

          < OK >      <Cancel>      < Help >
```

To configure and load data into the VMart database, complete the following steps:

- Step 4: Defining the Database Schema
- Step 5: Loading Data

If you installed the VMart database using the Quick Installation method, the schema, tables, and data are already defined. You can choose to drop the example database (see Restoring the Status of Your Host in this guide) and perform the Advanced Installation, or continue straight to Querying Your Data in this guide.

Step 4: Defining the Database Schema

The VMart database installs with sample scripts with SQL commands that are intended to represent queries that might be used in a real business. The `vmart_define_schema.sql` script runs a script that defines the VMart schema and creates tables. You must run this script before you load data into the VMart database.

This script performs the following tasks:

- Defines two schemas in the VMart database schema: *online_sales* and *store*.
- Defines tables in both schemas.
- Defines constraints on those tables.

```
Vmart=> \i vmart_define_schema.sql
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
```

Step 5: Loading Data

Now that you have created the schemas and tables, you can load data into a table by running the `vmart_load_data.sql` script. This script loads data from the 15 `.tbl` text files in `opt/vertica/examples/VMart_Schema` into the tables that `vmart_design_schema.sql` created.

It might take several minutes to load the data on a typical hardware cluster. Check the load status by monitoring the `vertica.log` file, as described in Monitoring Log Files in the Administrator's Guide.

```
VMart=> \i vmart_load_data.sql
Rows Loaded
-------------
1826

(1 row)

Rows Loaded
-------------
60000
(1 row)

Rows Loaded
-------------
250
(1 row)

Rows Loaded
-------------
1000
(1 row)

Rows Loaded
-------------
50
(1 row)

Rows Loaded
-------------
50000
(1 row)

Rows Loaded
-------------
10000
(1 row)

Rows Loaded
-------------
```

```
100
(1 row)

Rows Loaded
-------------
100
(1 row)

Rows Loaded
-------------
1000

(1 row)

Rows Loaded
-------------
200
(1 row)

Rows Loaded
-------------
5000000
(1 row)

Rows Loaded
-------------
300000
(1 row)

VMart=>
```

Querying Data

The VMart database installs with sample scripts that contain SQL commands that represent queries that might be used in a real business. Use basic SQL commands to query the database, or try out the following command. Once you're comfortable running the example queries, you might want to write your own.

Note: The data that your queries return might differ from the example output shown in this guide because the sample data generator is random.

Type the following SQL command to return the values for five products with the lowest fat content in the Dairy department. The command selects the fat content from Dairy department products in the `product_dimention` table in the `public` schema, orders them from low to high and limits the output to the first five (the five lowest fat contents).

```
VMart => SELECT fat_content
         FROM ( SELECT DISTINCT fat_content
                FROM product_dimension
                WHERE department_description
                IN ('Dairy') ) AS food
         ORDER BY fat_content
         LIMIT 5;
```

Your results will be similar to the following.

```
 fat_content
-------------
80
81
82
83
84
(5 rows)
```

The preceding example is from the `vmart_query_01.sql` file. You can execute more sample queries using the scripts that installed with the VMart database or write your own. For a list of the sample queries supplied with HP Vertica, see the Appendix.

Backing Up and Restoring the Database

HP Vertica supplies a comprehensive utility, the `vbr.py` Python script, that lets you back up and restore a full database, as well as create snapshots of specific schema or tables. The vbr.py utility creates backup directories during its initial execution; subsequently running the utility creates subdirectories.

The following information is intended to introduce the backup and restore functions. For more detailed information, see Backing Up and Restoring the Database in the *Administrator's Guide*.

Backing Up the Database

Use `vbr.py` to save your data to a variety of locations:

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

Note: Creating a database backup on a different cluster does not provide disaster recovery. The cloned database you create with `vbr.py` is entirely separate from the original, and is not kept synchronized with the database from which it is cloned.

When to Back up the Database

In addition to any guidelines established by your organization, Hewlett-Packard recommends that you back up your database:

- Before you upgrade HP Vertica to another release.
- Before you drop a partition.
- After you load a large volume of data.
- If the epoch in the latest snapshot is earlier than the current ancient history mark (AHM).
- Before and after you add, remove, or replace nodes in your database cluster.
- After recovering a cluster from a crash.

Note: When you restore a database snapshot, you must restore to a cluster that is identical to the one wherre you created the snapshot. For this reason, always create a new snapshot after adding, removing, or replacing nodes.

Ideally, create regular backups of your full database. You can run the HP Vertica `vbr.py` utility from a cron job or other task scheduler.

Creating the Backup Configuration File

The `vbr.py` utility uses a configuration file for the information required to back up and restore a full- or object-level snapshot. The configuration file defines where the database backup is saved, the temporary directories it uses, and which nodes, schema, and/or tables in the database are to be backed up. You cannot run `vbr.py` without a configuration file, and no default file exists.

To invoke the script to set up a configuration file, enter this command:

```
$ vbr.py --setupconfig
```

The script prompts you to answer the following questions regarding the configuration file. Type **Enter** to accept the default value in parentheses. See VBR Configuration File Reference in the Administrator's Guide for information about specific questions.

```
Snapshot name (backup_snapshot): Example_backup
Backup vertica configurations? (n) [y/n]: y
Number of restore points (1): 1
Specify objects (no default):
Vertica user name (dbadmin): dbadmin
Save password to avoid runtime prompt? (n) [y/n]: y
Password to save in vbr config file (no default): password
Node v_vmart_node0001
Backup host name (no default): localhost
Backup directory (no default): /home/dbadmin
Config file name (backup_snapshot.ini): exampleBackup.ini
Change advanced settings? (n) [y/n]: n
Saved vbr configuration to exampleBackup.ini.
```

After you answer the required questions, `vbr.py` generates a configuration file with the information you supplied. Use the `Config file name` you specified when you run the `--task backup` or other commands. The `vbr.py` utility uses the configuration file contents for both backup and restore tasks.

Creating Full and Incremental Backups

Before you create a database backup, ensure the following:

- Your database is running.
- All of the backup hosts are up and available.
- The backup location host has sufficient disk space to store the snapshots.
- The user who starts the utility has write access to the target directories on the host backup location.

Run the `vbr.py` script from a terminal using the database administrator account from an initiator node in your database cluster. You cannot run the utility as root.

Use the `--task backup` and `--config-file` *filename* directives as shown in this example.

```
$ vbr.py --task backup --config-file exampleBackup.ini
Copying…
[==============================================] 100%
All child processes terminated successfully.
Committing changes on all backup sites…
backup done!
```

By default, there is no screen output other than the progress bar.

If you do not specify a configuration file, the `vbr` utility searches for one at this location:

```
/opt/vertica/config/vbr.ini
```

If the utility does not find a configuration file at this location, it fails with an error and exits.

The first time you run the `vbr.py` utility, it performs a full backup; subsequent runs with the same configuration file create an incremental snapshot. When creating incremental snapshots, the utility copies new storage containers, which can include data that existed the last time you backed up the database, along with new and changed data since then. By default, `vbr.py` saves one archive backup, unless you set the `restorePointLimit` parameter value in the configuration file to a value greater than `1`.

Restoring the Database

To restore a full database snapshot, ensure that:

- The database is down.
- All of the backup hosts are up and available.
- The backup directory exists and contains the snapshots from which to restore.
- The cluster to which you are restoring the backup has the same number of hosts as the one used to create the snapshot; the node names and the IP addresses must also be identical.
- The database you are restoring already exists on the cluster to which you are restoring data; the database can be completely empty, without any data or schema. As long as the database name matches the name in the snapshot, and all of the node names match the names of the nodes, you can restore to it.

To begin a full database snapshot restore, log in using the database administrator's account. You cannot run the utility as root.

To restore the most recent snapshot, use the configuration file used to create the snapshot, specifying `vbr.py` with the `--task restore`.

```
$ vbr.py --task restore --config-file exampleBackup.ini
Copying...
[==================================================] 100%
All child processes terminated successfully.
restore done!
```

You can restore a snapshot only to the database from which it was taken. You cannot restore a snapshot into an empty database.

Using Database Designer to Create a Comprehensive Design

HP Vertica's Database Designer:

1. Analyzes your logical schema, sample data, and, optionally, your sample queries.
2. Creates a physical schema design (a set of projections) that can be deployed automatically or manually.
3. Can be used by anyone without specialized database knowledge.
4. Can be run and re-run any time for additional optimization without stopping the database.
5. Uses sophisticated strategies to provide excellent ad-hoc query performance while using disk space efficiently.

Use Database Designer to create a comprehensive design, which allows you to create new projections for all tables in your database.

You can also use Database Designer to create an incremental design, which creates projections for all tables referenced in the queries you supply. For more information, see Creating an Incremental Design Using the Database Designer in the Administrator's Guide.

You can create a comprehensive design using the Database Designer through the Administration Tools interface or you can access Database Designer functionality programmatically (see About Running Database Designer Programmatically in the Programmer's Guide).

Running the Database Designer with Administration Tools

In this procedure, you create a comprehensive design using Database Designer through the Administration Tools interface. If, in the future, you have a query that you want to optimize, you can create an enhanced (incremental) design with additional projections to be tuned specifically for the query you provide. See Creating an Incremental Design Using the Database Designer in the Administrator's Guide for more information.

Follow these steps to create the comprehensive design using the Database Designer:

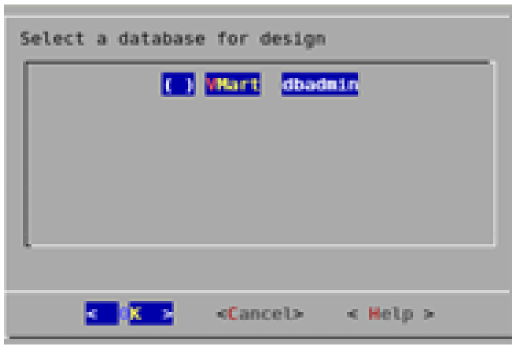
1. To exit the vsql session and return to the Main Menu in the Administration Tools, type `\q`.
2. Run Administration Tools:

```
$ /opt/vertica/bin/admintools
```

or simply type:

```
admintools
```

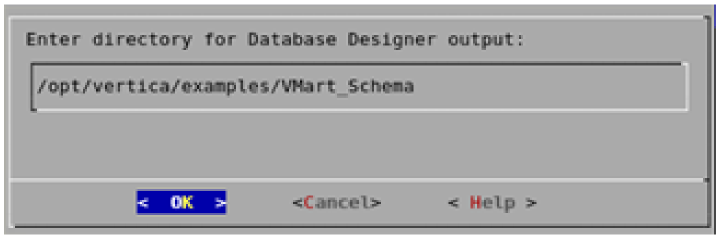
3. Start the database for which you want to create a design.
4. From the **Main Menu**, click **Configuration Menu** and click **OK**.
5. From the **Configuration Menu**, click **Run Database Designer** and click **OK**.
6. Select **VMart** in the **Select a database for design** dialog box and click **OK**.



```
Select a database for design

            ( ) VMart   dbadmin

      <  OK  >     <Cancel>     < Help >
```

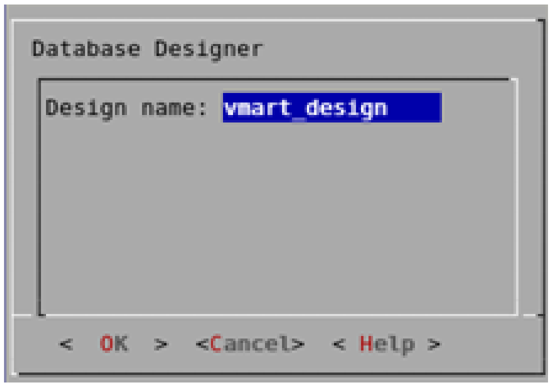
   If you are asked to enter the password for the database, click OK to bypass. No password was assigned when you installed the VMart database, so you do not need to enter one now.

7. Click **OK** to accept the default directory (`/opt/vertica/examples/VMart_Schema`, unless you changed it) for storing Database Designer output and log files.



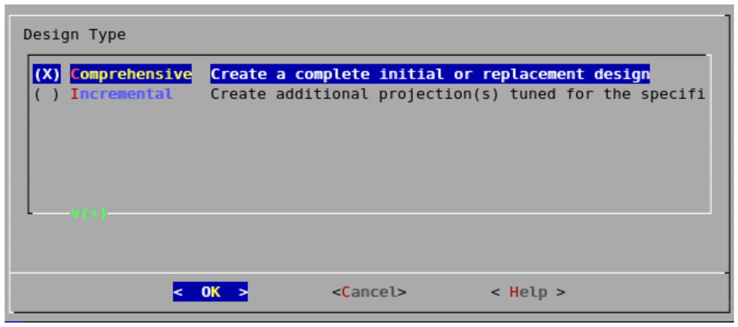
```
Enter directory for Database Designer output:
/opt/vertica/examples/VMart_Schema

      <  OK  >      <Cancel>      < Help >
```

8. In the **Database Designer** window, enter a name for the design, for example, `vmart_design`, and click **OK**. Design names can contain only alphanumeric characters or underscores. No other special characters are allowed.



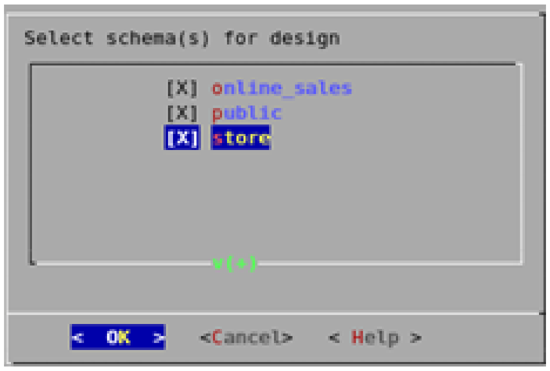
```
Database Designer

Design name: vmart_design

   <  OK  >  <Cancel>  < Help >
```

9. In the **Design Type** window, click **Comprehensive** to create a complete initial design and click **OK**.



```
Design Type

(X) Comprehensive   Create a complete initial or replacement design
( ) Incremental     Create additional projection(s) tuned for the specifi

         <  OK  >          <Cancel>          < Help >
```

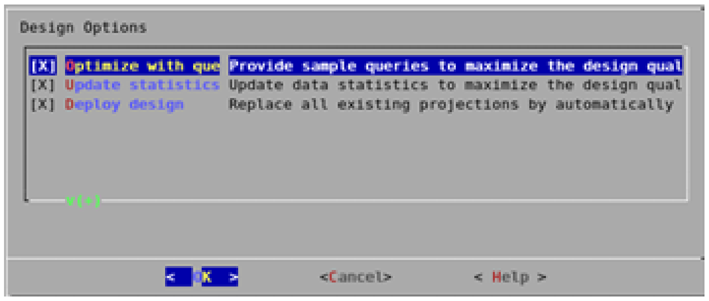
10. Because the VMart design is a multi-schema database, select all three schema (`online_sales`, `public`, `and store`,) for your design, click **OK**.



```
Select schema(s) for design

                  [X] online_sales
                  [X] public
                  [X] store

     <  OK  >  <Cancel>  < Help >
```

If you include a schema that contains tables without data, the Administration Tools notifies you that designing for tables without data could be suboptimal. You can choose to continue, but Hewlett-Packard recommends that you click **Cancel** and deselect the schema that contain empty tables before you proceed.

11. In the **Design Options** window, accept all three options and click **OK**.



```
Design Options

[X] Optimize with que Provide sample queries to maximize the design qual
[X] Update statistics Update data statistics to maximize the design qual
[X] Deploy design     Replace all existing projections by automatically
```

Generally, you enable all three options because Database Designer is best positioned to generate a new comprehensive design and create a complete set of projections for the tables in the selected schemas. The three options are:

- **Optimize with queries:** Supplying the Database Designer with queries is especially important if you want to optimize the database design for query performance.

  Database Designer does not impose hard limits to the number of queries or tables it accepts as inputs. However, it is limited by system resources, concurrent loads, and query/schema complexity. Hewlett-Packard recommends that you limit the design input to 100 queries.

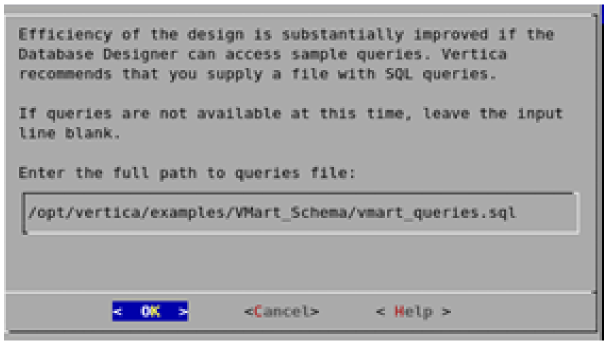
- **Update statistics:** Accurate statistics help the Database Designer choose the best strategy for data compression. If you select this option, the database statistics are updated to maximize design quality.

  Updating statistics takes time and resources, so if the concurrent statistics are up to date, this step is unnecessary. When in doubt, update statistics.

- **Deploy design:** The new design is automatically deployed, which means that during deployment, new projection are added, some existing projections might be retained, and any necessary existing projections are removed. Any new projections are refreshed so that they are populated with data.

12. Since you selected the **Optimize with queries** option, you must enter the full path to the file containing the queries that will be run on your database. In this example it is:

```
/opt/vertica/examples/VMart_Schema/vmart_queries.sql
```



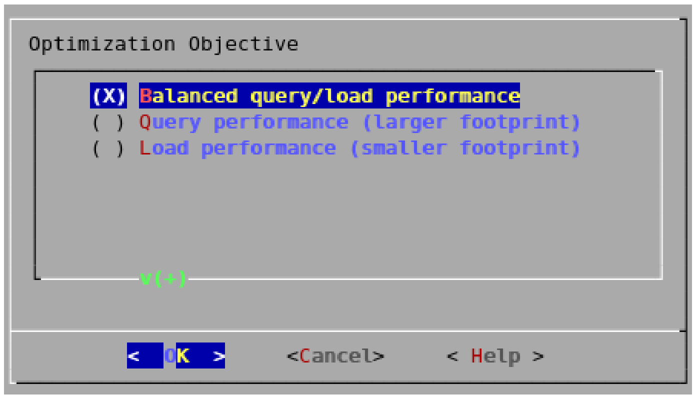
```
Efficiency of the design is substantially improved if the
Database Designer can access sample queries. Vertica
recommends that you supply a file with SQL queries.

If queries are not available at this time, leave the input
line blank.

Enter the full path to queries file:

/opt/vertica/examples/VMart_Schema/vmart_queries.sql
```

The queries in the query file must be delimited with a semicolon.

13. Choose the K-safety value you want. Click **OK**.

    If you are creating a comprehensive design on a single node, you are not asked to enter a K-safety value.

14. In the **Optimization Objective** window, select **Balanced query/load performance** and click **OK**.



```
Optimization Objective

    (X) Balanced query/load performance
    ( ) Query performance (larger footprint)
    ( ) Load performance (smaller footprint)
```

The optimization objectives are:

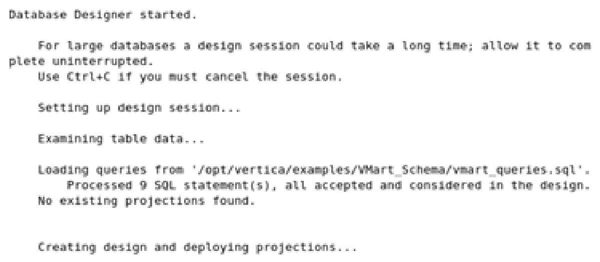
- **Balanced query/load performance** tells Database Designer to create a design that is balanced between database size and query performance.
- **Query performance (larger footprint)** creates a design focused on faster query performance, which might recommend additional projections. These projections could result in a larger database storage size.
- **Load performance (smaller footprint)** is optimized for loads, minimizing database size, potentially at the expense of query performance.

15. When the informational message displays, click **Proceed**.

    Database Designer:

    - Sets up the design session.
    - Examines table data.
    - Loads queries from the query file you provided (in this example, `/opt/vertica/examples/VMart_Schema/vmart_queries.sql`).
    - Creates the design.
    - Deploys the design or saves a SQL file containing the commands to create the design, depending on what you selected for the Deploy design option in step 10.

Depending on system resources, the design process could take several minutes. It is best to allow this process to complete uninterrupted. If the session must be canceled, use Ctrl+C.



```
Database Designer started.

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

    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.
    No existing projections found.


    Creating design and deploying projections...
```

16. When Database Designer finishes, press **Enter** to return to the Administration Tools menu.

    After Database Designer finishes, examine the steps taken to create the design. The files are stored in the directory you specified in step 6 (in this example, `/opt/vertica/examples/VMart_Schema`) and are named:

    - `<`*`design_name`*`>_design.sql`: Contains the CREATE PROJECTION statements.
    - `<`*`design_name`*`>_deploy.sql`: Contains the CREATE PROJECTION statements from the previous file, plus any additional SQL commands that do cleanup (mostly DROP PROJECTION commands to remove no-longer-necessary projections).
    - `<`*`design_name`*`>_parameters.txt`: A summary of the design's characteristics.

      When you run Database Designer using the Administration Tools, it creates a backup of the current design of your database before deploying the new design. This backup is stored in the directory you specified in step 6 (in this example, `/opt/vertica/examples/VMart_Schema`) and is named `catalog_dump.sql`

For additional information about managing your designs, see Designing a Physical Schema in the Administrator's Guide.

Restoring the Status of Your Host

When you finish the tutorial, you can restore your host machines to their original state. Use the following instructions to clean up your host and start over from scratch.

Stopping and Dropping the Database

Follow these steps to stop and/or drop your database. A database must be stopped before it can be dropped.

1. If connected to the database, disconnect by typing `\q`.
2. In the Administration Tools **Main Menu** dialog box, click **Stop Database** and click **OK**.
3. In the **Select database to stop** window, select the database you want to stop and click **OK**.
4. After stopping the database, click **Configuration Menu** and click **OK**.
5. Click **Drop Database** and click **OK**.
6. In the **Select database to drop** window, select the database you want to drop and click **OK**.
7. Click **Yes** to confirm.
8. In the next window type `yes` (lowercase) to confirm and click **OK**.

Alternatively, use the `delete_example` script, which stops and drops the database:

1. If connected to the database, disconnect by typing `\q`.
2. In the Administration Tools **Main Menu** dialog box, select **Exit**.
3. Log in as the database administrator.
4. Change to the `/examples` directory.

   ```
   $ cd /opt/vertica/examples
   ```

5. Run the delete_example script.

   ```
   $ /opt/vertica/sbin/delete_example Vmart
   ```

Uninstalling HP Vertica

Perform the steps in Uninstalling HP Vertica in the Installation Guide.

Optional Steps

You can also choose to:

- Remove the `dbadmin` account on all cluster hosts.
- Remove any example database directories you created.

Changing the GUI Appearance

The appearance of the Graphical User Interface (GUI) depends on the color and font settings used by your terminal window. The screen captures in this document were made using the default color and font settings in a PuTTY terminal application running on a Windows platform.

Note: If you are using a remote terminal application, such as PuTTY or a Cygwin bash shell, make sure your window is at least 81 characters wide and 23 characters high.

If you are using PuTTY, take these steps to make the Administration Tools look like the screen captures in this document.

1. In a PuTTY window, right-click the title area and select **Change Settings**.
2. Create or load a saved session.
3. In the **Category** dialog, click **Window > Appearance**.
4. In the **Font** settings, click the **Change**... button.
5. Select **Font**: Courier New, **Regular Size**: 10.
6. Click **Apply**.

Repeat these steps for each existing session that you use to run the Administration Tools.

You can also change the translation to support UTF-8.

1. In a PuTTY window, right-click the title area and select **Change Settings**.
2. Create or load a saved session.
3. In the **Category** dialog, click **Window > Translation**.
4. In the **Received data assumed to be in which character set** drop-down menu, select **UTF-8**.
5. Click **Apply**.

Appendix: VMart Example Database Schema, Tables, and Scripts

The Appendix provides detailed information about the VMart example database's schema, tables, and scripts.

The VMart example database contains three different schemas:

- `public`
- `store`
- `online_sales`

The term "schema" has several related meanings in HP Vertica:

- In SQL statements, a schema refers to named namespace for a logical schema.
- Logical schema refers to a set of tables and constraints.
- Physical schema refers to a set of projections.

Each schema contains tables that are created and loaded during database installation. See the schema maps for a list of tables and their contents:

- public Schema Map
- store Schema Map
- online_sales Schema Map

The VMart database installs with sample scripts that contain SQL commands that represent queries that might be used in a real business. The sample scripts are available in the Sample Scripts section of this Appendix. Once you're comfortable running the example queries, you might want to write your own.

Tables

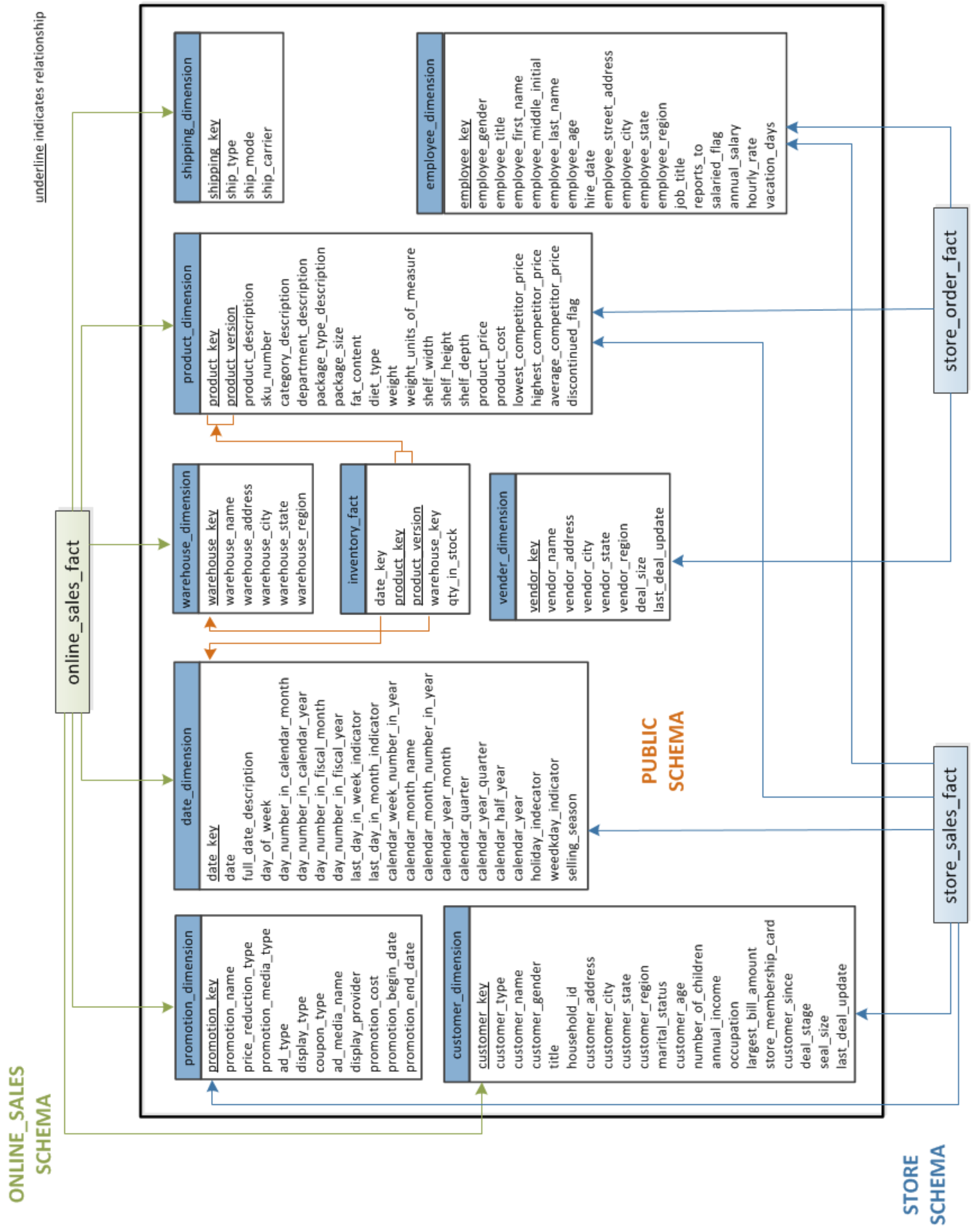
The three schemas in the VMart database include the following tables:

`public` **Schema**	`store` **Schema**	`online_sales` **Schema**
`inventory_fact`	`store_orders_fact`	`online_sales_fact`

customer_dimension	store_sales_fact	call_center_dimension
date_dimension	store_dimension	online_page_dimension
employee_dimension		
product_dimension		
promotion_dimension		
shipping_dimension		
vendor_dimension		
warehouse_dimension		

`public` Schema Map

The `public` schema is a snowflake schema. The following graphic illustrates the public schema and its relationships with tables in the `online_sales` and `store` schemas.



inventory_fact

This table contains information about each product in inventory.

Column Name	Data Type	NULLs

date_key	INTEGER	No
product_key	INTEGER	No
product_version	INTEGER	No
warehouse_key	INTEGER	No
qty_in_stock	INTEGER	No

customer_dimension

This table contains information about all the retail chain's customers.

Column Name	Data Type	NULLs
customer_key	INTEGER	No
customer_type	VARCHAR(16)	Yes
customer_name	VARCHAR(256)	Yes
customer_gender	VARCHAR(8)	Yes
title	VARCHAR(8)	Yes
household_id	INTEGER	Yes
customer_address	VARCHAR(256)	Yes
customer_city	VARCHAR(64)	Yes
customer_state	CHAR(2)	Yes
customer_region	VARCHAR(64)	Yes
marital_status	VARCHAR(32)	Yes
customer_age	INTEGER	Yes
number_of_children	INTEGER	Yes
annual_income	INTEGER	Yes
occupation	VARCHAR(64)	Yes
largest_bill_amount	INTEGER	Yes
store_membership_card	INTEGER	Yes
customer_since	DATE	Yes
deal_stage	VARCHAR(32)	Yes

deal_size	INTEGER	Yes
last_deal_update	DATE	Yes

date_dimension

This table contains information about dates. It is generated from a file containing correct date/time data.

Column Name	Data Type	NULLs
date_key	INTEGER	No
date	DATE	Yes
full_date_description	VARCHAR(18)	Yes
day_of_week	VARCHAR(9)	Yes
day_number_in_calendar_month	INTEGER	Yes
day_number_in_calendar_year	INTEGER	Yes
day_number_in_fiscal_month	INTEGER	Yes
day_number_in_fiscal_year	INTEGER	Yes
last_day_in_week_indicator	INTEGER	Yes
last_day_in_month_indicator	INTEGER	Yes
calendar_week_number_in_year	INTEGER	Yes
calendar_month_name	VARCHAR(9)	Yes
calendar_month_number_in_year	INTEGER	Yes
calendar_year_month	CHAR(7)	Yes
calendar_quarter	INTEGER	Yes
calendar_year_quarter	CHAR(7)	Yes
calendar_half_year	INTEGER	Yes
calendar_year	INTEGER	Yes
holiday_indicator	VARCHAR(10)	Yes
weekday_indicator	CHAR(7)	Yes
selling_season	VARCHAR(32)	Yes

employee_dimension

This table contains information about all the people who work for the retail chain.

Column Name	Data Type	NULLs
`employee_key`	INTEGER	No
`employee_gender`	VARCHAR(8)	Yes
`employee_title`	VARCHAR(8)	Yes
`employee_first_name`	VARCHAR(64)	Yes
`employee_middle_initial`	VARCHAR(8)	Yes
`employee_last_name`	VARCHAR(64)	Yes
`employee_age`	INTEGER	Yes
`hire_date`	DATE	Yes
`employee_street_address`	VARCHAR(256)	Yes
`employee_city`	VARCHAR(64)	Yes
`employee_state`	CHAR(2)	Yes
`employee_region`	CHAR(32)	Yes
`job_title`	VARCHAR(64)	Yes
`reports_to`	INTEGER	Yes
`salaried_flag`	INTEGER	Yes
`annual_salary`	INTEGER	Yes
`hourly_rate`	FLOAT	Yes
`vacation_days`	INTEGER	Yes

product_dimension

This table describes all products sold by the department store chain.

Column Name	Data Type	NULLs
`product_key`	INTEGER	No
`product_version`	INTEGER	No

`product_description`	VARCHAR(128)	Yes
`sku_number`	CHAR(32)	Yes
`category_description`	CHAR(32)	Yes
`department_description`	CHAR(32)	Yes
`package_type_description`	CHAR(32)	Yes
`package_size`	CHAR(32)	Yes
`fat_content`	INTEGER	Yes
`diet_type`	CHAR(32)	Yes
`weight`	INTEGER	Yes
`weight_units_of_measure`	CHAR(32)	Yes
`shelf_width`	INTEGER	Yes
`shelf_height`	INTEGER	Yes
`shelf_depth`	INTEGER	Yes
`product_price`	INTEGER	Yes
`product_cost`	INTEGER	Yes
`lowest_competitor_price`	INTEGER	Yes
`highest_competitor_price`	INTEGER	Yes
`average_competitor_price`	INTEGER	Yes
`discontinued_flag`	INTEGER	Yes

promotion_dimension

This table describes every promotion ever done by the retail chain.

Column Name	**Data Type**	**NULLs**
`promotion_key`	INTEGER	No
`promotion_name`	VARCHAR(128)	Yes
`price_reduction_type`	VARCHAR(32)	Yes
`promotion_media_type`	VARCHAR(32)	Yes
`ad_type`	VARCHAR(32)	Yes

display_type	VARCHAR(32)	Yes
coupon_type	VARCHAR(32)	Yes
ad_media_name	VARCHAR(32)	Yes
display_provider	VARCHAR(128)	Yes
promotion_cost	INTEGER	Yes
promotion_begin_date	DATE	Yes
promotion_end_date	DATE	Yes

shipping_dimension

This table contains information about shipping companies that the retail chain uses.

Column Name	Data Type	NULLs
shipping_key	INTEGER	No
ship_type	CHAR(30)	Yes
ship_mode	CHAR(10)	Yes
ship_carrier	CHAR(20)	Yes

vendor_dimension

This table contains information about each vendor that provides products sold through the retail chain.

Column Name	Data Type	NULLs
vendor_key	INTEGER	No
vendor_name	VARCHAR(64)	Yes
vendor_address	VARCHAR(64)	Yes
vendor_city	VARCHAR(64)	Yes
vendor_state	CHAR(2)	Yes
vendor_region	VARCHAR(32)	Yes
deal_size	INTEGER	Yes
last_deal_update	DATE	Yes

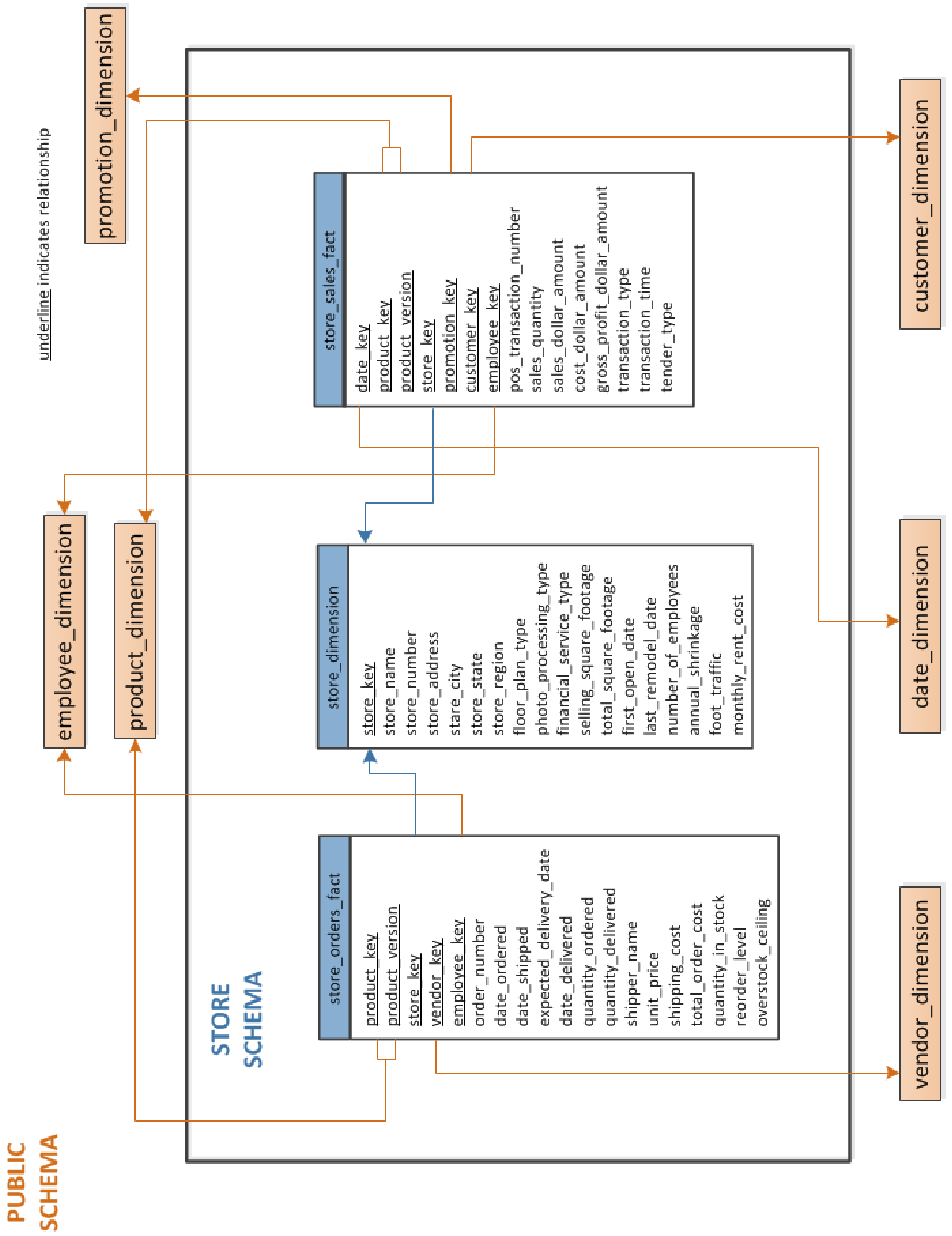
warehouse_dimension

This table provides information about each of the chain's warehouses.

Column Name	Data Type	NULLs
`warehouse_key`	INTEGER	No
`warehouse_name`	VARCHAR(20)	Yes
`warehouse_address`	VARCHAR(256)	Yes
`warehouse_city`	VARCHAR(60)	Yes
`warehouse_state`	CHAR(2)	Yes
`warehouse_region`	VARCHAR(32)	Yes

`store` Schema Map

The `store` schema is a snowflake schema that contains information about the retail chain's bricks-and-mortar stores. The following graphic illustrates the `store` schema and its relationship with tables in the `public` schema.



store_orders_fact

This table contains information about all orders made at the company’s brick-and-mortar stores.

Column Name	Data Type	NULLs
`product_key`	INTEGER	No
`product_version`	INTEGER	No
`store_key`	INTEGER	No
`vendor_key`	INTEGER	No
`employee_key`	INTEGER	No
`order_number`	INTEGER	No
`date_ordered`	DATE	Yes
`date_shipped`	DATE	Yes
`expected_delivery_date`	DATE	Yes
`date_delivered`	DATE	Yes
`quantity_ordered`	INTEGER	Yes
`quantity_delivered`	INTEGER	Yes
`shipper_name`	VARCHAR(32)	Yes
`unit_price`	INTEGER	Yes
`shipping_cost`	INTEGER	Yes
`total_order_cost`	INTEGER	Yes
`quantity_in_stock`	INTEGER	Yes
`reorder_level`	INTEGER	Yes
`overstock_ceiling`	INTEGER	Yes

store_sales_fact

This table contains information about all sales made at the company’s brick-and-mortar stores.

Column Name	Data Type	NULLs
`date_key`	INTEGER	No
`product_key`	INTEGER	No
`product_version`	INTEGER	No
`store_key`	INTEGER	No

promotion_key	INTEGER	No
customer_key	INTEGER	No
employee_key	INTEGER	No
pos_transaction_number	INTEGER	No
sales_quantity	INTEGER	Yes
sales_dollar_amount	INTEGER	Yes
cost_dollar_amount	INTEGER	Yes
gross_profit_dollar_amount	INTEGER	Yes
transaction_type	VARCHAR(16)	Yes
transaction_time	TIME	Yes
tender_type	VARCHAR(8)	Yes

store_dimension

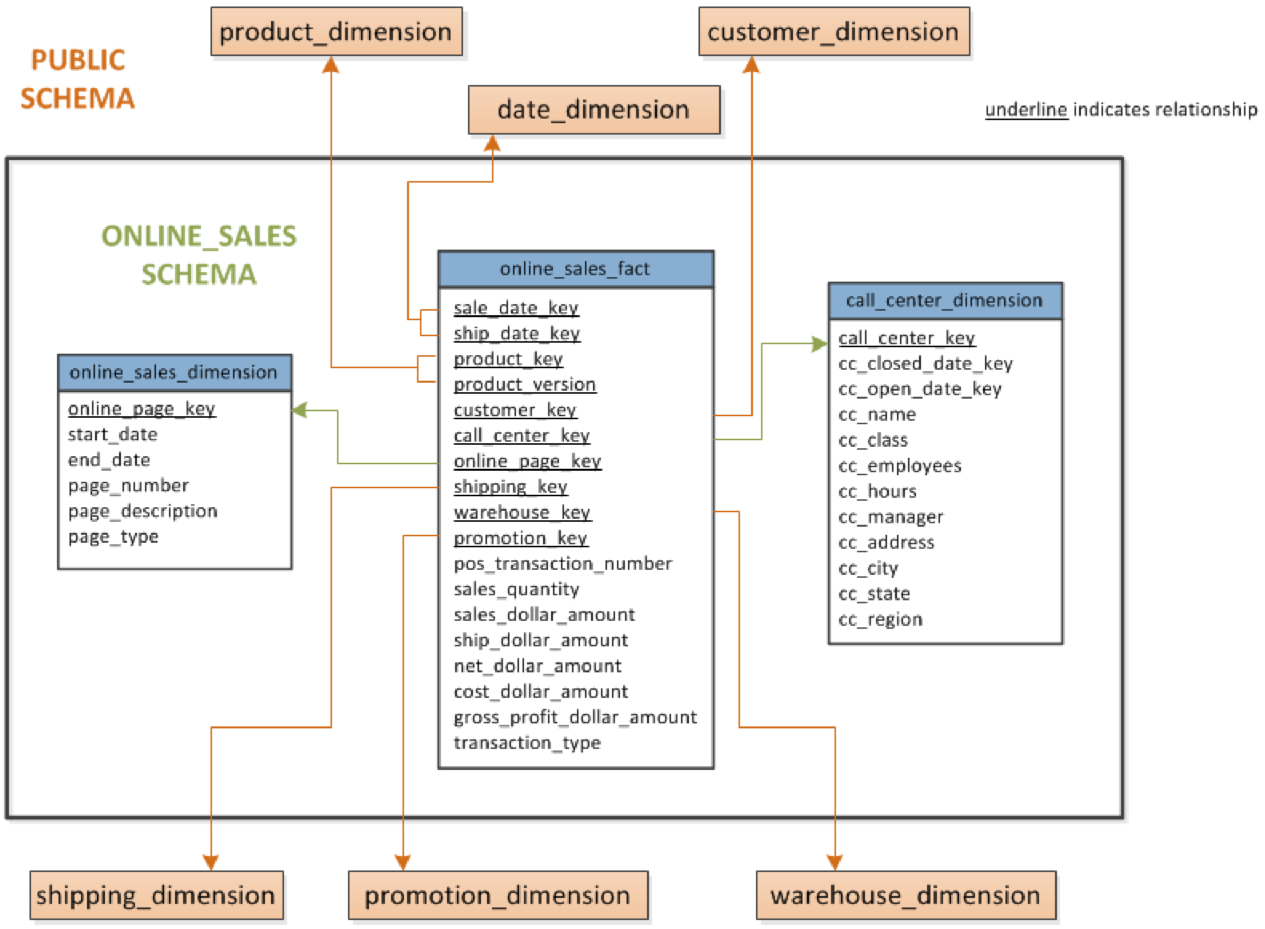
This table contains information about each brick-and-mortar store within the retail chain.

Column Name	Data Type	NULLs
store_key	INTEGER	No
store_name	VARCHAR(64)	Yes
store_number	INTEGER	Yes
store_address	VARCHAR(256)	Yes
store_city	VARCHAR(64)	Yes
store_state	CHAR(2)	Yes
store_region	VARCHAR(64)	Yes
floor_plan_type	VARCHAR(32)	Yes
photo_processing_type	VARCHAR(32)	Yes
financial_service_type	VARCHAR(32)	Yes
selling_square_footage	INTEGER	Yes
total_square_footage	INTEGER	Yes
first_open_date	DATE	Yes

`last_remodel_date`	DATE	Yes
`number_of_employees`	INTEGER	Yes
`annual_shrinkage`	INTEGER	Yes
`foot_traffic`	INTEGER	Yes
`monthly_rent_cost`	INTEGER	Yes

`online_sales` Schema Map

The `online_sales` schema is a snowflake schema that contains information about the retail chains. The following graphic illustrates the `online_sales` schema and its relationship with tables in the `public` schema.



online_sales_fact

This table describes all the items purchased through the online store front.

Column Name	Data Type	NULLs
`sale_date_key`	INTEGER	No
`ship_date_key`	INTEGER	No
`product_key`	INTEGER	No
`product_version`	INTEGER	No
`customer_key`	INTEGER	No
`call_center_key`	INTEGER	No
`online_page_key`	INTEGER	No
`shipping_key`	INTEGER	No
`warehouse_key`	INTEGER	No
`promotion_key`	INTEGER	No
`pos_transaction_number`	INTEGER	No
`sales_quantity`	INTEGER	Yes
`sales_dollar_amount`	FLOAT	Yes
`ship_dollar_amount`	FLOAT	Yes
`net_dollar_amount`	FLOAT	Yes
`cost_dollar_amount`	FLOAT	Yes
`gross_profit_dollar_amount`	FLOAT	Yes
`transaction_type`	VARCHAR(16)	Yes

call_center_dimension

This table describes all the chain's call centers.

Column Name	Data Type	NULLs
`call_center_key`	INTEGER	No
`cc_closed_date`	DATE	Yes
`cc_open_date`	DATE	Yes
`cc_date`	VARCHAR(50)	Yes
`cc_class`	VARCHAR(50)	Yes

cc_employees	INTEGER	Yes
cc_hours	CHAR(20)	Yes
cc_manager	VARCHAR(40)	Yes
cc_address	VARCHAR(256)	Yes
cc_city	VARCHAR(64)	Yes
cc_state	CHAR(2)	Yes
cc_region	VARCHAR(64)	Yes

online_page_dimension

This table describes all the pages in the online store front.

Column Name	Data Type	NULLs
online_page_key	INTEGER	No
start_date	DATE	Yes
end_date	DATE	Yes
page_number	INTEGER	Yes
page_description	VARCHAR(100)	Yes
page_type	VARCHAR(100)	Yes

Sample Scripts

You can create your own queries, but the VMart example directory includes sample query script files to help you get started quickly.

You can find the following sample scripts at this path `/opt/vertica/examples/VMart_Schema`.

To run any of the scripts, enter

```
=>  \i <script_name>
```

Alternatively, type the commands from the script file manually.

Note: The data that your queries return might differ from the example output shown in this guide because the sample data generator is random.

vmart_query_01.sql

```
-- vmart_query_01.sql
-- FROM clause subquery
-- Return the values for five products with the
-- lowest-fat content in the Dairy department

SELECT fat_content
FROM (
  SELECT DISTINCT fat_content
  FROM product_dimension
  WHERE department_description
  IN ('Dairy') ) AS food
  ORDER BY fat_content
  LIMIT 5;
```

Output

```
 fat_content
-------------
          80
          81
          82
          83
          84
(5 rows)
```

vmart_query_02.sql

```
-- vmart_query_02.sql
-- WHERE clause subquery
-- Asks for all orders placed by stores located in Massachusetts
-- and by vendors located elsewhere before March 1, 2003:

SELECT order_number, date_ordered
FROM store.store_orders_fact orders
WHERE orders.store_key IN (
  SELECT store_key
  FROM store.store_dimension
  WHERE store_state = 'MA')
    AND orders.vendor_key NOT IN (
  SELECT vendor_key
  FROM public.vendor_dimension
  WHERE vendor_state = 'MA')
    AND date_ordered < '2003-03-01';
```

Output

```
order_number | date_ordered
-------------+--------------
       53019 | 2003-02-10
      222168 | 2003-02-05
      160801 | 2003-01-08
      106922 | 2003-02-07
      246465 | 2003-02-10
      234218 | 2003-02-03
      263119 | 2003-01-04
       73015 | 2003-01-01
      233618 | 2003-02-10
       85784 | 2003-02-07
      146607 | 2003-02-07
      296193 | 2003-02-05
       55052 | 2003-01-05
      144574 | 2003-01-05
      117412 | 2003-02-08
      276288 | 2003-02-08
      185103 | 2003-01-03
      282274 | 2003-01-01
      245300 | 2003-02-06
      143526 | 2003-01-04
       59564 | 2003-02-06
...
```

vmart_query_03.sql

```
-- vmart_query_03.sql
-- Noncorrelated subquery
-- Requests female and male customers with the maximum
-- annual income from customers

SELECT customer_name, annual_income
FROM public.customer_dimension
WHERE (customer_gender, annual_income) IN (
  SELECT customer_gender, MAX(annual_income)
  FROM public.customer_dimension
  GROUP BY customer_gender);
```

Output

```
  customer_name   | annual_income
------------------+---------------
 James M. McNulty |        999979
 Emily G. Vogel   |        999998

(2 rows)
```

vmart_query_04.sql

```
-- vmart_query_04.sql
-- IN predicate
-- Find all products supplied by stores in MA
```

```
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;
```

Output

```
 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
3 | Brand #9 camera case
4 | Brand #12 rash ointment
4 | Brand #13 low fat milk
4 | Brand #14 chocolate chip cookies
4 | Brand #15 silver polishing cream
5 | Brand #16 cod
5 | Brand #17 band aids
6 | Brand #18 bananas
6 | Brand #19 starch
6 | Brand #20 vegetable soup
6 | Brand #21 bourbon
...
```

vmart_query_05.sql

```
-- vmart_query_05.sql
-- EXISTS predicate
-- Get a list of all the orders placed by all stores on
-- January 2, 2003 for the vendors with records in the
-- vendor_dimension table

SELECT store_key, order_number, date_ordered
FROM store.store_orders_fact
WHERE EXISTS (
   SELECT 1
   FROM public.vendor_dimension
   WHERE public.vendor_dimension.vendor_key = store.store_orders_fact.vendor_key)
   AND date_ordered = '2003-01-02';
```

Output

```
 store_key | order_number | date_ordered
-----------+--------------+--------------
        98 |       151837 | 2003-01-02
       123 |       238372 | 2003-01-02
       242 |       263973 | 2003-01-02
       150 |       226047 | 2003-01-02
       247 |       232273 | 2003-01-02
       203 |       171649 | 2003-01-02
       129 |        98723 | 2003-01-02
        80 |       265660 | 2003-01-02
       231 |       271085 | 2003-01-02
       149 |        12169 | 2003-01-02
       141 |       201153 | 2003-01-02
         1 |        23715 | 2003-01-02
       156 |        98182 | 2003-01-02
        44 |       229465 | 2003-01-02
       178 |       141869 | 2003-01-02
       134 |        44410 | 2003-01-02
       141 |       129839 | 2003-01-02
       205 |        54138 | 2003-01-02
       113 |        63358 | 2003-01-02
        99 |        50142 | 2003-01-02
        44 |       131255 | 2003-01-02

...
```

vmart_query_06.sql

```
-- vmart_query_06.sql
-- EXISTS predicate
-- Orders placed by the vendor who got the best deal
-- on January 4, 2004

SELECT store_key, order_number, date_ordered
FROM store.store_orders_fact ord, public.vendor_dimension vd
WHERE ord.vendor_key = vd.vendor_key
AND vd.deal_size IN (
   SELECT MAX(deal_size)
   FROM public.vendor_dimension)
AND date_ordered = '2004-01-04';
```

Output

```
 store_key | order_number | date_ordered
-----------+--------------+--------------
        45 |       202416 | 2004-01-04
        24 |       250295 | 2004-01-04
       121 |       251417 | 2004-01-04
       198 |        75716 | 2004-01-04
       166 |        36008 | 2004-01-04
        27 |       150241 | 2004-01-04
       148 |       182207 | 2004-01-04
         9 |       188567 | 2004-01-04
       113 |        66017 | 2004-01-04
```

```
...
```

vmart_query_07.sql

```
-- vmart_query_07.sql
-- Multicolumn subquery
-- Which products have the highest cost,
-- grouped by category and department

SELECT product_description, sku_number, department_description
FROM public.product_dimension
WHERE (category_description, department_description, product_cost) IN (
   SELECT category_description, department_description,
   MAX(product_cost) FROM product_dimension
   GROUP BY category_description, department_description);
```

Output

```
product_description        |      sku_number       |   department_description
---------------------------+-----------------------+---------------------------------
 Brand #601 steak          | SKU-#601              | Meat
 Brand #649 brooms         | SKU-#649              | Cleaning supplies
 Brand #677 veal           | SKU-#677              | Meat
 Brand #1371 memory card   | SKU-#1371             | Photography
 Brand #1761 catfish       | SKU-#1761             | Seafood
 Brand #1810 frozen pizza  | SKU-#1810             | Frozen Goods
 Brand #1979 canned peaches | SKU-#1979            | Canned Goods
 Brand #2097 apples        | SKU-#2097             | Produce
 Brand #2287 lens cap      | SKU-#2287             | Photography
...
```

vmart_query_08.sql

```
-- vmart_query_08.sql
-- Using pre-join projections to answer subqueries
-- between online_sales_fact and online_page_dimension

SELECT page_description, page_type, start_date, end_date
FROM online_sales.online_sales_fact f, online_sales.online_page_dimension d
WHERE f.online_page_key = d.online_page_key
AND page_number IN
   (SELECT MAX(page_number)
     FROM online_sales.online_page_dimension)
AND page_type = 'monthly' AND start_date = '2003-06-02';
```

Output

```
      page_description     | page_type | start_date | end_date
---------------------------+-----------+------------+-----------
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
Online Page Description #1 | monthly   | 2003-06-02 | 2003-06-11
(12 rows)
```

vmart_query_09.sql

```
-- vmart_query_09.sql
-- Equi join
-- Joins online_sales_fact table and the call_center_dimension
-- table with the ON clause

SELECT sales_quantity, sales_dollar_amount, transaction_type, cc_name
FROM online_sales.online_sales_fact
INNER JOIN online_sales.call_center_dimension
ON (online_sales.online_sales_fact.call_center_key
      = online_sales.call_center_dimension.call_center_key
     AND sale_date_key = 156)
ORDER BY sales_dollar_amount DESC;
```

Output

```
 sales_quantity | sales_dollar_amount | transaction_type |      cc_name
----------------+---------------------+------------------+-------------------
              7 |                 589 | purchase         | Central Midwest
              8 |                 589 | purchase         | South Midwest
              8 |                 589 | purchase         | California
              1 |                 587 | purchase         | New England
              1 |                 586 | purchase         | Other
              1 |                 584 | purchase         | New England
              4 |                 584 | purchase         | New England
              7 |                 581 | purchase         | Mid Atlantic
              5 |                 579 | purchase         | North Midwest
              8 |                 577 | purchase         | North Midwest
              4 |                 577 | purchase         | Central Midwest
              2 |                 575 | purchase         | Hawaii/Alaska
              4 |                 573 | purchase         | NY Metro
              4 |                 572 | purchase         | Central Midwest
              1 |                 570 | purchase         | Mid Atlantic
              9 |                 569 | purchase         | Southeastern
              1 |                 569 | purchase         | NY Metro
              5 |                 567 | purchase         | Other
              7 |                 567 | purchase         | Hawaii/Alaska
              9 |                 567 | purchase         | South Midwest
              1 |                 566 | purchase         | New England
...
```

We appreciate your feedback!

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Getting Started Guide (Vertica Analytic Database 7.0.x)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to vertica-docfeedback@hp.com.