Vertica® Analytic Database 4.1, Revision 1

Getting Started Guide

Date of Publication: January 7, 2011

CONFIDENTIAL

Contents

Technical Support

To submit problem reports, questions, comments, and suggestions, use the Technical Support page on the Vertica Systems, Inc., Web site.

> **Note:** You must be a registered user in order to access the support page.

1. Go to http://www.vertica.com/support (***http://www.vertica.com/support***).
2. Click **My Support**.

You can also email verticahelp@vertica.com.

Before you report a problem, run the Diagnostics Utility described in the Troubleshooting Guide and attach the resulting `.zip` file to your ticket.

About the Documentation

This section describes how to access and print Vertica documentation. It also includes ***suggested reading paths*** (page 4).

Where to Find the Vertica Documentation

You can read or download the Vertica documentation for the current release of Vertica® Analytic Database from the ***Product Documentation Page http://www.vertica.com/v-zone/product_documentation***. You must be a registered user to access this page.

The documentation is available as a compressed tarball (`.tar`) or a zip archive (`.zip`) file. When you extract the file on the database server system or locally on the client, contents are placed in a `/vertica41_doc/` directory.

> **Note:** The documentation on the Vertica Systems, Inc., Web site is updated each time a new release is issued. If you are using an older version of the software, refer to the documentation on your database server or client systems.

See Installing Vertica Documentation in the Installation Guide.

Reading the Online Documentation

Reading the HTML documentation files

The Vertica documentation files are provided in HTML browser format for platform independence. The HTML files require only a browser that displays frames properly with JavaScript enabled. The HTML files do not require a Web (HTTP) server.

The Vertica documentation is supported on the following browsers:

- Mozilla FireFox
- Internet Explorer
- Apple Safari
- Opera
- Google Chrome (server-side installations only)

The instructions that follow assume you have installed the documentation on a client or server machine.

Mozilla Firefox

1. Open a browser window.
2. Choose one of the following methods to access the documentation:
   - Select **File** > **Open File**, navigate to `..\HTML-WEBHELP\index.htm`, and click **Open**.
   - OR drag and drop `index.htm` into a browser window.

- OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Internet Explorer

Use one of the following methods:

1 Open a browser window.

2 Choose one of the following methods to access the documentation:

- Select **File** > **Open** > **Browse**, navigate to `..\HTML-WEBHELP\index.htm`, click **Open**, and click **OK**.
- OR drag and drop `index.htm` into the browser window.
- OR press **CTRL+O**, Browse to the file, click **Open**, and click **OK**.

Note: If a message warns you that Internet Explorer has restricted the web page from running scripts or ActiveX controls, right-click anywhere within the message and select **Allow Blocked Content**.

Apple Safari

1 Open a browser window.

2 Choose one of the following methods to access the documentation:

- Select **File** > **Open File**, navigate to `..\HTML-WEBHELP\index.htm`, and click Open.
- OR drag and drop `index.htm` into the browser window.
- OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Opera

1 Open a browser window.

2 Position your cursor in the title bar and right click > **Customize** > **Appearance**, click the **Toolbar** tab and select **Main Bar**.

3 Choose one of the following methods to access the documentation:

- Open a browser window and click Open, navigate to `..\HTML-WEBHELP\index.htm`, and click **Open**.
- OR drag and drop `index.htm` into the browser window.
- OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Google Chrome

Google does not support access to client-side installations of the documentation. You'll have to point to the documentation installed on a server system.

1 Open a browser window.

2 Choose one of the following methods to access the documentation:

In the address bar, type the location of the `index.htm` file on the server. For example: `file://<servername>//vertica41_doc//HTML/Master/index.htm`

- OR drag and drop `index.htm` into the browser window.
- OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Notes

The `.tar` or `.zip` file you download contains a complete documentation set.

The documentation page of the ***Downloads Web site http://www.vertica.com/v-zone/download_vertica*** is updated as new versions of Vertica are released. When the version you download is no longer the most recent release, refer only to the documentation included in your RPM.

The Vertica documentation contains links to Web sites of other companies or organizations that Vertica does not own or control. If you find broken links, please let us know.

Report any script, image rendering, or text formatting problems to ***Technical Support*** (on page 1).

Printing Full Books

Vertica also publishes books as Adobe Acrobat™ PDF. The books are designed to be printed on standard 8½ x 11 paper using full duplex (two-sided) printing.

> **Note:** Vertica manuals are topic driven and not meant to be read in a linear fashion. Therefore, the PDFs do not resemble the format of typical books. Each topic starts a new page, so some of the pages are very short, and there are blank pages between each topic.

Open and print the PDF documents using Acrobat Acrobat Reader. You can download the latest version of the free Reader from the ***Adobe Web site*** (***http://www.adobe.com/products/acrobat/readstep2.html***).

The following list provides links to the PDFs.

- Release Notes
- Concepts Guide
- Installation Guide
- Getting Started Guide
- Administrator's Guide
- Programmer's Guide
- SQL Reference Manual
- Troubleshooting Guide

Suggested Reading Paths

This section provides a suggested reading path for various users. Vertica recommends that you read the manuals listed under All Users first.

All Users

- Release Notes — Release-specific information, including new features and behavior changes to the product and documentation
- Concepts Guide — Basic concepts critical to understanding Vertica

- Getting Started Guide — A tutorial that takes you through the process of configuring a Vertica database and running example queries
- Troubleshooting Guide — General troubleshooting information

System Administrators

- Installation Guide — Platform configuration and software installation
- Release Notes — Release-specific information, including new features and behavior changes to the product and documentation

Database Administrators

- Installation Guide — Platform configuration and software installation
- Administrator's Guide — Database configuration, loading, security, and maintenance

Application Developers

- Programmer's Guide — Connecting to a database, queries, transactions, and so on
- SQL Reference Manual — SQL and Vertica-specific language information

Where to Find Additional Information

Visit the ***Vertica Systems, Inc. Web site*** (***http://www.vertica.com***) to keep up to date with:

- Downloads
- Frequently Asked Questions (FAQs)
- Discussion forums
- News, tips, and techniques
- Training

Typographical Conventions

The following are the typographical and syntax conventions used in the Vertica documentation.

Typographical Convention	Description
Bold	Indicates areas of emphasis, such as a special menu command.
Button	Indicates the word is a button on the window or screen.
`Code`	SQL and program code displays in a monospaced (fixed-width) font.
`Database objects`	Names of database objects, such as tables, are shown in san-serif type.
Emphasis	Indicates emphasis and the titles of other documents or system files.
`monospace`	Indicates literal interactive or programmatic input/output.
`monospace italics`	Indicates user-supplied information in interactive or programmatic input/output.
`UPPERCASE`	Indicates the name of a SQL command or keyword. SQL keywords are case insensitive; `SELECT` is the same as `Select`, which is the same as `select`.
`User input`	Text entered by the user is shown in bold san serif type.
↵	indicates the Return/Enter key; implicit on all user input that includes text
Right-angle bracket >	Indicates a flow of events, usually from a drop-down menu.
Click	Indicates that the reader clicks options, such as menu command buttons, radio buttons, and mouse selections; for example, "Click OK to proceed."
Press	Indicates that the reader perform some action on the keyboard; for example, "Press Enter."

Syntax Convention	Description
`Text without brackets/braces`	Indicates content you type as shown.
`<` *`Text inside angle brackets`* `>`	Placeholder for which you must supply a value. The variable is usually shown in italics. See Placeholders below.
`[ Text inside brackets ]`	Indicates optional items; for example, CREATE TABLE [*schema_name.*]*table_name* The brackets indicate that the *schema_name* is optional. Do not type the square brackets.
`{ Text inside braces }`	Indicates a set of options from which you choose one; for example: `QUOTES { ON | OFF }` indicates that exactly one of ON or OFF must

be provided.You do not type the braces: `QUOTES ON`

Backslash `\`

Continuation character used to indicate text that is too long to fit on a single line.

Ellipses `...`

Indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options.

Note: Showing an ellipses in code examples might also mean that part of the text has been omitted for readability, such as in multi-row result sets.

`    Indentation`

Is an attempt to maximize readability; SQL is a free-form language.

`Placeholders`

Items that must be replaced with appropriate identifiers or expressions are shown in italics.

Vertical bar `|`

Is a separator for mutually exclusive items. For example: `[ASC | DESC]`

Choose one or neither. You do not type the square brackets.

Overview to Getting Started

This guide serves as as a tutorial, walking you through the process of configuring a Vertica database and running example queries.

Before you start, Vertica recommends that you read the Concepts Guide to gain a quick understanding of unfamiliar concepts.

Prerequisites

All example databases described in this guide (including the one-step example) share the following prerequisites:

- You have installed Vertica on a cluster of hosts, as described in the Installation Guide.
- You have downloaded and installed the documentation, as described in ***Where to Find the Vertica Documentation*** (page 2).
- You are logged in to the server as the Database Administrator user; for example, `dbadmin`.
- You access to your database either by an SSH client or through the terminal utility in your Linux Console, such as vsql.

User Interfaces

By following this tutorial, you use the following user interfaces:

- The Linux command line (shell) interface
- The Vertica Administration Tools (see the Administrator's Guide for details)
- The vsql client interface (see the Programmer's Guide for details)

Example Databases

Vertica provides several simplified versions of databases that might actually be used in real-world applications. Detailed descriptions of each are provided in ***Example Databases*** (page 11).

You can use these databases as examples for learning purposes or as templates for actual databases. Even if your business has nothing to do with any of the sample schemas, the tutorial is useful because the techniques are the same, regardless of the type of data warehouse you use.

If you installed the product RPM, the example databases are located in `/opt/vertica/examples` on the host.

One-step Example Database

Vertica provides a one-step installation script that lets you create an example database and start using it immediately. The scripts are located in `/opt/vertica/sbin` and are called:

- `install_example` — Creates a database on the default port (5433), generates data, creates the schema and a default superprojection, and loads the data.
- `delete_example` — Drops the database

Note: For a more advanced but equally-simple example using the Vertica databases, see the ***Tutorial*** (page 87) in the Getting Started Guide.

Tutorial

The Tutorial describes how to configure a Vertica database that you'll use to run sample queries. It assumes that you have already installed Vertica on a cluster of hosts, as described in the Installation Guide. You can copy the example databases to non-cluster hosts for reference purposes, but you must perform the tasks in the Tutorial on the Administration Host.

Example Queries

Each example database includes several queries that are intended to represent queries that might be used in a real business. Once you're comfortable running the example queries, you'll probably want to write your own. Instructions are provided in ***Running Simple Queries*** (page 105).

Cleanup Procedure

When you have finished with the tutorial, you can restore your host machines to their original state. Instructions are provided in ***Cleanup Procedure*** (page 106).

Example Databases

Vertica provides several example databases that you can use in the tutorial:

- ***ClickStream Example Database*** (page 13)
- ***Credit History Example Database*** (page 22)
- ***Retail Sales Example Database*** (page 31)
- ***Stock Exchange Example Database*** (page 45)
- ***Telecom Example Database*** (page 58)
- ***VMart Example Database*** (page 67)

You can perform this tutorial using any or all of the example databases. The actual data returned from your queries will differ from the data published within this guide because the sample data generator program generates a new set of data for each of your example databases.

Caution: Although you can define multiple example databases within a single Vertica installation, Vertica Systems, Inc. strongly recommends that you start only one example database at a time to avoid unpredictable results.

Example Database File Locations

The example databases are installed in:

```
/opt/vertica/examples/ClickStream_Schema
/opt/vertica/examples/CreditHistory_Schema
/opt/vertica/examples/Retail_Schema
/opt/vertica/examples/Stock_Schema
/opt/vertica/examples/Telecom_Schema
/opt/vertica/examples/VMart_Schema
```

Example Database File Descriptions

Each example database has an identical set of files except for the file name prefix and the number of query files. In each of the names in the list below, replace *example* with the prefix string that corresponds to one of the example databases:

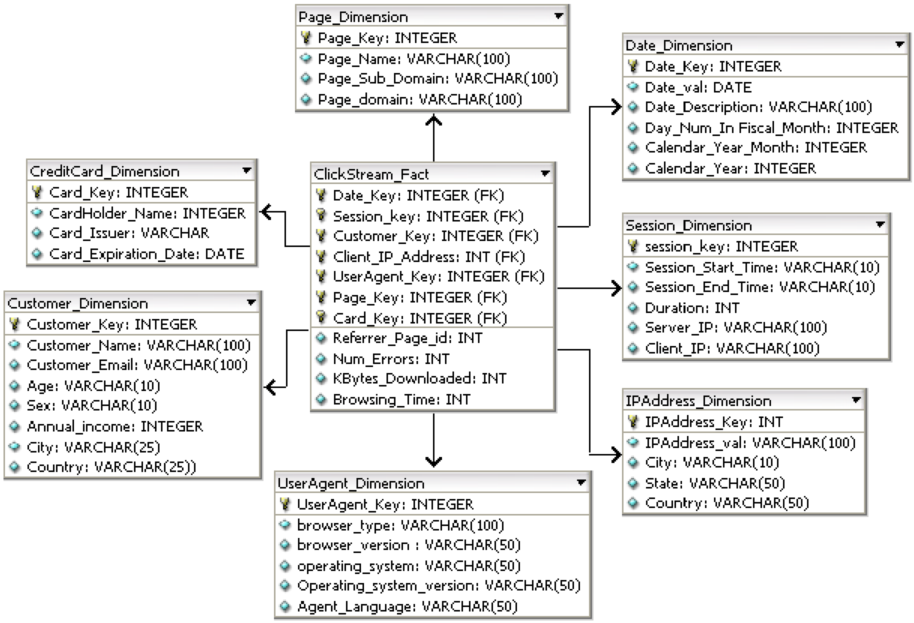
```
clickstream
credithistory
retail
stock
telecom
vmart
```

*example*_count_data.sql	SQL script that does a COUNT(*) of each table; can be used to verify load.
*example*_define_schema.sql	SQL script that defines the logical schema: tables and referential integrity constraints.
*example*_gen.cpp	Data generator source code (C++).
*example*_gen	Data generator executable file.

*example*_load_data.sql	SQL script that loads the generated sample data.
*example*_queries.sql	SQL script contain concatenated queries for use as a training set for the Database Designer.
*example*_query_01.sql ⋮	SQL scripts containing individual queries.
*example*_schema_drop.sql	SQL script that drops the schema.
README	Text file containing instructions for using the data generator.
Time.txt	Text file containing pre-computed data for date dimension tables.

ClickStream Example Database

The ClickStream Example Database is a simple star schema that represents a record of the clicks made by a user on a web site. This data can be analyzed and used, for example, for business/marketing purposes or the detection of malicious activities on the web site. Each table is described in a separate section.



The ClickStream schema is focused towards discovering interesting and useful information from Web content and usage. This schema can be used for

- Marketing promotions
- Click Fraud Detection
- Improving Web site design and performance etc.

The data in the ClickStream schema is populated from parsing Web Server logs, users browsing activities and habits etc. This data can be used for tracking malicious and fraudulent activities in real time. The schema is focused towards recognizing patterns either by using statistical models, by manual off-line analysis or by SQL queries.

The schema is intended to answer following queries for fraud detection or other purposes

1 Number of users accessing web server from a given server IP per day? This helps us analyze whether any particular server is clogging the network or is involved in malicious attack.

2 Which client IP is generating excessively large hits?

3 Which customer (Client_IP) address is downloading huge amount of Data?

4 Which customer is coming from more then one client IP?

5 Which customer is creating large number of sessions per day?

6 On which page do users stay for maximum duration?

Table Name	Default Number of Rows
ClickStream_Fact (on page 14)	5000000
Customer_Dimension (on page 15)	5000
Session_Dimension (on page 17)	50000
UserAgent_Dimension (on page 17)	500
IPAddress _Dimension (on page 16)	1000
Page_Dimension (on page 16)	5000
CreditCard_Dimension (on page 15)	5000

ClickStream_Fact

Each row in the fact table represents a summary of the user clicks done during browser session.

Field Name	Data Type	Description/Example
`Date_Key`	INTEGER	Date Key
`Session_Key`	INTEGER	Foreign Key, references Session_Dimension table
`Customer_Key`	INTEGER	Foreign Key, references Customer_Dimension Table
`ClientIP_Key`	INTEGER	Client IP Address, Foreign Key, references IPAddress_Dimension Table
`ServerIP_Key`	INTEGER	WebServer IP Address Foreign Key, references IPAddress_Dimension Table
`UserAgent_ID`	INTEGER	Foreign Key, references UserAgent_Dimension table
`Page_Id`	INTEGER	Foreign Key, references Page_Dimension table
`Referrer_Page_id`	INTEGER	Referring Page id
`CreditCard_ID`	INTEGER	Foreign Key, references CreditCard_Dimension Table
`Num_Errors`	INTEGER	Number of Errors encountered while browsing

KBytes_Downloaded	INTEGER	Amount of Data downloaded at client machine
Browsing_Time_Per_Page	INTEGER	Browsing time in minutes

Customer_Dimension

This table describes the user demographic information. Data in this table is populated from parsing strings from web logs of server.

Field Name	Data Type	Description/Example
Customer_Key	INTEGER	Primary key
Name	VARCHAR	Name of customer
E-mail_ID	VARCHAR	Unique mail id of customer
Sex	CHAR	Sex of the customer
Age	INTEGER	Age of customer
Annual_income	INTEGER	Annual income of the customer; for example, 50000 ($50000)
City	VARCHAR	Home city of customer
State	VARCHAR	Home state of customer
Country	VARCHAR	Home country of customer

CreditCard_Dimension

This table describes the all domain pages.

Field Name	Data Type	Description/Example
Card_Key	INTEGER	PrimaryKey
CardHolder_Name	VARCHAR	Varchar
Card_Type	VARCHAR	MasterCard/Visa/Amex
Card_Expiration_Date	DATE	Date

Date_Dimension

Contains data for dates.

Field Name	Data Type	Description/Example
Date_Key	INTEGER	Primary key
Date_Val	DATE	Date in ‘mm/dd/yyyy’ format
Date_Description	VARCHAR	Description of the date; for example, January 1, 2000
Day_Num_In_Fiscal_Month	INTEGER	The day number in the month (1-31); for example, 21 for 21st of any month.
Calendar_Year_Month	INTEGER	Calendar month of the date (1-12); for example, 9 for September
Calendar_Year	INTEGER	Calendar year of the date; for example, 2001

IPAddress _Dimension

This table describes the customer demographic information. Data in this table is populated from parsing strings from web logs of server.

Field Name	Data Type	Description/Example
IPAddress_Key	INTEGER	Primary key
IPAddress_Val	VARCHAR	IP Address value in dotted decimal; for example, 172.16.0.1
City	VARCHAR	City part of IP address
State	VARCHAR	State part of IP address
Country	VARCHAR	Country part of IP address

Page_Dimension

This table describes each page's domain relationships.

Field Name	Data Type	Description/Example
Page_Key	INTEGER	Primary key
Page_Name	VARCHAR	Page description and name
Page_Sub_Domain	VARCHAR	Page sub domain
Page_Domain	VARCHAR	Page domain

Session_Dimension

This table details user browsing session information.

Field Name	Data Type	Description/Example
Session_Key	INTEGER	Primary key
Session_Start_Time	VARCHAR	Session start time
Session_End_Time	VARCHAR	Session end time
Duration	INTEGER	Duration of the session in minutes
Server_IP	VARCHAR	IP address of server
Client_IP	VARCHAR	IP address of client

UserAgent_Dimension

This table describes user agent types for all machine types.

Field Name	Data Type	Description/Example
UserAgent_Key	INTEGER	Primary key
Browser_Type	VARCHAR	Mozilla
Browser_Version	VARCHAR	4.7
Operating_System	VARCHAR	WinNT/Linux
Operating_System_Version	VARCHAR	4.0/5.0 etc
Agent_Language	VARCHAR	English/French etc

clickstream_query_01.sql

Query

```
-- Customer hitting the web server the most
-- number of times in a day

SELECT   Date_Val,
         Customer_Name,
         COUNT(*) AS Hits
FROM     ClickStream_Fact A,
         Customer_Dimension B,
         Date_Dimension C
WHERE    A.Customer_Key = B.Customer_Key
         AND A.Date_Key = C.Date_Key
```

```
GROUP BY Date_Val,Customer_Name
ORDER BY Hits DESC;
```

Example

```
  Date_Val  | Customer_Name | Hits
------------+---------------+------
 2000-11-19 | Michael       |  321
 2000-03-03 | Michael       |  320
 2000-12-20 | Sophie        |  317
 2000-12-03 | Sophie        |  314
 2000-07-02 | Sophie        |  313
 2000-05-17 | Michael       |  311
```

clickstream_query_02.sql

Query

```
-- Client IP hitting the server the most
-- number of times in a day

SELECT   Date_Val,
         IPAddress_Val,
         City,
         COUNT(*) AS Hits
FROM     ClickStream_Fact A,
         IPAddress_Dimension B,
         Date_Dimension C
WHERE    A.ClientIP_Key = B.IPAddress_Key
         AND A.Date_Key = C.Date_Key
GROUP BY Date_Val,IPAddress_Val,City
ORDER BY Hits DESC;
```

Example

```
  Date_Val  | IPaddress_Val |    City    | Hits
------------+---------------+------------+------
 2000-08-06 | 172.16.2.15   | Noida      |   11
 2000-10-19 | 172.16.1.3    | Tokyo      |   10
 2000-06-05 | 172.16.2.4    | Paris      |   10
 2000-07-05 | 172.16.1.6    | London     |   10
 2000-07-29 | 172.16.1.6    | London     |   10
 2000-01-19 | 172.16.2.15   | Noida      |   10
 2000-02-10 | 172.16.0.4    | Detroit    |   10
```

clickstream_query_03.sql

Query

```
-- Page with the maximum number of hits
-- and total browsing time
```

```
SELECT   Date_Val,
         Page_Name,
         SUM(Browsing_Time) AS Browsing_Time,
         COUNT(*) AS Hits
FROM     ClickStream_Fact A,
         Page_Dimension B,
         Date_Dimension C
WHERE    A.Date_Key = C.Date_Key
         AND A.Page_Key = B.Page_Key
GROUP BY Date_Val,Page_Name
ORDER BY Browsing_Time DESC,
         Hits DESC;
```

Example

```
  Date_Val   |                    Page_Name                     | Browsing_Time | Hits
-------------+--------------------------------------------------+---------------+-----
 2000-06-06 | http://www.Geocities.Yahoo.com/page72.html       |            90 |   16
 2000-11-19 | http://www.Jewellery.Rediff.com/page23.html      |            87 |   11
 2000-03-16 | http://www.MP3-Players.Rediff.com/page34.html    |            81 |   14
 2000-05-04 | http://www.Cricket.Rediff.com/page90.html        |            80 |   13
 2000-04-27 | http://www.Laptops.Rediff.com/page69.html        |            79 |   11
 2000-01-20 | http://www.Mobiles.Rediff.com/page97.html        |            75 |   12
```

clickstream_query_04.sql

Query

```
-- Customers creating more than 5 sessions per day

SELECT   Date_Val,
         Customer_Name,
         SUM(Duration),
         COUNT(*) AS Count_Session
FROM     ClickStream_Fact A,
         Date_Dimension B,
         Session_Dimension C,
         Customer_Dimension D
WHERE    A.Date_Key = B.Date_Key
         AND A.Customer_Key = D.Customer_Key
         AND A.Session_Key = C.Session_Key
GROUP BY Date_Val,Customer_Name,Duration
HAVING   COUNT(*) > 5
ORDER BY Duration DESC;
```

Example

```
  Date_Val   | Customer_Name | SUM   | Count_Session
-------------+---------------+-------+---------------
 2000-06-29 | Matthew       | 1320 |            11
 2000-07-08 | Hannah        | 1200 |            10
 2000-07-11 | Hannah        |  960 |             8
 2000-07-12 | Hannah        |  840 |             7
 2000-07-13 | Hannah        | 1800 |            15
 2000-07-15 | Hannah        | 1920 |            16
```

clickstream_query_05.sql

Query

```
-- Customers coming from more than one IP address

SELECT   Date_Val,
         Customer_Name,
         COUNT(ClientIP_Key) AS Client_IPS
FROM     ClickStream_Fact A,
         Date_Dimension B,
         Customer_Dimension C
WHERE    A.Date_Key = B.Date_Key
         AND A.Customer_Key = C.Customer_Key
         AND A.Date_Key > 100
         AND A.Date_Key < 105
GROUP BY Date_Val,Customer_Name
HAVING   COUNT(ClientIP_Key) > 10
ORDER BY Client_IPs DESC;
```

Example

```
  Date_Val  | Customer_Name | Client_IPS
------------+---------------+------------
 2000-04-11 | Sophie        |        308
 2000-04-11 | Michael       |        307
 2000-04-11 | Samuel        |        224
 2000-04-11 | Hannah        |        222
 2000-04-11 | Emily         |        214
 2000-04-13 | Sophie        |        213
```

clickstream_query_06.sql

Query

```
-- Pages visited by the maximum number of
-- IP addresses for a given month

SELECT   Page_Name,
         count(DISTINCT ipaddress_val) AS IP_Address_Count
FROM     ClickStream_Fact A,
         Page_Dimension B,
         IPAddress_Dimension C,
         Date_Dimension D
WHERE    A.Page_Key = B.Page_Key
         AND A.ClientIP_Key = C.IPAddress_Key
         AND A.Date_Key = D.Date_Key
         AND D.Calendar_Year = 2004
         AND D.Calendar_Month_Number_In_Year = 2
GROUP BY Page_Name
ORDER BY IP_Address_Count DESC;
```

Example

```
                       page_name                        | ip_address_count
--------------------------------------------------------+------------------
 http://www.Geocities.Yahoo.com/page2.html              |               46
 http://www.Geocities.Yahoo.com/page67.html             |               45
 http://www.Auctions.Rediff.com/page32.html             |               44
 http://www.Books.Amazon.com/page90.html                |               44
 http://www.Games.Yahoo.com/page33.html                 |               43
 http://www.Messenger.Rediff.com/page17.html            |               43
 http://www.Yellow-Pages.Yahoo.com/page60.html          |               42
 http://www.Groups.Yahoo.com/page73.html                |               41
 http://www.Electronics.Amazon.com/page16.html          |               41
 http://www.Real-Estate.Yahoo.com/page50.html           |               41
 http://www.Games.Yahoo.com/page61.html                 |               41
 http://www.Jewellery&Watches.Amazon.com/page6.html     |               41
 http://www.Tools&Automotive.Amazon.com/page79.html     |               40
 http://www.Home&Garden.Amazon.com/page8.html           |               40
 http://www.Aparel&Accessories.Amazon.com/page63.html   |               40
 http://www.Maps.Yahoo.com/page66.html                  |               39
 http://www.Jobs.Rediff.com/page6.html                  |               39
```

…

Credit History Example Database

The Credit History database is a simple star schema that represents customer credit history.

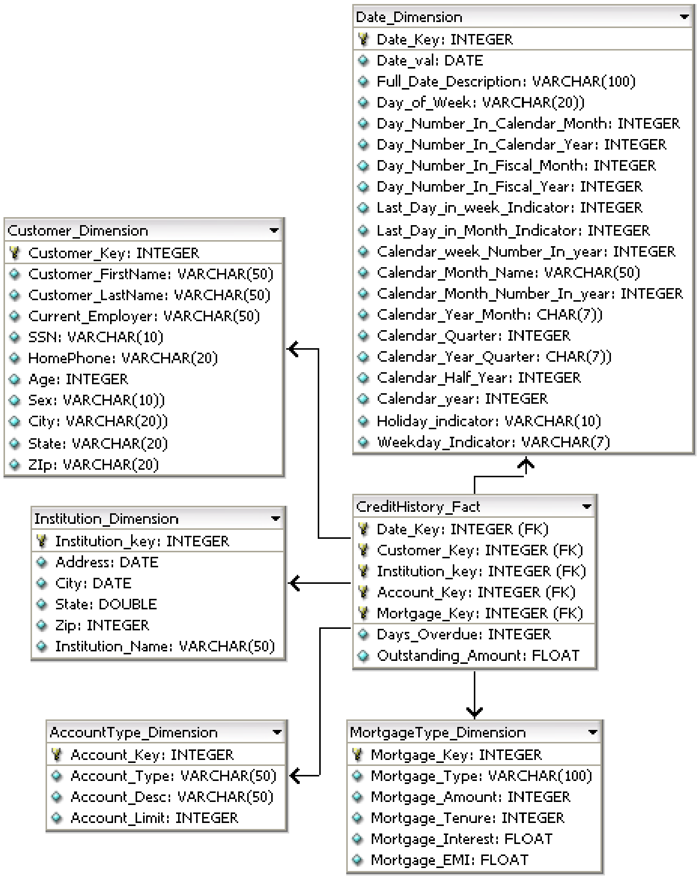

Date_Dimension
Date_Key: INTEGER
Date_val: DATE
Full_Date_Description: VARCHAR(100)
Day_of_Week: VARCHAR(20))
Day_Number_In_Calendar_Month: INTEGER
Day_Number_In_Calendar_Year: INTEGER
Day_Number_In_Fiscal_Month: INTEGER
Day_Number_In_Fiscal_Year: INTEGER
Last_Day_in_week_Indicator: INTEGER
Last_Day_in_Month_Indicator: INTEGER
Calendar_week_Number_In_year: INTEGER
Calendar_Month_Name: VARCHAR(50)
Calendar_Month_Number_In_year: INTEGER
Calendar_Year_Month: CHAR(7))
Calendar_Quarter: INTEGER
Calendar_Year_Quarter: CHAR(7))
Calendar_Half_Year: INTEGER
Calendar_year: INTEGER
Holiday_indicator: VARCHAR(10)
Weekday_Indicator: VARCHAR(7)
Customer_Dimension
Customer_Key: INTEGER
Customer_FirstName: VARCHAR(50)
Customer_LastName: VARCHAR(50)
Current_Employer: VARCHAR(50)
SSN: VARCHAR(10)
HomePhone: VARCHAR(20)
Age: INTEGER
Sex: VARCHAR(10))
City: VARCHAR(20))
State: VARCHAR(20)
ZIp: VARCHAR(20)
CreditHistory_Fact
Date_Key: INTEGER (FK)
Customer_Key: INTEGER (FK)
Institution_key: INTEGER (FK)
Account_Key: INTEGER (FK)
Mortgage_Key: INTEGER (FK)
Days_Overdue: INTEGER
Outstanding_Amount: FLOAT
Institution_Dimension
Institution_key: INTEGER
Address: DATE
City: DATE
State: DOUBLE
Zip: INTEGER
Institution_Name: VARCHAR(50)
AccountType_Dimension
Account_Key: INTEGER
Account_Type: VARCHAR(50)
Account_Desc: VARCHAR(50)
Account_Limit: INTEGER
MortgageType_Dimension
Mortgage_Key: INTEGER
Mortgage_Type: VARCHAR(100)
Mortgage_Amount: INTEGER
Mortgage_Tenure: INTEGER
Mortgage_Interest: FLOAT
Mortgage_EMI: FLOAT

Table Name	Default Number of Rows
CreditHistory_Fact (on page 23)	5000000
Customer_Dimension (on page 23)	5000
Institution_Dimension (on page 25)	100
AccountType_Dimension (on page 23)	50
MortgageType_Dimension (on page 25)	1000

CreditHistory_Fact

Each row in the fact table represents a credit transaction performed by an individual.

Field Name	Data Type	Description
`Date_Key`	INTEGER	Foreign Key reference Date table
`Customer_Key`	INTEGER	Foreign Key reference Customer table
`Institution_Key`	INTEGER	Foreign Key reference Institution table
`Account_Key`	INTEGER	Foreign Key reference AccountType Table
`Mortgage_Key`	INTEGER	Foreign Key reference MortgageType Table
`Days_Overdue`	INTEGER	This field represents the number of days credit is overdue. 99999 represents bad debt.
`Outstanding_Amount`	FLOAT	Outstanding amount for a credit transaction

AccountType_Dimension

This table describes the type of accounts that can be offered by financial institutions

Field Name	Data Type	Description/Example
`Account_Key`	INTEGER	Primary key
`Account_Type`	VARCHAR	Type of account checking/current/loan
`Account_Desc`	VARCHAR	Brief description of account type
`Account_Limit`	INTEGER	If loan account then sanctioned credit limit

Customer_Dimension

This table describes details of customers whose credit history is maintained by the company.

Field Name	Data Type	Description/Example
`Customer_Key`	INTEGER	Primary key
`Customer_FirstName`	VARCHAR	Customer first name
`Customer_LastName`	VARCHAR	Customer last name
`Current_Employer`	VARCHAR	Current employer
`SSN`	VARCHAR	Social security number
`HomePhone`	VARCHAR	Home phone

Age	VARCHAR	Customer age
Sex	VARCHAR	Customer sex
City	VARCHAR	Customer city
State	INTEGER	Customer state
Zip	VARCHAR	Zip code

Date_Dimension

Contains data for dates.

Field Name	Data Type	Description/Example
Date_Key	INTEGER	Primary Key
Date_val	DATE	Date In ‘mm/dd/yyyy’ format
Full_date_description	VARCHAR(18)	Description of the date; for example, January 1, 2000
Day_of_week	VARCHAR(9)	Calendar year of the date; for example, 2001
Day_number_in_calendar_month	INTEGER	Calendar month of the date (1-12); for example, for September
Day_number_in_calendar_year	INTEGER	The day number in the month (1-31); for example, 21 for 21st of any month.
Day_number_in_fiscal_month	INTEGER	
Day_number_in_fiscal_year	INTEGER	
Last_day_in_week_indicator	INTEGER	
Last_day_in_month_indicator	INTEGER	
Calendar_week_number_in_year	INTEGER	
Calendar_month_name	VARCHAR(9)	
Calendar_month_number_in_year	INTEGER	
Calendar_year_month	CHAR(7)	
Calendar_quarter	INTEGER	
Calendar_year_quarter	CHAR(7)	
Calendar_half_year	INTEGER	
Calendar_year	INTEGER	
Holiday_indicator	VARCHAR(10)	
Weekday_indicator	CHAR(7)	

Institution_Dimension

This table describes all the banking and financial institutions in the country.

Field Name	Data Type	Description/Example
`Institution_Key`	INTEGER	Primary key
`Institution_Name`	VARCHAR	Bank/credit lending institutions
`Address`	VARCHAR	Address of institution
`City`	VARCHAR	City of institution
`State`	VARCHAR	State of institution
`Zip`	VARCHAR	Zip code

MortgageType_Dimension

This table describes types of mortgages.

Field Name	Data Type	Description/Example
`Mortgage_Key`	INTEGER	Primary key
`Mortgage_Type`	VARCHAR	Car/home/personal mortgage
`Mortgage_Amount`	INTEGER	Mortgage Amount, such as $1000, $10000, and so on
`Mortgage_Tenure`	INTEGER	Mortgage tenure in months, such as 12, 24, 36, and so on
`Mortgage_Interest`	DOUBLE	Applicable interest rate.
`Mortgage_EMI`	DOUBLE	Amount payable monthly as installments

credithistory_query_01.sql

Query

```
-- Overdue statistics for 2001 by state
-- a. Avg Overdue (Amount and Days)
-- b. Max Overdue (Amount and Days)
-- c. Min Overdue (Amount and Days)

SELECT   State,
         MAX(Days_Overdue) AS Max_Days,
         MIN(Days_Overdue) AS Min_Days,
         AVG(Days_Overdue) AS Avg_Days,
         MAX(Outstanding_Amount) AS Max_Amount,
         MIN(Outstanding_Amount) AS Min_Amount,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Overdue_Recs
FROM     CreditHistory_Fact A,
         Customer_Dimension B,
         Date_Dimension C
WHERE    A.Date_Key = C.Date_Key
         AND A.Customer_Key = B.Customer_Key
         AND C.Calendar_Year = 2001
GROUP BY State
ORDER BY Avg_Amount DESC,
         Avg_Days DESC;
```

Example

```
State | Max_Days | Min_Days |     Avg_Days     | Max_Amount | Min_Amount |    Avg_Amount    |
Overdue_Recs
-------+----------+----------+------------------+------------+------------+------------------+---
-----------
 IL    |      999 |        0 | 498.137946406459 |    15000.3 |     500.67 | 7785.36343702016 |
20189
 NY    |      999 |        0 | 500.163568584688 |   15000.11 |     500.02 | 7750.80704536809 |
39433
 CA    |      999 |        0 | 499.313933330031 |   15000.51 |     500.73 | 7733.53519366982 |
40378
```

credithistory_query_02.sql

Query

```
-- Overdue statistics for 2001 by Institution
-- a. Avg Overdue (Amount and Days)
-- b. Max Overdue (Amount and Days)
-- c. Min Overdue (Amount and Days)

SELECT   Institution_Name,
         MAX(Days_Overdue) AS Max_Days,
         MIN(Days_Overdue) AS Min_Days,
         AVG(Days_Overdue) AS Avg_Days,
         MAX(Outstanding_Amount) AS Max_Amount,
         MIN(Outstanding_Amount) AS Min_Amount,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Overdue_Recs
FROM     CreditHistory_Fact A,
         Institution_Dimension B,
         Date_Dimension C
WHERE    A.Date_Key = C.Date_Key
         AND A.Institution_Key = B.Institution_Key
         AND C.Calendar_Year = 2000
GROUP BY Institution_Name
ORDER BY Avg_Amount DESC;
```

Example

```
Institution_Name  | Max_Days | Min_Days |     Avg_Days     | Max_Amount | Min_Amount |    Avg_Amount
```

```
| Overdue_Recs
------------------+----------+----------+------------------+------------+------------+-----------
-------+--------------
 INSTT#98         |      997 |        0 | 506.386450381679 |   14986.93 |     511.55 |
8034.51529580153 |         1048
 INSTT#57         |      999 |        2 |  494.70480081716 |   15000.01 |     508.57 |
8023.94215526047 |          979
 INSTT#83         |      999 |        0 | 508.528806584362 |   14994.48 |     502.39 |
8019.49127572016 |          972
 INSTT#56         |      999 |        3 |  516.19877675841 |   14979.93 |     511.46 |
7998.86175331295 |          981
 INSTT#45         |      997 |        1 | 498.116596638655 |   14994.69 |     507.47 |
7985.12201680672 |          952
 INSTT#66         |      998 |        0 | 488.579420579421 |   14990.66 |      501.3 |
7973.51433566434 |         1001
 INSTT#84         |      998 |        0 | 505.276302851524 |   14985.32 |     504.76 |
7964.23406096362 |         1017
 INSTT#90         |      996 |        1 |  510.30303030303 |   14990.34 |     536.82 |
7951.99204301075 |         1023
 INSTT#44         |      998 |        0 | 484.883883883884 |   14970.27 |     525.28 |
7945.75424424424 |          999
 INSTT#69         |      999 |        2 | 507.625502008032 |    14986.1 |     509.98 |
7936.75596385542 |          996
 INSTT#93         |      998 |        0 | 502.520669291339 |   15000.51 |     502.02 |
7936.17729330709 |         1016
 INSTT#73         |      998 |        0 | 491.066198595787 |   14993.14 |     559.45 |
7924.45994984955 |          997
```

credithistory_query_03.sql

Query

```
-- Overdue mortgage statistics by year with mortgage type

SELECT   Mortgage_Type,
         AVG(Days_Overdue) AS Avg_Days,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Overdue_Recs
FROM     CreditHistory_Fact A,
         Mortgage_Dimension B,
         Date_Dimension C
WHERE    A.Mortgage_Key = B.Mortgage_Key
         AND A.Date_Key = C.Date_Key
GROUP BY Calendar_Year,Mortgage_Type
ORDER BY Calendar_Year,
         Mortgage_Type;
```

Example

```
 Mortgage_Type |     Avg_Days     |    Avg_Amount    | Overdue_Recs
---------------+------------------+------------------+--------------
 Car           | 499.179450730653 | 7758.45090843616 |       105522
 Home          | 499.670780499164 | 7742.27507610237 |        94478
(2 rows)
```

credithistory_query_04.sql

Query

```
-- Overdue mortgage statistics by year with tenure
```

```
SELECT   Mortgage_Type,
         Mortgage_Tenure,
         AVG(Days_Overdue) AS Avg_Days,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Record_Count
FROM     CreditHistory_Fact A,
         Mortgage_Dimension B,
         Date_Dimension C
WHERE    A.Mortgage_Key = B.Mortgage_Key
         AND A.Date_Key = C.Date_Key
GROUP BY Calendar_Year,Mortgage_Type,Mortgage_Tenure
ORDER BY Calendar_Year,
         Mortgage_Type,
         Mortgage_Tenure;
```

Example

```
Mortgage_Type | Mortgage_Tenure |     Avg_Days     |    Avg_Amount    | Record_Count
--------------+-----------------+------------------+------------------+-------------
 Car          |              12 | 498.664561695056 | 7745.60994349813 |        24070
 Car          |              24 | 502.332021237642 | 7753.32524533138 |        21848
 Car          |              36 | 500.580798992262 | 7793.29573420911 |        22228
 Car          |              48 | 498.262124831239 |  7730.3517000727 |        19258
 Car          |              60 | 495.317695109836 | 7768.80918644442 |        18118
 Home         |              60 | 500.719860896445 | 7858.66575637558 |        20704
 Home         |              96 | 500.386262760763 | 7710.50094429649 |        18024
 Home         |             120 |  496.92023054755 | 7751.82940172911 |        17350
 Home         |             180 | 498.150733659404 | 7721.11076144953 |        17992
 Home         |             240 | 501.653077224618 | 7662.79473049784 |        20408
(10 rows)
```

credithistory_query_05.sql

Query

```
-- Overdue mortgage statistics by year with account type

SELECT   Account_Type,
         AVG(Days_Overdue) AS Avg_Days,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Record_Count
FROM     CreditHistory_Fact A,
         AccountType_Dimension B,
         Date_Dimension C
WHERE    A.AccountType_Key = B.AccountType_Key
         AND A.Date_Key = C.Date_Key
GROUP BY Calendar_Year,Account_Type
ORDER BY Calendar_Year,
         Account_Type;
```

Example

```
 Account_Type |     Avg_Days     |    Avg_Amount    | Record_Count
--------------+------------------+------------------+--------------
 Checking     | 500.261721483555 | 7741.41345971209 |        40012
 Current      | 501.090460467923 | 7785.66681471225 |        28167
 Saving       | 496.856415574621 | 7731.76984318532 |        31821
```

credithistory_query_06.sql

Query

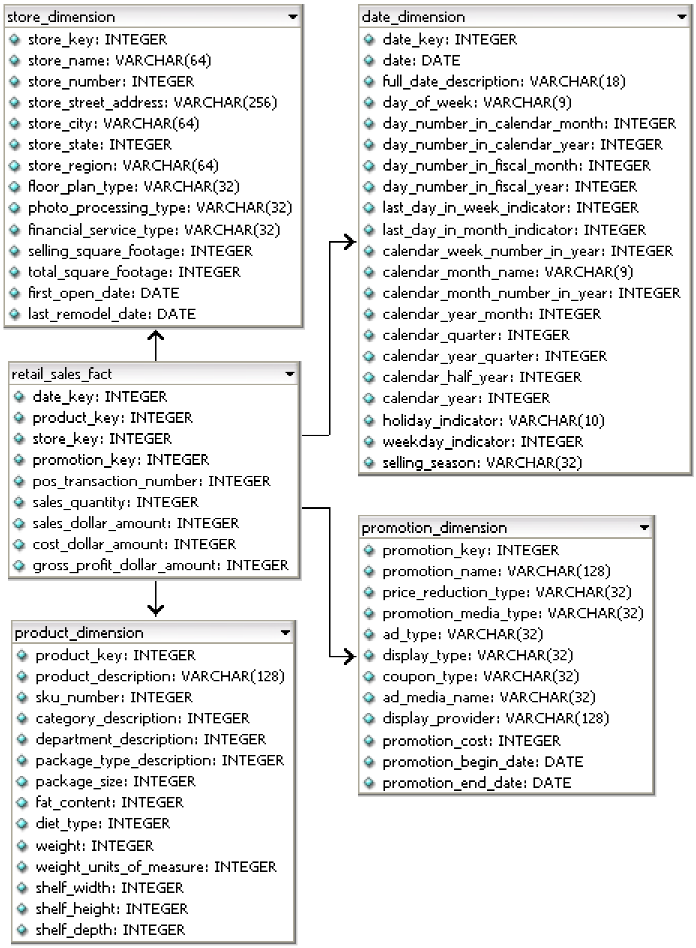
```
-- Overdue statistics for 2001 by Customer age group
-- (in 5-year intervals)
-- a. Avg Overdue (Amount and Days)
-- b. Max Overdue (Amount and Days)
-- c. Min Overdue (Amount and Days)


SELECT   (Age - MOD(age, 5)) AS Age_Group,
         COUNT(DISTINCT A.Customer_Key) AS Num_Customers,
         MAX(Days_Overdue) AS Max_Days,
         MIN(Days_Overdue) AS Min_Days,
         AVG(Days_Overdue) AS Avg_Days,
         MAX(Outstanding_Amount) AS Max_Amount,
         MIN(Outstanding_Amount) AS Min_Amount,
         AVG(Outstanding_Amount) AS Avg_Amount,
         COUNT(*) AS Overdue_Recs
FROM     CreditHistory_Fact A,
         Institution_Dimension B,
         Date_Dimension C,
         Customer_Dimension D
WHERE    A.Date_Key = C.Date_Key
         AND A.Institution_Key = B.Institution_Key
         AND A.Customer_Key = D.Customer_Key
         AND C.Calendar_Year = 2001
GROUP BY (Age - MOD(age, 5))
ORDER BY (Age - MOD(age, 5));
```

Example

```
age_group  | num_customers | max_days | min_days |     avg_days     | max_amount | min_amount |
avg_amount    | overdue_recs
-----------+---------------+----------+----------+------------------+------------+------------+--
----------------+--------------
        15 |           222 |      999 |        0 | 497.936727480296 |   15000.93 |     500.23 |
7766.68663241784 |        45549
        20 |           628 |      999 |        0 | 501.106663765097 |      15001 |     500.01 |
7771.59768322634 |       128666
        25 |           507 |      999 |        0 | 499.240936955664 |   15000.96 |     500.03 |
7766.86130486666 |       103911
        30 |           585 |      999 |        0 | 500.578115364744 |   15000.98 |     500.03 |
7747.69135839403 |       120002
        35 |           582 |      999 |        0 | 499.998786631186 |   15000.98 |     500.15 |
7740.60186540811 |       119502
        40 |           596 |      999 |        0 | 500.107618470409 |    15000.9 |     500.06 |
7766.74586684467 |       122098
        45 |           583 |      999 |        0 | 499.767012348991 |   15000.92 |        500 |
7745.33969399988 |       119281
        50 |           586 |      999 |        0 | 498.041474731711 |   15000.97 |     500.03 |
7748.55477013954 |       119181
        55 |           583 |      999 |        0 | 498.440118960102 |    15000.9 |     500.07 |
7743.21703075921 |       119704
        60 |           128 |      999 |        0 | 499.436279703345 |   15000.92 |     500.14 |
7794.25127364916 |        26428
(10 rows)
```

Retail Sales Example Database

The Retail Sales Example Database is based on a fictional retail grocery chain store. It a simple star schema that represents individual line items on POS (Point of Sale) transactions. Each tuple in the fact table represents an item purchased from a store. Each table is described in a separate section.



store_dimension
- store_key: INTEGER
- store_name: VARCHAR(64)
- store_number: INTEGER
- store_street_address: VARCHAR(256)
- store_city: VARCHAR(64)
- store_state: INTEGER
- store_region: VARCHAR(64)
- floor_plan_type: VARCHAR(32)
- photo_processing_type: VARCHAR(32)
- financial_service_type: VARCHAR(32)
- selling_square_footage: INTEGER
- total_square_footage: INTEGER
- first_open_date: DATE
- last_remodel_date: DATE

date_dimension
- date_key: INTEGER
- date: DATE
- full_date_description: VARCHAR(18)
- day_of_week: VARCHAR(9)
- day_number_in_calendar_month: INTEGER
- day_number_in_calendar_year: INTEGER
- day_number_in_fiscal_month: INTEGER
- day_number_in_fiscal_year: INTEGER
- last_day_in_week_indicator: INTEGER
- last_day_in_month_indicator: INTEGER
- calendar_week_number_in_year: INTEGER
- calendar_month_name: VARCHAR(9)
- calendar_month_number_in_year: INTEGER
- calendar_year_month: INTEGER
- calendar_quarter: INTEGER
- calendar_year_quarter: INTEGER
- calendar_half_year: INTEGER
- calendar_year: INTEGER
- holiday_indicator: VARCHAR(10)
- weekday_indicator: INTEGER
- selling_season: VARCHAR(32)

retail_sales_fact
- date_key: INTEGER
- product_key: INTEGER
- store_key: INTEGER
- promotion_key: INTEGER
- pos_transaction_number: INTEGER
- sales_quantity: INTEGER
- sales_dollar_amount: INTEGER
- cost_dollar_amount: INTEGER
- gross_profit_dollar_amount: INTEGER

promotion_dimension
- promotion_key: INTEGER
- promotion_name: VARCHAR(128)
- price_reduction_type: VARCHAR(32)
- promotion_media_type: VARCHAR(32)
- ad_type: VARCHAR(32)
- display_type: VARCHAR(32)
- coupon_type: VARCHAR(32)
- ad_media_name: VARCHAR(32)
- display_provider: VARCHAR(128)
- promotion_cost: INTEGER
- promotion_begin_date: DATE
- promotion_end_date: DATE

product_dimension
- product_key: INTEGER
- product_description: VARCHAR(128)
- sku_number: INTEGER
- category_description: INTEGER
- department_description: INTEGER
- package_type_description: INTEGER
- package_size: INTEGER
- fat_content: INTEGER
- diet_type: INTEGER
- weight: INTEGER
- weight_units_of_measure: INTEGER
- shelf_width: INTEGER
- shelf_height: INTEGER
- shelf_depth: INTEGER

Table Name	Default Number of Rows
Retail_Sales_Fact (on page 32)	5000000
Product_Dimension (on page 71)	60000
Store_Dimension (on page 34)	250

Promotion_Dimension (on page 72)	1000

Retail_Sales_Fact

The Retail_Sales_Fact table describes individual items purchased from a grocery store. The generated data file contains data for five million items purchased by default.

Column Name	Data Type
Date_Key	INTEGER
Product_key	INTEGER
Store_key	INTEGER
Promotion_key	INTEGER
Pos_transaction_number	INTEGER
Sales_quantity	INTEGER
Sales_dollar_amount	INTEGER
Cost_dollar_amount	INTEGER
Gross_profit_dollar_amount	INTEGER

Date_Dimension

The Date Dimension table contains data for 1,828 dates for the years 2000-2004. It is generated from a file containing correct date/time data.

Column Name	Data Type	Description/Example
Date_Key	INTEGER	1
Date	DATE	01/01/2000
Full_date_description	VARCHAR(18)	January 1, 2000
Day_of_week	VARCHAR(9)	Sunday
Day_number_in_calendar_month	INTEGER	1
Day_number_in_calendar_year	INTEGER	1
Day_number_in_fiscal_month	INTEGER	1
Day_number_in_fiscal_year	INTEGER	1
Last_day_in_week_indicator	INTEGER	1
Last_day_in_month_indicator	INTEGER	0

Calendar_week_number_in_year	INTEGER	52
Calendar_month_name	VARCHAR(9)	January
Calendar_month_number_in_year	INTEGER	1
Calendar_year_month	CHAR(7)	2000-1
Calendar_quarter	INTEGER	1
Calendar_year_quarter	CHAR(7)	2000-q1
Calendar_half_year	INTEGER	1
Calendar_year	INTEGER	2000
Holiday_indicator	VARCHAR(10)	Holiday
Weekday_indicator	CHAR(7)	Weekend
Selling_season	VARCHAR(32)	Valentines Day

Product_Dimension

The Product Dimension table describes all products sold by the grocery chain since its beginning. Typically, individual stores only carry a subset of the products. The generated data file contains data for 60,000 products by default.

Column Name	Data Type	Description/Example
Product_Key	INTEGER	1
Product_description	VARCHAR(128)	Seafood Product 1
Sku_number	CHAR(32)	Sku-#1
Category_description	CHAR(32)	Food
Department_description	CHAR(32)	Seafood
Package_type_description	CHAR(32)	Box
Package_size	CHAR(32)	18 Oz
Fat_content	INTEGER	89
Diet_type	CHAR(32)	South Beach
Weight	INTEGER	50
Weight_units_of_measure	CHAR(32)	Gram
Shelf_width	INTEGER	2
Shelf_height	INTEGER	4
Shelf_depth	INTEGER	4

Promotion_Dimension

The Promotion Dimension describes every promotion (announced temporary price reduction) ever done by the grocery chain. The generated data file contains data for one thousand promotions by default.

Column Name	Data Type	Description/Example
Promotion_Key	INTEGER	1
Product_description	VARCHAR(128)	Seafood Product 1
Promotion_name	VARCHAR(128)	July 4th Liquidation Promotion
Price_reduction_type	VARCHAR(32)	20 Cents Off
Promotion_media_type	VARCHAR(32)	Magazine
Ad_type	VARCHAR(32)	1 Minute
Display_type	VARCHAR(32)	Pos
Coupon_type	VARCHAR(32)	Register Receipt
Ad_media_name	VARCHAR(32)	Other
Display_provider	VARCHAR(128)	Corporate
Promotion_cost	INTEGER	492
Promotion_begin_date	DATE	3-6-2001
Promotion_end_date	DATE	3-15-2001

Store_Dimension

The Store Dimension table describes all the stores in the chain. The generated data file contains data for 250 stores by default.

Column Name	Data Type	Example
`Store_Key`	INTEGER	1
`Store_name`	VARCHAR(64)	Store1
`Store_name`	VARCHAR(64)	Store1
`Store_number`	INTEGER	1
`Store_street_address`	VARCHAR(256)	3, Main St
`Store_city`	VARCHAR(64)	Concord
`Store_state`	CHAR(2)	Ca
`Store_region`	VARCHAR(64)	West
`Floor_plan_type`	VARCHAR(32)	Plan1
`Photo_processing_type`	VARCHAR(32)	Premium
`Financial_service_type`	VARCHAR(32)	None
`Selling_square_footage`	INTEGER	100
`Total_square_footage`	INTEGER	2000
`First_open_date`	DATE	3-1-2004
`Last_remodel_date`	DATE	null

retail_query_01.sql

This query joins the fact table (five million rows) with one dimension table (1,828 rows).

Query

```
-- The best day of the week in gross profit
-- for each year of operation.

SELECT   Calendar_Year,
         Day_Of_Week,
         SUM(Gross_Profit_Dollar_Amount) AS Profit
FROM     Retail_Sales_Fact,
         Date_Dimension
WHERE    Retail_Sales_Fact.Date_Key = Date_Dimension.Date_Key
GROUP BY Calendar_Year,Day_Of_Week
ORDER BY Calendar_Year,
         Profit DESC;
```

Example

```
Retail_Single_Node=> \i retail_query_01.sql
 calendar_year | day_of_week |  profit
---------------+-------------+----------
          2000 | Sunday      | 24610107
          2000 | Tuesday     | 24389067
          2000 | Thursday    | 23973851
          2000 | Friday      | 23392757
          2000 | Saturday    | 22134302
          2000 | Wednesday   | 21427790
          2000 | Monday      | 20650172
          2001 | Thursday    | 24057786
          2001 | Sunday      | 22808366
          2001 | Friday      | 22262470
          2001 | Tuesday     | 21207805
          2001 | Wednesday   | 20648615
          2001 | Saturday    | 20522518
          2001 | Monday      | 16566382
          2002 | Saturday    | 23068736
          2002 | Wednesday   | 22749773
          2002 | Monday      | 22728810
          2002 | Sunday      | 20862246
          2002 | Friday      | 20825621
          2002 | Tuesday     | 20034320
          2002 | Thursday    | 18856255
          2003 | Friday      | 24563166
          2003 | Tuesday     | 22913972
          2003 | Wednesday   | 22255964
          2003 | Thursday    | 21596220
          2003 | Saturday    | 21039048
          2003 | Monday      | 20685036
          2003 | Sunday      | 20529061
          2004 | Friday      | 23675620
          2004 | Saturday    | 22815560
          2004 | Wednesday   | 21332928
          2004 | Tuesday     | 21303355
          2004 | Sunday      | 21190484
          2004 | Monday      | 20863037
          2004 | Thursday    | 20419213
(35 rows)
```

retail_query_02.sql

This query joins five million rows of fact table data with three dimension tables (1,828 rows, 250 rows, and 1,000 rows).

Query

```
-- Promotion Profits by Year, Month, and Region

SELECT   Calendar_Year,
         Calendar_Month_Name,
         Store_Region,
         Promotion_Name,
         SUM(Gross_Profit_Dollar_Amount) AS Profit
FROM     Retail_Sales_Fact POS_Fact,
         Date_Dimension Date_Dim,
         Store_Dimension Store_Dim,
         Promotion_Dimension Prom_Dim
WHERE    POS_Fact.Date_Key = Date_Dim.Date_Key
         AND POS_Fact.Store_Key = Store_Dim.Store_Key
         AND POS_Fact.Promotion_Key = Prom_Dim.Promotion_Key
GROUP BY Calendar_Year,
         Calendar_Month_Name,
         Promotion_Name,
         Store_Region
HAVING   SUM(Gross_Profit_Dollar_Amount) >= 4500
ORDER BY Profit DESC;
```

Example

```
Retail_Single_Node=> \a
Output format is unaligned.
Retail_Single_Node=> \i retail_query_02.sql
 calendar_year  | calendar_month_name  |  store_region  |  promotion_name               |  profit
----------------+----------------------+----------------+-------------------------------+--------
--
2000            | January              | West           | Summer Cool Sale              | 97451
2000            | October              | West           | July 4th Discount Sale        | 96588
2003            | March                | West           | Thanksgiving Super Sellathon  | 96169
2000            | January              | West           | Thanksgiving Super Sellathon  | 95184
2000            | October              | West           | Thanksgiving Super Sellathon  | 95134
2000            | January              | West           | July 4th Super Sale           | 94871
2000            | December             | West           | Summer Liquidation Promotion  | 94343
2000            | January              | West           | Summer Liquidation Promotion  | 94014
2000            | January              | West           | July 4th Cool Sellathon       | 92744
2004            | January              | West           | Summer Cool Sale              | 92659
2004            | January              | West           | Thanksgiving Super Sellathon  | 92310
2000            | October              | West           | Summer Liquidation Promotion  | 91872
2001            | August               | West           | Thanksgiving Super Sellathon  | 91837
2001            | May                  | West           | Thanksgiving Super Sellathon  | 91389
2004            | January              | West           | Summer Liquidation Promotion  | 90615
2000            | December             | West           | Thanksgiving Super Sellathon  | 90423
2004            | January              | West           | July 4th Discount Sellathon   | 90282
2003            | December             | West           | Thanksgiving Super Sellathon  | 89181
2004            | December             | West           |Thanksgiving Super Sellathon   | 88236
(20 rows)
Retail_Single_Node=>
```

retail_query_03.sql

This query joins five million rows of fact table data with four dimension tables.

Query

```
-- Most Profitable Seafood Products in the East in 2003

SELECT   Product_Description,
         SUM(Gross_Profit_Dollar_Amount) AS Profit
FROM     Retail_Sales_Fact,
         Product_Dimension,
         Store_Dimension,
         Date_Dimension
WHERE    Retail_Sales_Fact.Product_Key = Product_Dimension.Product_Key
         AND Retail_Sales_Fact.Store_Key = Store_Dimension.Store_Key
         AND Retail_Sales_Fact.Date_Key = Date_Dimension.Date_Key
         AND Department_Description = 'Seafood'
         AND Store_Region = 'East'
         AND Calendar_Year = 2003
GROUP BY Store_Region,
         Product_Description
ORDER BY Store_Region,
         Profit DESC;
```

Example

```
Retail_Single_Node=> \i retail_query_03.sql
------------------------------------------------------

  product_description  | profit
```

```
-----------------------+--------
 Seafood Product 10370 |   2432
 Seafood Product 47983 |   2331
 Seafood Product 43929 |   2095
 Seafood Product 6474  |   2008
 Seafood Product 18213 |   1976
 Seafood Product 53224 |   1935
 Seafood Product 57425 |   1896
 Seafood Product 10608 |   1888
 Seafood Product 2989  |   1869
 Seafood Product 258   |   1812
 Seafood Product 25835 |   1809
 Seafood Product 40207 |   1794
 Seafood Product 16271 |   1794
 Seafood Product 1429  |   1791
 Seafood Product 58142 |   1777
 Seafood Product 33695 |   1772
 Seafood Product 20455 |   1765
 Seafood Product 12616 |   1757
 Seafood Product 57498 |   1750
 Seafood Product 29837 |   1748
 Seafood Product 53700 |   1745
 Seafood Product 31991 |   1733
 Seafood Product 16584 |   1731
 Seafood Product 19347 |   1724
 Seafood Product 25424 |   1719
 Seafood Product 49094 |   1694
 Seafood Product 57111 |   1683
 Seafood Product 53686 |   1681
 Seafood Product 32016 |   1680
 Seafood Product 48506 |   1676
 Seafood Product 12294 |   1669
 Seafood Product 21983 |   1667
 Seafood Product 30662 |   1666
 Seafood Product 30073 |   1663
 Seafood Product 27621 |   1662
 Seafood Product 37650 |   1650
 Seafood Product 37755 |   1645
 Seafood Product 32757 |   1644
 Seafood Product 21454 |   1636
 Seafood Product 50994 |   1632
 Seafood Product 32028 |   1630
 Seafood Product 41263 |   1626
 Seafood Product 6438  |   1606
 Seafood Product 57315 |   1605
 Seafood Product 11539 |   1605
 Seafood Product 51685 |   1603
 Seafood Product 34664 |   1600
 Seafood Product 5798  |   1591
```

retail_query_04.sql

Query

```
-- Query 04
-- Report: Maximum sales_dollar_amount according to department

SELECT   a12.department_description AS department_description,
         a12.category_description AS category_description,
         sum(a11.sales_dollar_amount) AS total_sales_dollar_amount,
         max(a11.sales_dollar_amount) AS max_sales_dollar_amount
FROM     retail_sales_fact a11,
         product_dimension a12
WHERE    a11.product_key = a12.product_key
```

```
GROUP BY a12.department_description,
         a12.category_description
ORDER BY a12.department_description;
```

Example

```
department_description  |  category_description  | total_sales_dollar_amount |
max_sales_dollar_amount
------------------------+-------------------------+---------------------------+-------------------
------
 Bakery                 | Food                    |                 116489955 |
600
 Canned Goods           | Food                    |                 115699108 |
600
 Cleaning supplies      | Non-food                |                 113700725 |
600
 Dairy                  | Food                    |                 117254596 |
600
 Frozen Goods           | Food                    |                 116859512 |
600
 Gifts                  | Misc                    |                 115800323 |
600
 Liquor                 | Non-food                |                 118948581 |
600
 Meat                   | Food                    |                 119924642 |
600
 Medical                | Medical                 |                 115532701 |
600
 Pharmacy               | Medical                 |                 119401892 |
600
 Photography            | Misc                    |                 113603404 |
600
 Produce                | Food                    |                 113376462 |
600
 Seafood                | Food                    |                 119005848 |
600
(13 rows)
```

retail_query_05.sql

Query

```
-- Query 05
-- Top 5 Stores in a quarter on the basis of gross profit --

SELECT   a12.calendar_year_quarter AS calendar_year_quarter,
         a11.store_key AS store_key,
         max(a13.store_name) AS store_name,
         sum(a11.sales_quantity) AS total_sales_quantity,
         sum(a11.sales_dollar_amount) AS total_sales_dollar_amount,
         sum(a11.cost_dollar_amount) AS total_cost_dollar_amount,
         sum(a11.gross_profit_dollar_amount) AS total_gross_profit
FROM     retail_sales_fact a11,
         date_dimension a12,
         store_dimension a13
WHERE    a11.date_key = a12.date_key
         AND a11.store_key = a13.store_key
         AND a12.calendar_year_quarter = '2004-Q4'
GROUP BY a12.calendar_year_quarter,
         a11.store_key
```

```
ORDER BY total_gross_profit DESC
LIMIT    5;
```

Example

```
calendar_year_quarter | store_key | store_name | total_sales_quantity | total_sales_dollar_amount |
total_cost_dollar_amount | total_gross_profit
----------------------+-----------+------------+----------------------+------------------------
--+--------------------------+-------------------
 2004-Q4              |       247 | Store247   |                 5844 |                    333410
|                   164461 |             168949
 2004-Q4              |       239 | Store239   |                 5772 |                    328880
|                   162774 |             166106
 2004-Q4              |       234 | Store234   |                 5828 |                    332649
|                   166579 |             166070
 2004-Q4              |       168 | Store168   |                 5995 |                    325382
|                   159744 |             165638
 2004-Q4              |        15 | Store15    |                 5864 |                    324643
|                   159716 |             164927
(5 rows)
```

retail_query_06.sql

Query

```
-- Query 06
-- Region & state wise profit per unit --

SELECT   a12.store_region AS store_region,
         a12.store_state AS store_state,
         sum(a11.gross_profit_dollar_amount) AS
total_gross_profit_dollar_amount,
         sum(a11.sales_quantity) AS total_sales_quantity,
         ROUND((sum(a11.gross_profit_dollar_amount)::float /
sum(a11.sales_quantity)), 2)
           AS avg_profit_per_unit
FROM     retail_sales_fact a11,
         store_dimension a12
WHERE    a11.store_key = a12.store_key
GROUP BY a12.store_region,
         a12.store_state
ORDER BY a12.store_region,
         a12.store_state;
```

Example

```
store_region | store_state | total_gross_profit_dollar_amount | total_sales_quantity |
avg_profit_per_unit
--------------+-------------+----------------------------------+----------------------+----------
-----------
 East         | CT          |                         27419268 |               988967 |
27.73
 East         | DC          |                          6083983 |               219193 |
27.76
 East         | MA          |                         24425494 |               877070 |
27.85
 East         | MD          |                         18224018 |               657368 |
27.72
 East         | MI          |                         12192531 |               440660 |
27.67
```

```
 East         | NC          |                           9225154 |               332518 |
27.74
 East         | NH          |                          12227838 |               442973 |
27.6
 East         | NJ          |                           9153991 |               329633 |
27.77
 East         | NY          |                           3021472 |               108867 |
27.75
 East         | PA          |                          18426900 |               663897 |
27.76
 East         | SC          |                          12200472 |               439344 |
27.77
 East         | TN          |                          27424332 |               989000 |
27.73
 East         | VA          |                          12220511 |               439039 |
27.83
 MidWest      | IA          |                           3077406 |               111375 |
27.63
 MidWest      | IL          |                          36419633 |              1310671 |
27.79
 MidWest      | IN          |                          27569070 |               993308 |
27.75
 MidWest      | MI          |                          45764788 |              1653583 |
27.68
 MidWest      | OH          |                           6150916 |               219351 |
28.04
 MidWest      | SD          |                          12162880 |               437176 |
27.82
 MidWest      | WI          |                          12146497 |               437395 |
27.77
 NorthWest    | OR          |                           6090896 |               220814 |
27.58
 NorthWest    | WA          |                           3104690 |               110723 |
28.04
 South        | FL          |                          24266821 |               877191 |
27.66
 South        | GA          |                          24364400 |               880309 |
27.68
 South        | LA          |                           6080205 |               220558 |
27.57
 South        | MS          |                           3044063 |               110011 |
27.67
 South        | TX          |                          70301249 |              2536343 |
27.72
 SouthWest    | AZ          |                          15291817 |               551088 |
27.75
 SouthWest    | CO          |                          33598118 |              1208581 |
27.8
 SouthWest    | KS          |                           6109911 |               221061 |
27.64
 SouthWest    | NV          |                          12207238 |               439893 |
27.75
 West         | CA          |                         201597518 |              7262311 |
27.76
 West         | UT          |                          21366333 |               769336 |
27.77
(33 rows)
```

retail_query_07.sql

Query

```
-- Query 07
-- Listing of sales quantity of all products with id less than 100 --

SELECT   a11.product_key AS product_key,
         max(a12.product_description) AS product_description,
         max(a12.package_size) AS package_size,
         max(a12.weight_units_of_measure) AS weight_units_of_measure,
         sum(a11.sales_quantity) AS total_sales_quantity
FROM     retail_sales_fact a11, product_dimension a12
WHERE    a11.product_key = a12.product_key
         AND a11.product_key < 100
GROUP BY a11.product_key
ORDER BY a11.product_key;
```

Example

```
product_key |  product_description      |  package_size   | weight_units_of_measure   |
total_sales_quantity
------------+---------------------------+-----------------+---------------------------+-----------
-----------
          1 | Seafood Product 1         | Other           | gram                      | 487
          2 | Seafood Product 2         | 18 oz           | gram                      | 491
          3 | Seafood Product 3         | 1 liter         | ounce                     | 408
          4 | Medical Product 4         | Other           | pound                     | 561
          5 | Seafood Product 5         | 1 gallon        | pound                     | 540
          6 | Meat Product 6            | Economy         | gram                      | 576
          7 | Medical Product 7         | Other           | pound                     | 465
          8 | Photography Product 8     | Other           | pound                     | 504
```

retail_query_08.sql

Query

```
-- Query 08
-- Quarterly and monthly sales for promotional schemes --

SELECT   a13.calendar_year_quarter AS calendar_year_quarter,
         a13.calendar_month_name AS calendar_month_name,
         a12.ad_type AS ad_type,
         sum(a11.sales_dollar_amount) AS total_sales_dollar_amount,
         sum(a11.sales_quantity) AS total_sales_quantity
FROM     retail_sales_fact a11,
         promotion_dimension a12,
         date_dimension a13
WHERE    a11.promotion_key = a12.promotion_key
         AND a11.date_key = a13.date_key
GROUP BY a13.calendar_year_quarter,
         a13.calendar_month_name,
         a13.calendar_month_number_in_year,
         a12.ad_type
```

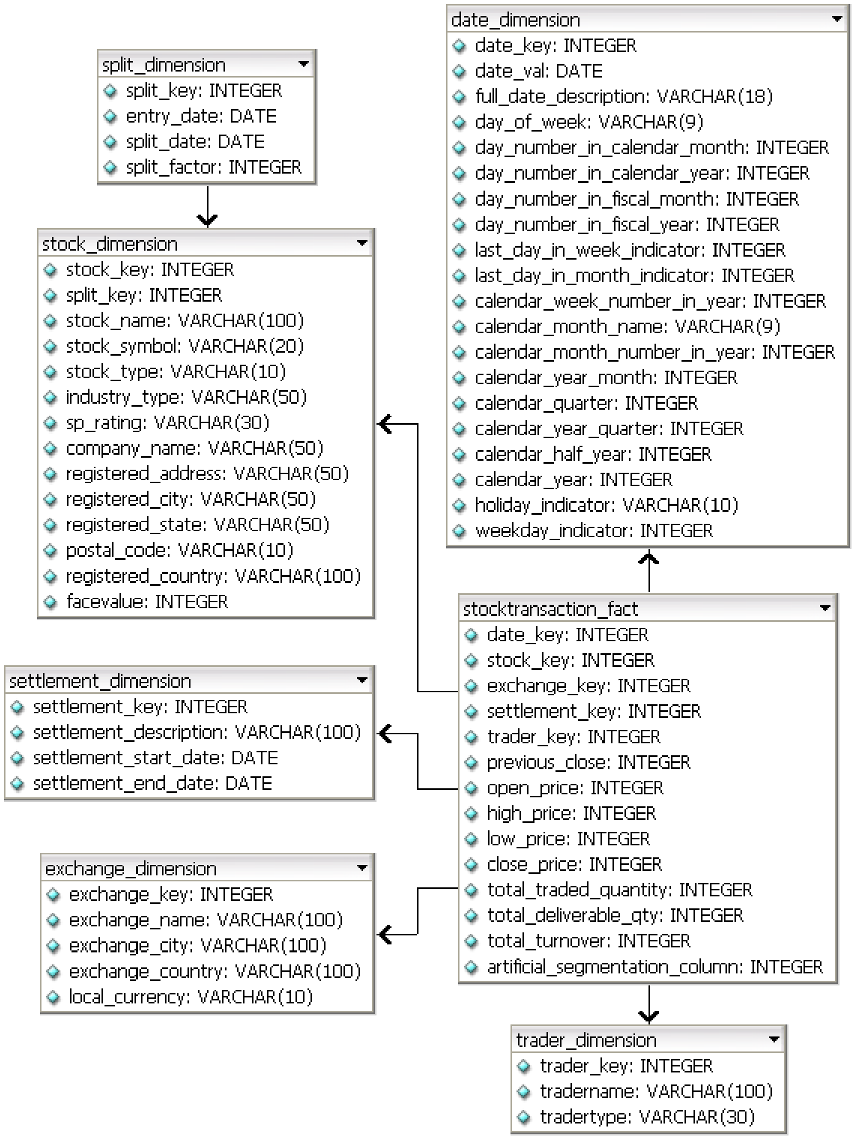
```
ORDER BY a13.calendar_year_quarter,
         a13.calendar_month_number_in_year,
         a12.ad_type;
```

Example

```
calendar_year_quarter | calendar_month_name |  ad_type   | total_sales_dollar_amount |
total_sales_quantity
-----------------------+---------------------+------------+---------------------------+----------
------------
 2000-Q1               | January             | 1 minute   |                   7858712 |
142398
 2000-Q1               | January             | 30 seconds |                   7805195 |
140665
 2000-Q1               | January             | Fullpage   |                   8622042 |
156397
 2000-Q1               | January             | Halfpage   |                   7710140 |
139833
 2000-Q1               | February            | 1 minute   |                   5712781 |
103815
 2000-Q1               | February            | 30 seconds |                   5674174 |
103718
 2000-Q1               | February            | Fullpage   |                   6327306 |
114023
 2000-Q1               | February            | Halfpage   |                   5623861 |
101407
 2000-Q1               | March               | 1 minute   |                   6174128 |
112232
 2000-Q1               | March               | 30 seconds |                   6274003 |
113892
 2000-Q1               | March               | Fullpage   |                   6890512 |
124231
 2000-Q1               | March               | Halfpage   |                   6064541 |
110426
 2000-Q2               | April               | 1 minute   |                   5990593 |
108544
 2000-Q2               | April               | 30 seconds |                   5976581 |
107663
 2000-Q2               | April               | Fullpage   |                   6548556 |
118904
 2000-Q2               | April               | Halfpage   |                   5834707 |
105976
 2000-Q2               | May                 | 1 minute   |                   6496888 |
117450
 2000-Q2               | May                 | 30 seconds |                   6386322 |
116323
 2000-Q2               | May                 | Fullpage   |                   7106788 |
129226
 2000-Q2               | May                 | Halfpage   |                   6332820 |
115019
 2000-Q2               | June                | 1 minute   |                   6435020 |
116730
 2000-Q2               | June                | 30 seconds |                   6451742 |
117210
 2000-Q2               | June                | Fullpage   |                   7143086 |
128928
 2000-Q2               | June                | Halfpage   |                   6262982 |
114252
 2000-Q3               | July                | 1 minute   |                   6558197 |
119295
 2000-Q3               | July                | 30 seconds |                   6497357 |
117877
 2000-Q3               | July                | Fullpage   |                   7284518 |
131812
…
```

Stock Exchange Example Database

The Stock Exchange schema is a simple star schema that represents summary of trades done during the day by various business such as banks, insurance companies, retail investors, mutual funds, and so on. It is commonly known as a "tick store." Each table is described in a separate section.


split_dimension
split_key: INTEGER
entry_date: DATE
split_date: DATE
split_factor: INTEGER
date_dimension
date_key: INTEGER
date_val: DATE
full_date_description: VARCHAR(18)
day_of_week: VARCHAR(9)
day_number_in_calendar_month: INTEGER
day_number_in_calendar_year: INTEGER
day_number_in_fiscal_month: INTEGER
day_number_in_fiscal_year: INTEGER
last_day_in_week_indicator: INTEGER
last_day_in_month_indicator: INTEGER
calendar_week_number_in_year: INTEGER
calendar_month_name: VARCHAR(9)
calendar_month_number_in_year: INTEGER
calendar_year_month: INTEGER
calendar_quarter: INTEGER
calendar_year_quarter: INTEGER
calendar_half_year: INTEGER
calendar_year: INTEGER
holiday_indicator: VARCHAR(10)
weekday_indicator: INTEGER
stock_dimension
stock_key: INTEGER
split_key: INTEGER
stock_name: VARCHAR(100)
stock_symbol: VARCHAR(20)
stock_type: VARCHAR(10)
industry_type: VARCHAR(50)
sp_rating: VARCHAR(30)
company_name: VARCHAR(50)
registered_address: VARCHAR(50)
registered_city: VARCHAR(50)
registered_state: VARCHAR(50)
postal_code: VARCHAR(10)
registered_country: VARCHAR(100)
facevalue: INTEGER
stocktransaction_fact
date_key: INTEGER
stock_key: INTEGER
exchange_key: INTEGER
settlement_key: INTEGER
trader_key: INTEGER
previous_close: INTEGER
open_price: INTEGER
high_price: INTEGER
low_price: INTEGER
close_price: INTEGER
total_traded_quantity: INTEGER
total_deliverable_qty: INTEGER
total_turnover: INTEGER
artificial_segmentation_column: INTEGER
settlement_dimension
settlement_key: INTEGER
settlement_description: VARCHAR(100)
settlement_start_date: DATE
settlement_end_date: DATE
exchange_dimension
exchange_key: INTEGER
exchange_name: VARCHAR(100)
exchange_city: VARCHAR(100)
exchange_country: VARCHAR(100)
local_currency: VARCHAR(10)
trader_dimension
trader_key: INTEGER
tradername: VARCHAR(100)
tradertype: VARCHAR(30)

StockTransaction_Fact

Each record in the fact table represents summary of stocks traded in a day.

Field Name	Data Type	Description
Date_key	INTEGER	Date Key
Exchange_Key	INTEGER	Foreign Key, references Exchange table
Settlement_Key	INTEGER	Foreign Key, references Settlement table
Trader_Key	INTEGER	Foreign Key, references Trader Table
Stock_Key	INTEGER	Foreign Key, references Stock Dimension table
Previous_Close	FLOAT	Previous close of the Script
Open_Price	FLOAT	Opening price of Script for the given day
High_Price	FLOAT	High price of Script for the given day
Low_Price	FLOAT	Low price of Script for the given day
Close_Price	FLOAT	Closing price of Script for the given day
Total_Traded_Quantity	FLOAT	Total traded quantity of the Script for the given day
Total_ Deliverable Qty	FLOAT	Total Deliverable quantity Script for the given day
Total_Turnover	FLOAT	Total value of transactions for the given day
artificial_segmentation_ column	INTEGER	Generated values for load-balancing nodes

Date_Dimension

The Date Dimension table contains data for dates.

Field Name	Data Type	Description
Date_Key	INTEGER	Primary Key
Date	DATE	
Full_date_description	VARCHAR(18)	
Day_of_week	VARCHAR(9)	
Day_number_in_calendar_month	INTEGER	
Day_number_in_calendar_year	INTEGER	
Day_number_in_fiscal_month	INTEGER	
Day_number_in_fiscal_year	INTEGER	
Last_day_in_week_indicator	INTEGER	
Last_day_in_month_indicator	INTEGER	
Calendar_week_number_in_year	INTEGER	
Calendar_month_name	VARCHAR(9)	
Calendar_month_number_in_year	INTEGER	
Calendar_year_month	CHAR(7)	
Calendar_quarter	INTEGER	
Calendar_year_quarter	CHAR(7)	
Calendar_half_year	INTEGER	
Calendar_year	INTEGER	
Holiday_indicator	VARCHAR(10)	
Weekday_indicator	CHAR(7)	

Exchange_Dimension

This table describes the stock exchanges included in the fact table.

Field Name	Data Type	Description
Exchange_Key	INTEGER	Primary Key
Exchange_Name	VARCHAR	Complete Exchange name
Exchange_City	VARCHAR	City where exchange is located
Exchange_Country	VARCHAR	Country where exchange is located
Local_Currency	VARCHAR	Local currency of where exchange is located

Settlement_Dimension

This table describes the types of settlements.

Field Name	Data Type	Description
Settlement_Key	INTEGER	Primary Key
Settlement_Description	VARCHAR	Exchange specific Settlement Number in which all transactions of specific period have to be settled
Settlement_Start_Date	DATE	Settlement Start Date
Settlement_End_Date	DATE	Settlement End Date

Split_Dimension

This tables contains stock split dates and factors.

Field Name	Data Type	Description
Split_Id	INTEGER	Primary Key
EntryDate	DATE	Date the split is announced.
SplitDate	DATE	Date the split is actually effective.
SplitFactor	FLOAT	The split factor expressed as a decimal value. For example, a 2 for 1 split is expressed as 0.5 and a 4 for 3 is expressed as 0.75.

Stock Dimension

Describes all publicly traded stocks in stock exchanges.

Field Name	Data Type	Description
Stock_Key	INTEGER	Primary Key
Split_Key	INTEGER	Foreign Key references Split_Dimension Table
Stock_Name	VARCHAR	Publicly traded stock name
Stock_ Symbol	VARCHAR	Symbol of traded security
Stock_Type	VARCHAR	Equity/Bond
Industry_Type	VARCHAR	Chemical/Computers/Steel
SP_Rating	VARCHAR	S&P Rating, ‘AAA’,AA,A B+, B etc.
Company_Name	VARCHAR	Complete name of company
Registered_Address	VARCHAR	Complete address where the company is registered
Registered_City	VARCHAR	City where company is registered
Registered_State	VARCHAR	State where company is registered
Postal_Code	VARCHAR	Postal code
Registered_County	VARCHAR	Country where company is registered
FaceValue	INTEGER	Issue Price of stock in country where company is located; for example, $1, $5

Trader_Dimension

This table describes the institutions that trade stocks.

Field Name	Data Type	Description
Trader_Key	INTEGER	Primary Key
TraderName	VARCHAR	Name of institution.
TraderType	VARCHAR	Type of trader (broker, bank, insurance company, etc.)

stock_query_01

Query

```
--- QUERY #1
--- Stocks that gained between 70% and 75% on a given day
SELECT B.Stock_Name,
       MIN(A.Close_Price),
       MAX(A.Close_Price)
FROM     StockTransaction_Fact A,
         Stock_Dimension B
WHERE    A.Date_Key > 50
         AND A.Date_Key < 53
         AND A.Stock_Key = B.Stock_Key
         AND (((A.close_Price - A.Previous_Close) * 100) / A.Previous_Close) > 70
         AND (((A.close_Price - A.Previous_Close) * 100) / A.Previous_Close) < 75
GROUP BY B.Stock_Name
ORDER BY B.Stock_Name;
```

Example

```
   stock_name    |  min  |  max
-----------------+-------+-------
 STOCK_NAME#229 | 70.42 | 70.42
 STOCK_NAME#50  | 70.12 | 70.12
(2 rows)
```

stock_query_02

Query

```
--- QUERY #2
--- Total traded quantity and value of stock in a
--- given settlement period
```

```
SELECT Settlement_Description,
       Stock_Name,
       SUM(Total_Traded_Quantity)  AS Total_Traded_Qty,
       SUM(Total_Turnover)         AS Total_Trade_value
FROM     StockTransaction_Fact A,
         Settlement_Dimension B,
         Stock_Dimension C,
         Date_Dimension D
WHERE    A.Settlement_Key = B.Settlement_Key
         AND A.Stock_Key = C.Stock_Key
         AND A.Date_Key = D.Date_Key
         AND B.Settlement_Description = '2000010'
         AND D.Calendar_Month_Number_in_Year = 1
         AND D.Calendar_Year = 2004
GROUP BY Settlement_Description,
         Stock_Name
ORDER BY Settlement_Description,
         Stock_Name;
```

Example

```
settlement_description |   stock_name   | total_traded_qty | total_trade_value
------------------------+----------------+------------------+-------------------
 2000010                | STOCK_NAME#1   |             1319 |          14927.65
 2000010                | STOCK_NAME#100 |              913 |           8832.14
 2000010                | STOCK_NAME#101 |             1236 |          23758.96
 2000010                | STOCK_NAME#102 |               42 |          12155.77
 2000010                | STOCK_NAME#103 |              828 |          12645.93
 2000010                | STOCK_NAME#104 |             1891 |          16389.46
 2000010                | STOCK_NAME#105 |              384 |           9297.98
 2000010                | STOCK_NAME#106 |              999 |           5227.98
 2000010                | STOCK_NAME#107 |              161 |           9753.48
 2000010                | STOCK_NAME#108 |              570 |          10918.92
 2000010                | STOCK_NAME#109 |              460 |           8521.47
 2000010                | STOCK_NAME#110 |             1088 |          20814.29
 2000010                | STOCK_NAME#111 |              854 |          35174.67
 2000010                | STOCK_NAME#112 |              325 |          10343.32
 2000010                | STOCK_NAME#114 |             1967 |          27991.93
 2000010                | STOCK_NAME#115 |              986 |          24233.03
…
```

stock_query_03

Query

```
--- QUERY #3
--- Stocks with maximum traded quantity and value in a
--- given week of the year
SELECT Day_Number_in_Calendar_Month,
       C.Stock_Name,
       SUM(Total_Traded_Quantity)   AS Total_Traded_Qty,
       SUM(Total_Turnover)          AS Total_Trade_value
FROM     StockTransaction_Fact A,
         Date_Dimension B,
         Stock_Dimension C
WHERE    A.Date_Key = B.Date_Key
         AND A.Stock_Key = C.Stock_Key
         AND B.Calendar_Week_Number_in_Year = 7
GROUP BY Day_Number_in_Calendar_Month,
         Stock_Name
ORDER BY Day_Number_in_Calendar_Month,
         Total_Traded_Qty DESC;
```

Example

```
day_number_in_calendar_month |   stock_name   | total_traded_qty | total_trade_value
-----------------------------+----------------+------------------+-------------------
                            9 | STOCK_NAME#88  |            14060 |         263374.23
                            9 | STOCK_NAME#84  |            11695 |         147378.71
                            9 | STOCK_NAME#35  |            11571 |         188538.46
                            9 | STOCK_NAME#139 |            11448 |            187385
                            9 | STOCK_NAME#225 |            11225 |         196794.57
                            9 | STOCK_NAME#247 |            11215 |         154407.54
                            9 | STOCK_NAME#37  |            11142 |         160855.92
                            9 | STOCK_NAME#237 |            11020 |         215696.29
                            9 | STOCK_NAME#70  |            10965 |         177607.71
                            9 | STOCK_NAME#114 |            10806 |         146490.98
                            9 | STOCK_NAME#49  |            10641 |         184186.92
                            9 | STOCK_NAME#272 |            10410 |         207722.41
                            9 | STOCK_NAME#61  |            10324 |         152053.87
                            9 | STOCK_NAME#18  |            10155 |         179153.75
                            9 | STOCK_NAME#113 |             9993 |          177771.6
```

stock_query_04

Query

```
--- Query 04
--- Types of traders who have a maximum turnover in a given week
SELECT TraderType,
       SUM(Total_Traded_Quantity)   AS Total_Traded_Quantity,
       SUM(Total_Deliverable_Qty)   AS Total_Deliverable_Qty,
       SUM(Total_Deliverable_Qty)
       / SUM(Total_Traded_Quantity) AS Delivery_Trade_Ratio
FROM     StockTransaction_Fact A,
         Date_Dimension B,
         Trader_Dimension C
WHERE    A.Date_Key = B.Date_Key
         AND A.Trader_Key = C.Trader_Key
         AND B.Calendar_Week_Number_in_Year = 9
GROUP BY TraderType
ORDER BY Delivery_Trade_Ratio;
```

Example

```
    tradertype     | total_traded_quantity | total_deliverable_qty | delivery_trade_ratio
-------------------+-----------------------+-----------------------+----------------------
 Retail-Investor   |               8909895 |              26430691 |                2.966
 Stock-Broker      |               9715181 |              28858933 |                 2.97
 Insurance-Company |               8740296 |              26128178 |                2.989
 Others            |              11123241 |              33559823 |                3.017
 Bank              |              10993831 |              33219510 |                3.022
(5 rows)
```

stock_query_05

Query

```
--- Query 05
--- Exchange that has a maximum turnover in a year

SELECT Calendar_Year,
       Exchange_Name,
       SUM(Total_Traded_Quantity)   AS Total_Traded_Quantity,
       SUM(Total_Turnover)          AS Total_Trade_value
FROM     StockTransaction_Fact A,
         Date_Dimension B,
         Exchange_Dimension C
WHERE    A.Date_Key = B.Date_Key
         AND A.Exchange_Key = C.Exchange_Key
GROUP BY Calendar_Year,
         Exchange_Name
ORDER BY Total_Trade_value DESC,
         Total_Traded_Quantity;
```

Example

```
calendar_year | exchange_name | total_traded_quantity | total_trade_value
---------------+---------------+-----------------------+-------------------
          2000 | SHSE          |             140598876 |     2804225676.41
          2000 | LSE           |              70490890 |     1407294711.73
          2000 | NYSE          |              69888940 |      1403007907.5
          2000 | TSE           |              70013693 |     1402097341.09
          2000 | BSE           |              69828761 |     1396841233.24
          2000 | MSE           |              69536068 |     1391241795.02
          2001 | SHSE          |             144673882 |      2901380718.8
          2001 | BSE           |              72325862 |     1450172021.31
          2001 | LSE           |              72255304 |     1446701214.62
          2001 | TSE           |              72053573 |     1444195757.99
          2001 | NYSE          |              72086902 |     1442130576.81
          2001 | MSE           |              71951644 |     1440488299.91
          2002 | SHSE          |             146419985 |     2929077855.41
          2002 | LSE           |              73391671 |     1472145875.43
          2002 | MSE           |              73491704 |     1470316584.75
          2002 | NYSE          |              73413826 |     1468898162.55
          2002 | TSE           |              73131808 |     1466021941.78
          2002 | BSE           |              72713813 |     1458532063.79
          2003 | SHSE          |             141327981 |     2833415190.98
          2003 | MSE           |              70796425 |     1414284020.43
          2003 | TSE           |              70275688 |     1412779754.15
          2003 | NYSE          |              70483411 |     1412075549.07
          2003 | LSE           |              70573600 |     1411885663.35
          2003 | BSE           |              70085099 |     1405457706.97
          2004 | SHSE          |             141762728 |     2832792072.94
          2004 | NYSE          |              70852202 |     1420409267.29
          2004 | BSE           |              70645192 |     1418096154.21
          2004 | TSE           |              70803574 |     1416000504.84
          2004 | LSE           |              70644325 |     1413213041.39
          2004 | MSE           |              70485707 |     1409461236.42
(30 rows)
```

stock_query_06

Query

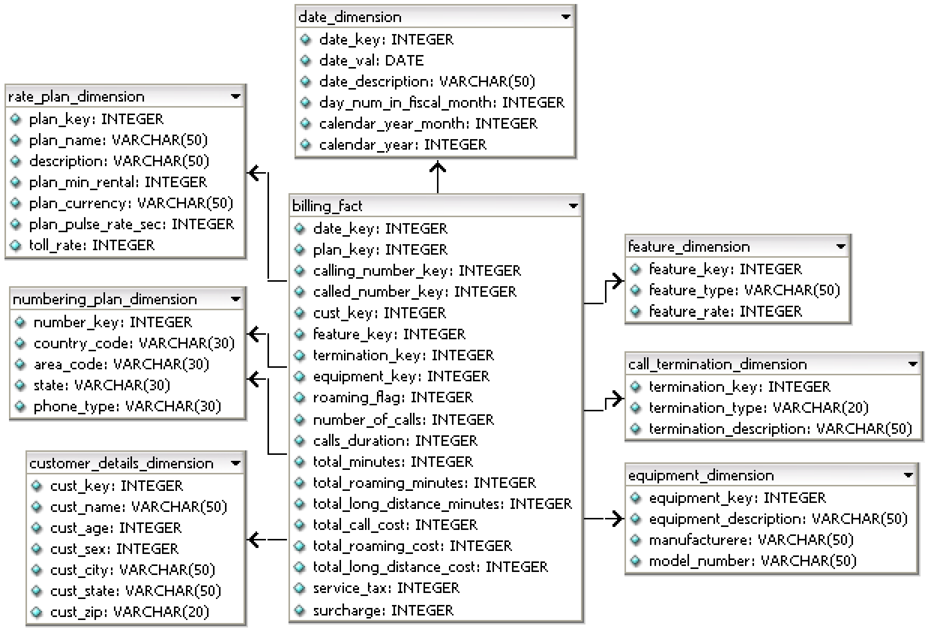
```
--- Query 06
--- Get the closing price of a set of 10 stocks for a 10-year period
--- and weekly aggregates.
SELECT Calendar_Year,
       Calendar_Year_Month,
       Calendar_Week_Number_in_Year,
       Stock_name,
       MIN(Close_Price),
       MAX(Close_Price),
       AVG(Close_Price)
FROM     StockTransaction_fact A,
         Date_Dimension B,
         Stock_Dimension C
WHERE    A.stock_key = C.stock_key
         AND A.date_key = B.date_key
         AND Calendar_Year >= 1900
         AND Calendar_Year <= 2007
GROUP BY Calendar_Year,
         Calendar_Year_Month,
         Calendar_Week_Number_in_Year,
         Stock_name
ORDER BY Stock_name,
         Calendar_Year,
         Calendar_Year_Month,
         Calendar_Week_Number_in_Year;
```

Example

```
   stock_name    | calendar_year | calendar_year_month | week_number | min_close_price |
max_close_price | avg_close_price
----------------+---------------+---------------------+-------------+-----------------+----------
-------+-----------------
 STOCK_NAME#1   |          2000 | 2000-1              |           1 |           40.14 |
69.8 |          53.37
 STOCK_NAME#1   |          2000 | 2000-1              |           2 |           40.08 |
70.53 |          53.95
 STOCK_NAME#1   |          2000 | 2000-1              |           3 |           42.08 |
70.52 |          56.54
 STOCK_NAME#1   |          2000 | 2000-1              |           4 |           40.07 |
70.35 |          55.33
 STOCK_NAME#1   |          2000 | 2000-1              |           5 |           42.75 |
67.86 |          56.61
 STOCK_NAME#1   |          2000 | 2000-2              |           5 |            40.2 |
70.65 |          53.92
 STOCK_NAME#1   |          2000 | 2000-2              |           6 |           40.13 |
69.26 |          55.86
 STOCK_NAME#1   |          2000 | 2000-2              |           7 |           40.25 |
71 |          55.58
 STOCK_NAME#1   |          2000 | 2000-2              |           8 |           40.52 |
70.88 |          56.67
 STOCK_NAME#1   |          2000 | 2000-3              |           9 |           40.23 |
70.71 |           54.71
 STOCK_NAME#1   |          2000 | 2000-2              |           9 |           40.85 |
69.86 |           55.23
 STOCK_NAME#1   |          2000 | 2000-3              |          10 |           40.28 |
70.73 |           56.48
 STOCK_NAME#1   |          2000 | 2000-3              |          11 |           40.06 |
70.14 |           55.82
 STOCK_NAME#1   |          2000 | 2000-3              |          12 |           40.26 |
70.91 |           57.74
…
```

Telecom Example Database

The Telecom schema is a simple star schema that represents a summary of the calls made by the customers of a fictional cell phone service provider. Each table is described in a separate section.



date_dimension
- date_key: INTEGER
- date_val: DATE
- date_description: VARCHAR(50)
- day_num_in_fiscal_month: INTEGER
- calendar_year_month: INTEGER
- calendar_year: INTEGER

rate_plan_dimension
- plan_key: INTEGER
- plan_name: VARCHAR(50)
- description: VARCHAR(50)
- plan_min_rental: INTEGER
- plan_currency: VARCHAR(50)
- plan_pulse_rate_sec: INTEGER
- toll_rate: INTEGER

billing_fact
- date_key: INTEGER
- plan_key: INTEGER
- calling_number_key: INTEGER
- called_number_key: INTEGER
- cust_key: INTEGER
- feature_key: INTEGER
- termination_key: INTEGER
- equipment_key: INTEGER
- roaming_flag: INTEGER
- number_of_calls: INTEGER
- calls_duration: INTEGER
- total_minutes: INTEGER
- total_roaming_minutes: INTEGER
- total_long_distance_minutes: INTEGER
- total_call_cost: INTEGER
- total_roaming_cost: INTEGER
- total_long_distance_cost: INTEGER
- service_tax: INTEGER
- surcharge: INTEGER

feature_dimension
- feature_key: INTEGER
- feature_type: VARCHAR(50)
- feature_rate: INTEGER

numbering_plan_dimension
- number_key: INTEGER
- country_code: VARCHAR(30)
- area_code: VARCHAR(30)
- state: VARCHAR(30)
- phone_type: VARCHAR(30)

call_termination_dimension
- termination_key: INTEGER
- termination_type: VARCHAR(20)
- termination_description: VARCHAR(50)

customer_details_dimension
- cust_key: INTEGER
- cust_name: VARCHAR(50)
- cust_age: INTEGER
- cust_sex: INTEGER
- cust_city: VARCHAR(50)
- cust_state: VARCHAR(50)
- cust_zip: VARCHAR(20)

equipment_dimension
- equipment_key: INTEGER
- equipment_description: VARCHAR(50)
- manufacturere: VARCHAR(50)
- model_number: VARCHAR(50)

Table Name	Default Number of Rows
Billing_Fact (on page 59)	5000000
Customer_Details_Dimension (on page 60)	50000
Rate_Plan_Dimension (on page 61)	500
Numbering_Plan_Dimension (on page 61)	500
Equipment_Dimension (on page 61)	200
Feature_Dimension (on page 61)	20
Call_Termination_Dimension (on page 59)	20

Billing_Fact

Each tuple in the fact table represents a summary of the CDR records generated at the switch for each customer.

Field Name	Data Type	Description/Example
Date_Key	INTEGER	Call Date. Foreign Key, references Date table key
Plan_Key	INTEGER	Foreign Key, references Rate_Plan table key
Calling_Number_key	INTEGER	Calling party number. Foreign Key, references Number table key
Called_Number_Key	INTEGER	Called party location. Foreign Key, references Number table key
Cust_Key	INTEGER	Calling Party customer id who is billed for the call and services. Foreign Key, references Customer_Details table key
Feature_key	INTEGER	Foreign Key, references Feature table key
Termination_key	INTEGER	Call Termination Type; for example, normal or abnormal
Equipment_Key	INTEGER	Type of Equipment
Roaming_Flag	BOOL	Whether this call is made/received while roaming
Call_Duration	TIMESTAMP	Duration of the call
Number_of_Calls	INTEGER	Total number of calls made during the day
Total_Minutes	INTEGER	Total number of minutes of calls made that day
Total_Roaming_Minutes	INTEGER	Total number of roaming minutes used
Total_Long_Distance_Minutes	INTEGER	Total number of long distance calls made.
Total_Call_Cost	FLOAT	Total cost of all the call
Total_Roaming_Cost	FLOAT	Total roaming charges
Total_Long_Distance_Cost	FLOAT	Total Long Distance charges
Service_tax	FLOAT	Service tax
Surcharge	FLOAT	Surcharge

Call_Termination_Dimension

This table describes all possible reasons for which a call can be terminated.

Field Name	Data Type	Description/Example
Termination_key	INTEGER	Primary Key
Termination_type	VARCHAR	ITAU Termination Type (normal, service failure, equipment failure, etc.)
Termination_Description	VARCHAR	ITAU Termination Description

Customer_Details_Dimension

This table describes the customers of the service provider.

Field Name	Data Type	Description/Example
Cust_Key	INTEGER	Primary Key
Cust_Name	VARCHAR	Customer/Subscriber Name
Cust_Age	INTEGER	Age of the customer
Cust_Sex	CHAR	Male/Female (M/F)
Cust_City	VARCHAR	City of the customer
Cust_State	VARCHAR	State of the customer
Cust_Zip	VARCHAR	Zip/postal code of the customer

Date_Dimension

This table contains data for dates.

Field Name	Data Type	Description/Example
Date_Key	INTEGER	Primary Key
Date_val	DATE	Date in ‘mm/dd/yyyy’ format
Date_Description	VARCHAR	Description of the date; for example, January 1, 2000
Calendar_Year	INTEGER	Calendar year of the date; for example, 2001
Calendar_Year_Month	INTEGER	Calendar month of the date (1-12); for example, 9 for September
Day_Num_in_Fiscal_Month	INTEGER	The day number in the month (1-31); for example, 21 for 21st of any month.

Equipment_Dimension

This table describes type of equipment (handsets) used by customers of the service provider.

Field Name	Data Type	Description/Example
Equipment_Key	INTEGER	Primary Key
Equipment type	VARCHAR	Landline/mobile/WLL/blackberry/wireless card
Manufacturer	VARCHAR	Nokia/Sony-Ericsson/Motorola
Model Number	INTEGER	Phone model number

Feature_Dimension

This table describes every feature offered by the service provider.

Field Name	Data Type	Description/Example
Feature_Key	INTEGER	Primary Key
Feature_type	VARCHAR	SMS/MMS/Call Forwarding/Call Waiting/
Feature_rate	VARCHAR	Feature cost per invocation

Numbering_Plan_Dimension

This table describes the types of numbering plans. This is used to distinguish between originating and terminating destination calls.

Field Name	Data Type	Description/Example
Number_Key	INTEGER	Primary Key
Country_Code	VARCHAR	Country code part of the phone number.
Area_Code	VARCHAR	3 Digit Area Code of the Phone
State	VARCHAR	State Code of the Numbering Plan.
Phone_Type	VARCHAR	Type of the phone fixed/GSM/CDMA

Rate_Plan_Dimension

This table describes all the rate plans offered by the service provider.

Field Name	Data Type	Description/Example
Plan_Key	INTEGER	Primary Key
Plan_Name	VARCHAR	Common/Business Name of the rate plan say ‘Freedom25’
Plan_Description	VARCHAR	Description of the rate plan
Plan_Min_Rental	FLOAT	Minimum monthly rental for this rate plan say 24.99 (USD)
Plan_Currency	VARCHAR	Plan Currency (USD)
Plan_Pulse_Rate_sec	INTEGER	Pulse rate available in the plan say 30 sec pulse or 60 sec pulse
Toll_rate	FLOAT	Call charges for the plan

telecom_query_01.sql

Query

```
-- Best month of the year in terms of
-- minutes of usage for each year of operation.

SELECT   Calendar_Year,
         Calendar_Year_Month,
         SUM(Total_Minutes) AS Total_Minutes
FROM     Billing_Fact,
         Date_Dimension
WHERE    Billing_Fact.Date_Key = Date_Dimension.Date_Key
GROUP BY Calendar_Year,Calendar_Year_Month
ORDER BY Calendar_Year,
         Calendar_Year_Month;
```

Example

```
Calendar_Year | Calendar_Year_Month | Total_Minutes
--------------+---------------------+---------------
         2000 |                   1 |          1451
         2000 |                   2 |          1616
         2000 |                   3 |          1397
         2000 |                   4 |          1334
         2000 |                   5 |          1076
(17 rows)
```

telecom_query_02.sql

Query

```
-- Best rate plan in use

SELECT   Calendar_Year,
         Calendar_Year_Month,
         Plan_Name,
         SUM(Number_Of_Calls) AS Calls,
         SUM(Total_Minutes) AS Total_Minutes
FROM     Billing_Fact Bill_Fact,
         Date_Dimension Date_Dim,
         Rate_Plan_Dimension Rate_Dim
WHERE    Bill_Fact.Date_Key = Date_Dim.Date_Key
         AND Bill_Fact.Plan_Key = Rate_Dim.Plan_Key
GROUP BY Calendar_Year,Calendar_Year_Month,Plan_Name
HAVING   SUM(Number_Of_Calls) >= 10
ORDER BY Calls;
```

Example

```
 Calendar_Year | Calendar_Year_Month |     Plan_Name      | Calls | Total_Minutes
---------------+---------------------+--------------------+-------+---------------
          2000 |                  12 | Freedom_40         |    10 |            18
          2000 |                   9 | Youth_45           |    10 |            48
          2000 |                   2 | Freedom_30         |    10 |            49
          2000 |                   6 | Flexi_40           |    10 |            35
          2000 |                   1 | Flexi_30           |    10 |            36
          2000 |                   9 | Youth_30           |    10 |            81
          2000 |                   6 | Youth_25           |    10 |            55
          2000 |                  10 | Executive_40       |    10 |            42
(319 rows)
```

telecom_query_03.sql

Query

```
-- Customer using the most roaming minutes in 2000

SELECT   Cust_Name,
         Calendar_Year,
         SUM(Total_Roaming_minutes) AS TOTAL_ROAMING
FROM     Billing_Fact Bill_Fact,
         Date_Dimension Date_Dim,
         Customer_Details_Dimension Cust_Dim
WHERE    Bill_Fact.Cust_Key = Cust_Dim.Cust_Key
         AND Bill_Fact.Date_Key = Date_Dim.Date_Key
         AND Date_Dim.Calendar_Year = 2000
         AND Bill_Fact.Roaming_Flag = 1
GROUP BY Cust_Name,Calendar_Year
ORDER BY Cust_Name,
         TOTAL_ROAMING DESC;
```

Example

```
  Cust_Name   | Calendar_Year | Total_Roaming
--------------+---------------+---------------
 (null)       |          2000 |           361
 Abigail      |          2000 |           323
 Andrew       |          2000 |           216
 Anthony      |          2000 |           384
 AshleyJack   |          2000 |           378
 Ava          |          2000 |           243
(29 rows)
```

telecom_query_04.sql

Query

```
-- Total service tax and surcharge paid to government in 2000

SELECT   Calendar_Year,
         Calendar_Year_Month,
         SUM(Service_Tax) AS SERVICE_TAX,
         SUM(SURCHARGE) AS SURCHARGE
FROM     Billing_Fact Bill_Fact,
         Date_Dimension Date_Dim
WHERE    Bill_Fact.Date_Key = Date_Dim.Date_Key
         AND Date_Dim.Calendar_Year = 2000
GROUP BY Calendar_Year,Calendar_Year_Month
ORDER BY Calendar_Year,
         Calendar_Year_Month DESC;
```

Example

```
 Calendar_Year | Calendar_Year_Month | Service_Tax | Surcharge
---------------+---------------------+-------------+-----------
          2000 |                  12 |      67.405 |    6.7405
          2000 |                  11 |      45.615 |    4.5615
          2000 |                  10 |      49.315 |    4.9315
          2000 |                   9 |      63.495 |    6.3495
          2000 |                   8 |       62.53 |     6.253
(12 rows)
```

telecom_query_05.sql

Query

```
-- Total number of calls with abnormal termination code

SELECT   Calendar_Year,
         Termination_Description,
         SUM(Number_Of_Calls) AS CALL_COUNT
FROM     Billing_Fact Bill_Fact,
         Date_Dimension Date_Dim,
         Call_Termination_Dimension Term_Dim
WHERE    Bill_Fact.Date_Key = Date_Dim.Date_Key
         AND Bill_Fact.Termination_Key = Term_Dim.Termination_Key
         AND Term_Dim.Termination_Type = 'Abnormal'
GROUP BY Calendar_Year,Termination_Description
ORDER BY Calendar_Year,
         CALL_COUNT;
```

Example

```
 Calendar_Year |  Termination_Description  | Call_Count
---------------+---------------------------+------------
          2000 | Abnormal Call Termination |       2010
          2001 | Abnormal Call Termination |        873
```

```
(2 rows)
```

telecom_query_06.sql

Query

```
-- Show average phone usage by customer age group
-- (5-year intervals) for a given month
SELECT calendar_year_month,
       (cust_age - MOD(cust_age, 5)) AS age_group,
       count(DISTINCT A.cust_key) AS num_customers,
       ROUND(AVG(number_of_calls)) AS avg_num_calls,
       ROUND(AVG(calls_duration)) AS avg_call_duration,
       ROUND(AVG(total_minutes)) AS avg_total_mins
FROM   Billing_Fact A,
       Date_Dimension B,
       Customer_Details_Dimension C
WHERE  A.date_key = B.date_key
       AND A.cust_key = C.cust_key
       AND calendar_year_month = '2004-8'
GROUP BY calendar_year_month,
         (cust_age - MOD(cust_age, 5))
ORDER BY (cust_age - MOD(cust_age, 5));
```

Example

```
calendar_year_month | age_group | num_customers | avg_num_calls | avg_call_duration | avg_total_mins
---------------------+-----------+---------------+---------------+-------------------+-----------
-----
 2004-8              |        10 |          2871 |             8 |                25 |
24
 2004-8              |        15 |          2903 |             7 |                25 |
24
 2004-8              |        20 |          2790 |             7 |                24 |
25
 2004-8              |        25 |          2787 |             8 |                25 |
25
 2004-8              |        30 |          2768 |             7 |                24 |
25
 2004-8              |        35 |          2843 |             7 |                25 |
24
 2004-8              |        40 |          2873 |             8 |                25 |
25
 2004-8              |        45 |          2856 |             8 |                24 |
24
 2004-8              |        50 |          2926 |             8 |                24 |
25
 2004-8              |        55 |          2809 |             8 |                25 |
24
 2004-8              |        60 |          2836 |             7 |                25 |
24
 2004-8              |        65 |          2855 |             7 |                24 |
24
 2004-8              |        70 |          2772 |             7 |                24 |
25
 2004-8              |        75 |          2798 |             8 |                24 |
25
 2004-8              |        80 |           572 |             7 |                25 |
25
(15 rows)
```

VMart Example Database

The VMart Example Database is based on a fictional department store chain that has an online store front in addition to traditional brick and mortar stores. This database contains the following schemas:

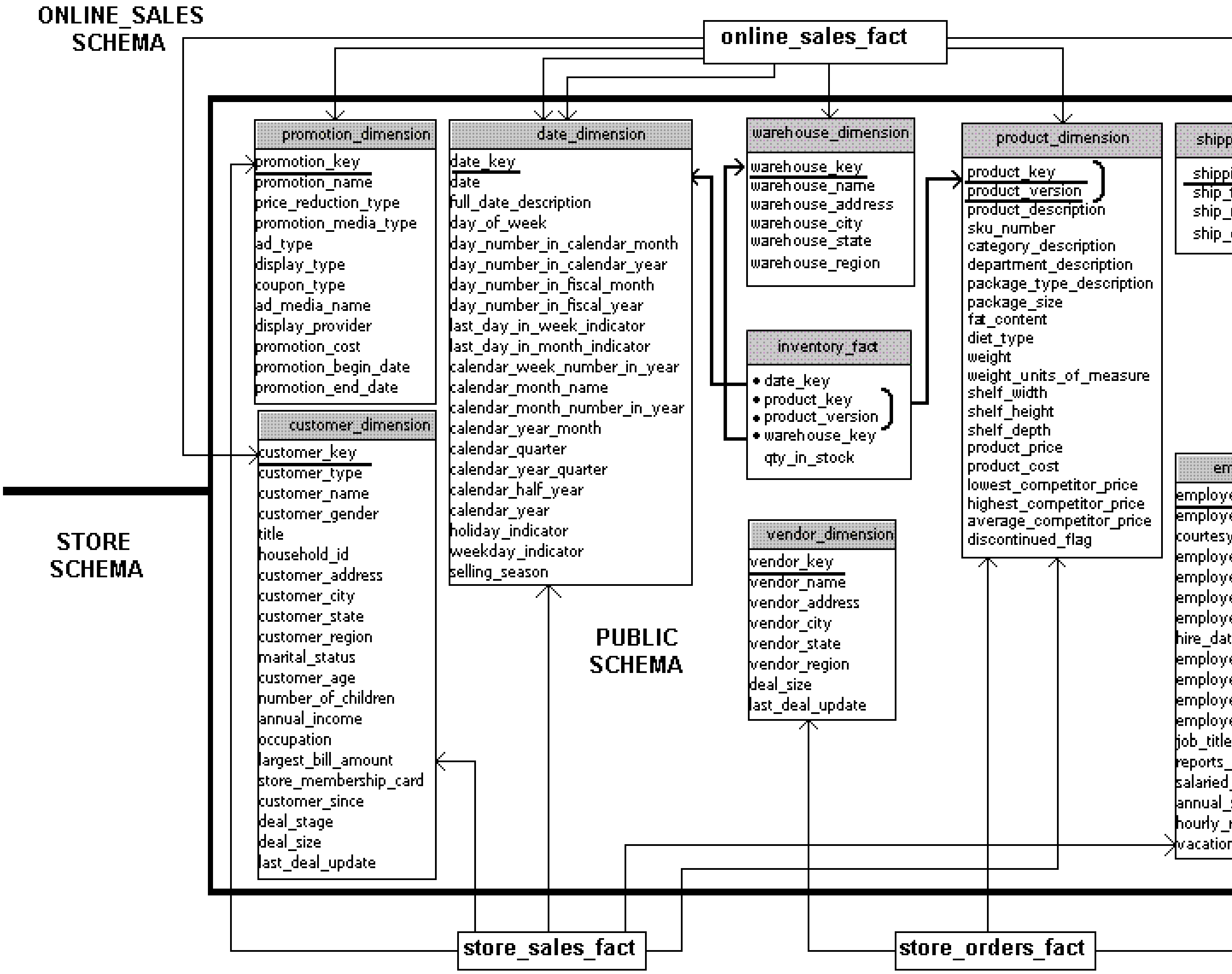
- ***Public Schema*** (page 68)
- ***Store Schema*** (page 74)
- ***Online_Sales Schema*** (page 77)

Each schema is described in a separate section.

Note: The example VMart queries in subsequent sections are for illustrative purposes only. Your results could differ slightly from those listed in this guide.

Public Schema

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

The product_dimension 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

The promotion_dimension 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 the 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

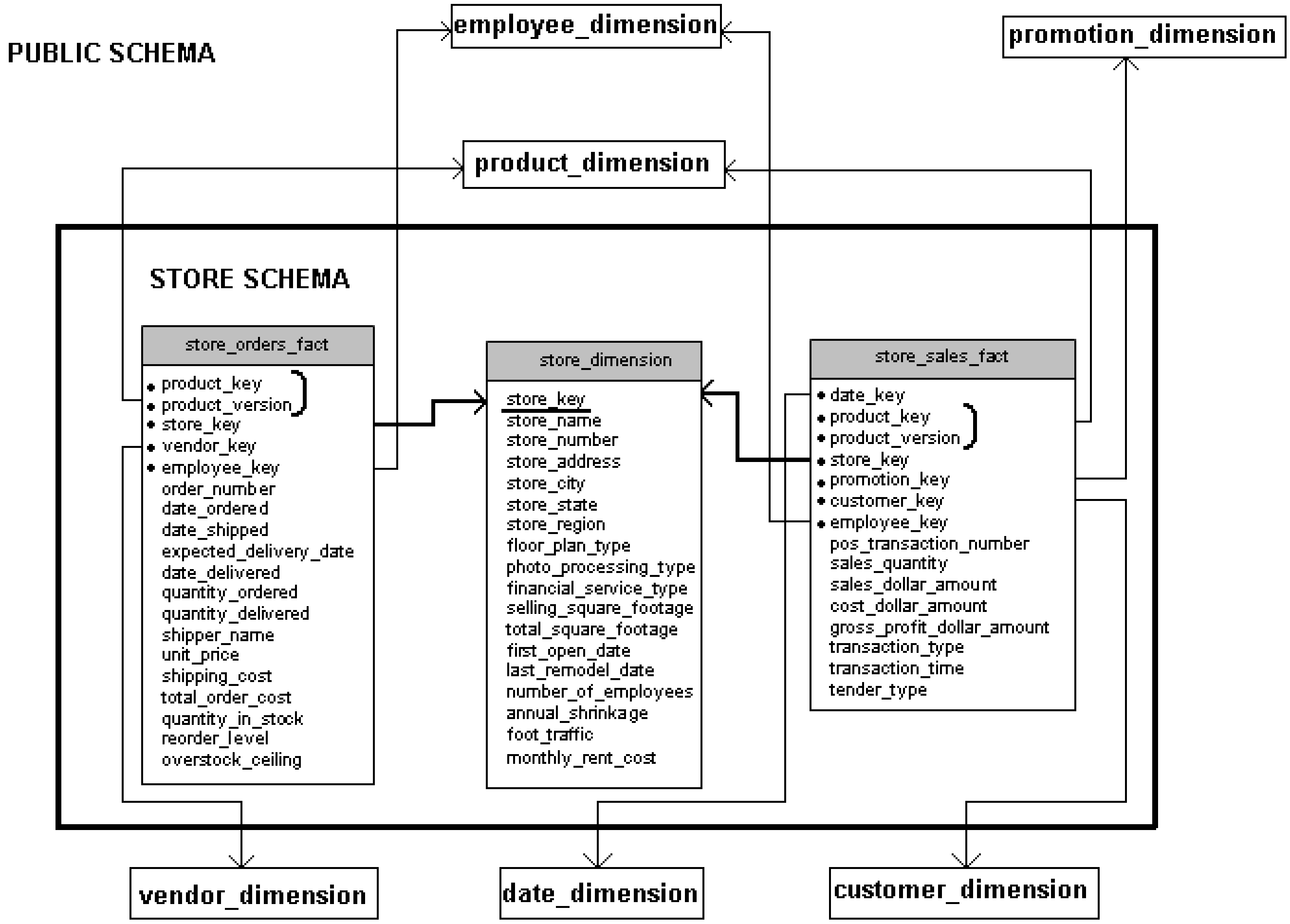
warehouse_dimension

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

The Store schema is a snowflake schema that contains information about the retail chain's brick 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

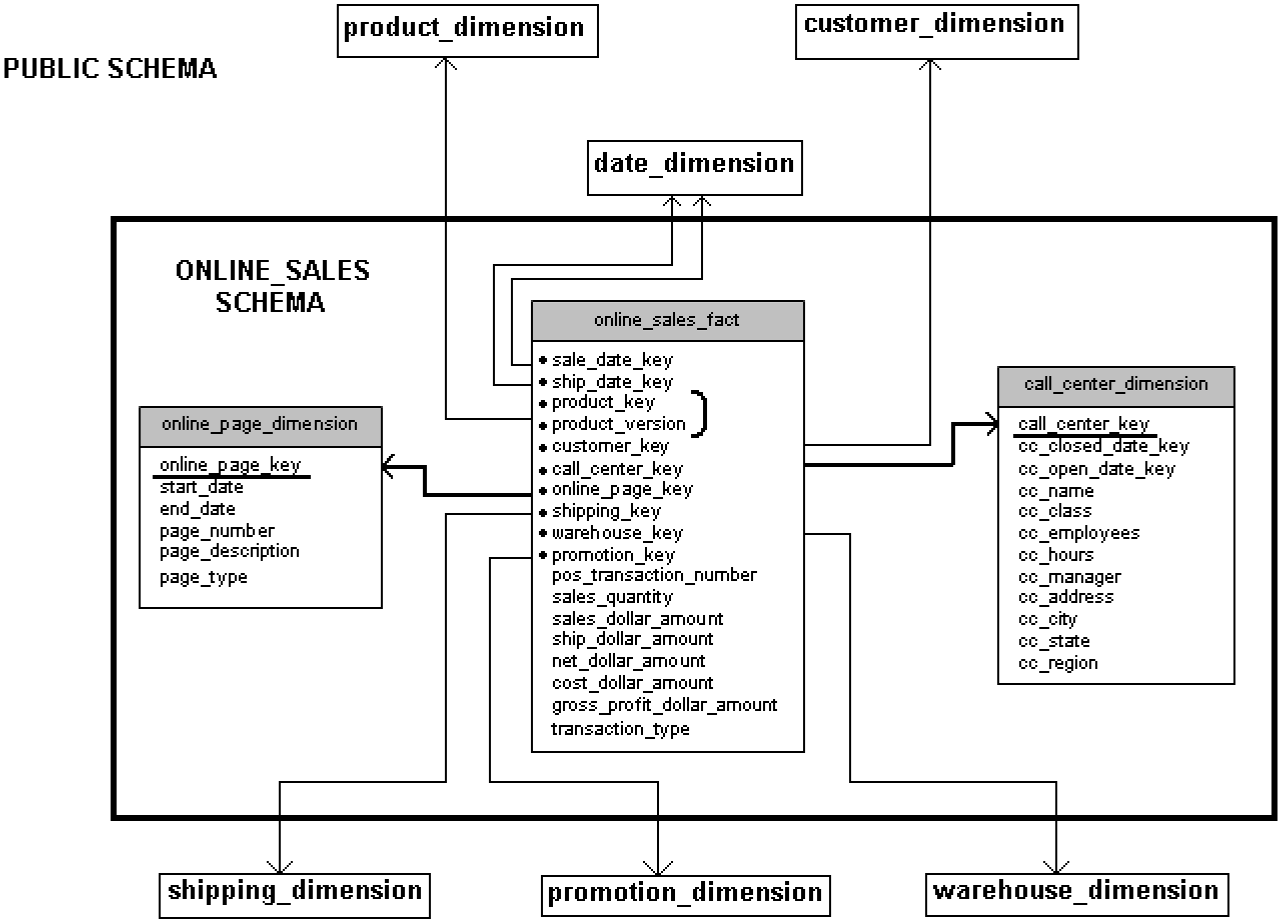
store_dimension

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

The Online_Sales schema is a snowflake schema. The following graphic illustrates the Online_Sales schema and its relationship with tables in the Public schema.



online_sales_fact

The online_sales_fact 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

The call_center_dimension 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_name	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

The online_page_dimension 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

vmart_query_01.sql

Query

```
-- 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
```

Example

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

vmart_query_02.sql

Query

```
-- 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';
```

Example

```
 order_number | date_ordered
--------------+--------------
         1584 | 2003-01-05
        39396 | 2003-02-05
        83738 | 2003-01-04
         8898 | 2003-02-05
        69712 | 2003-01-06
        74866 | 2003-01-03
        75397 | 2003-02-06
        60069 | 2003-01-10
        85854 | 2003-01-03
        21982 | 2003-02-03
        47766 | 2003-02-07
        31284 | 2003-02-03
        28005 | 2003-01-09
        79963 | 2003-02-01
        19515 | 2003-02-05
(15 rows)
```

vmart_query_03.sql

Query

```
-- 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);
```

Example

```
   customer_name    | annual_income
--------------------+---------------
 Meghan U. Miller   |        999960
 Michael T. Jackson |        999981
(2 rows)
```

vmart_query_04.sql

Query

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

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;
```

Example

```
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
(15 rows)
```

vmart_query_05.sql

Query

```
-- 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';
```

Example

```
store_key  | order_number | date_ordered
-----------+--------------+--------------
       213 |       148816 | 2003-01-02
       111 |       184148 | 2003-01-02
        89 |       279732 | 2003-01-02
       115 |         3677 | 2003-01-02
       212 |       117057 | 2003-01-02
        65 |       198323 | 2003-01-02
       238 |       246942 | 2003-01-02
       140 |       257554 | 2003-01-02
        43 |        79699 | 2003-01-02
       219 |       240925 | 2003-01-02
       249 |         4789 | 2003-01-02
        12 |       234175 | 2003-01-02
       119 |       176211 | 2003-01-02
       107 |       249378 | 2003-01-02
       228 |       251959 | 2003-01-02
(15 rows)
```

vmart_query_06.sql

Query

```
-- 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';
```

Example

```
 store_key | order_number | date_ordered
-----------+--------------+--------------
```

```
       168 |        51386 | 2004-01-04
        88 |        73316 | 2004-01-04
       241 |        68520 | 2004-01-04
(3 rows)
```

vmart_query_07.sql

Query

```
-- 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);
```

Example

```
         product_description          | sku_number | department_description
-------------------------------------+------------+------------------------
 Brand #7979 cheddar cheese           | SKU-#7979  | Dairy
 Brand #2197 sushi                    | SKU-#2197  | Seafood
 Brand #28902 strawberries            | SKU-#28902 | Produce
 Brand #54595 sliced turkey           | SKU-#54595 | Meat
 Brand #26127 chocolate chip cookies | SKU-#26127 | Bakery
 Brand #32608 chocolate ice cream     | SKU-#32608 | Frozen Goods
 Brand #27213 shrimp                  | SKU-#27213 | Seafood
 Brand #12533 canned green beans      | SKU-#12533 | Canned Goods
 Brand #3957 canned tuna              | SKU-#3957  | Canned Goods
 Brand #22103 cod                     | SKU-#22103 | Seafood
(10 rows)
```

vmart_query_08.sql

Query

```
-- 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';
```

Example

```
      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

Query

```
-- 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;
```

Example

```
 sales_quantity | sales_dollar_amount | transaction_type |      cc_name
----------------+---------------------+------------------+-------------------
              7 |                 513 | purchase         | Southeastern
              3 |                 439 | purchase         | Southwest
             10 |                 425 | purchase         | North Midwest
              5 |                 364 | purchase         | North Midwest
              7 |                 320 | purchase         | Pacific Northwest
              2 |                 314 | purchase         | Pacific Northwest
              9 |                 299 | purchase         | California
              9 |                 265 | purchase         | Central Midwest
              9 |                 247 | purchase         | Southwest
              6 |                 221 | purchase         | Central Midwest
              1 |                 198 | purchase         | Central Midwest
              5 |                 177 | purchase         | Central Midwest
              7 |                 131 | purchase         | Southwest
             10 |                 110 | purchase         | North Midwest
              2 |                -329 | return           | Other
(15 rows)
```

Installing the Example Database

Vertica provides a one-step installation script that lets you create an example database and start using it immediately. The scripts are located in `/opt/vertica/sbin` and are called:

- `install_example` — Creates a database on the default port (5433), generates data, creates the schema and a default superprojection, and loads the data.
- `delete_example` — Drops the database

Note: For a more advanced but equally-simple example using the Vertica databases, see the ***Tutorial*** (page 87) in the Getting Started Guide.

Installing the Example Database

1 In a terminal window, log in as the DBA user:

```
# su dbadmin
```

2 Change to the `/example` directory and run the install script:

```
$ /opt/vertica/sbin/install_example <example_name>
```

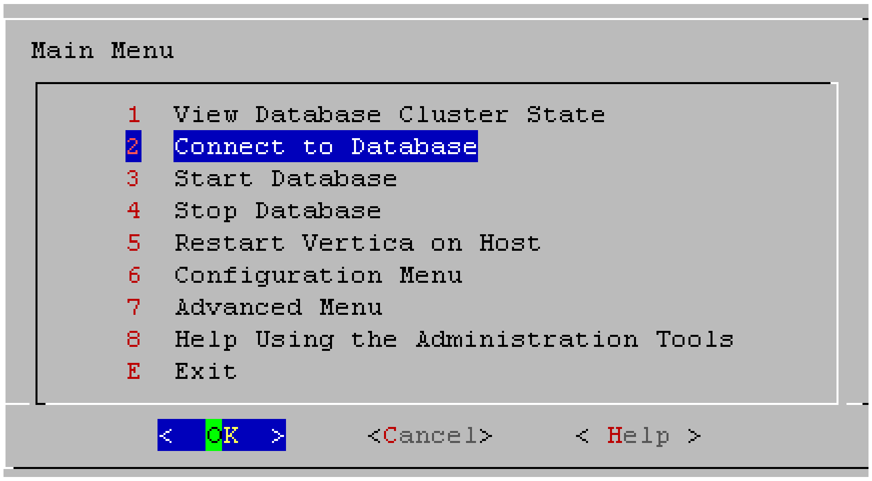
where <*example_name*> is one of the following: ClickStream, CreditHistory, Retail, Stock, TickStore, Telecom, VMart.

3 Connect to the database:

```
$ /opt/vertica/bin/vsql
```

Alternatively connect to the database using the Administration Tools and select **Connect to Database** from the Main Menu:

```
$ admintools
```



4 Run a simple query. For example, to count all the records in the `store_sales_fact` table:

```
=> SELECT COUNT(1) FROM store.store_sales_fact;
```

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

Example Database Scripts

Each of the example directories contains query script files that you can use. While you can create your own queries, Vertica provides scripts to get you started quickly. All SQL scripts used by the `install_example` installation script are available for review in the following folders. You can use the scripts as templates for your own applications.

- `/opt/vertica/examples/ClickStream_Schema`
- `/opt/vertica/examples/CreditHistory_Schema`
- `/opt/vertica/examples/Retail_Schema`
- `/opt/vertica/examples/Stock_Schema`
- `/opt/vertica/examples/TickStore_Schema`
- `/opt/vertica/examples/Telecom_Schema`
- `/opt/vertica/examples/VMart_Schema`

The following table describes the scripts available, where *{identifier}* is the name of the example database:

Script Name	Description
{identifier}_count_data.sql	Counts rows of all example database tables
{identifier}_define_schema.sql	Defines the schema for each table
{identifier}_gen	Is the sample data generator
{identifier}_load_data.sql	Loads data to the corresponding tables using COPY DIRECT
{identifier}_queries.sql	Contains all sample queries
{identifier}_schema_drop.sql	Drops all example database tables
{identifier}_query_##.sql	Are the individual queries; for example query #1 through “n”

Note: The number of example databases you create is limited only by the disk space available on your system.

Deleting the One-Step Example Database

To remove an example database:

1. Log in as the DBA user; for example:

   ```
   # su dbadmin
   ```

2. Run the `delete_example` script:

   ```
   $ /opt/vertica/sbin/delete_example <example_name>
   ```

   where <*example_name*> is the name of the example database you provided to the install script.

Tutorial: Setting up an Example Database

Prerequisites

Before you proceed, Vertica must be installed on one host or a cluster of hosts, as described in the Installation Guide. Vertica recommends a minimum of three hosts in the cluster.

Audience

This tutorial targets anyone who wants to learn how to create and run a Vertica database. No special database knowledge is required at this point, though a rudimentary knowledge of basic SQL commands could be useful when you begin to run queries.

Objectives

You'll follow the simple steps below to create a fully-functioning, comprehensive design using one of the schemas described in ***Example Databases*** (page 11).

1. ***Set up the example environment*** (page 88)
2. ***Create the example database*** (page 89)
3. ***Define the database schema*** (page 91)
4. ***Load the data*** (page 92)
5. ***Create a comprehensive design*** (page 93)
6. ***Test the design*** (page 101)
7. ***(Optional) Generate custom data files*** (page 103)

It's that easy! The whole process takes about 15 minutes, and when you are finished, you can proceed directly to ***Running Simple Queries*** (page 105).

Notes

- Although the ***VMart database*** (page 67) is used throughout this tutorial, the steps are the same for all of the ***example databases*** (page 11). If you choose a different database, replace VMart with Clickstream, Credit History, Retail Sales, Stock Exchange, or Telecom in each example provided.
- This tutorial uses a Vertica-provided query, but you can follow the same set of procedures later, when you create your own design and use your own query file.
- If, in the future, you have a query that you want to optimize, you can create an enhanced design with additional projections to be tuned specifically for the query you provide. See Creating a Query-specific Design Using the Database Designer in the Administrator's Guide.
- For additional information about managing your designs, see Creating a Physical Design in the Administrator's Guide.

Step 1: Set Up the Example Environment

In this procedure, you set up the example database environment.

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

2 Choose the ***example database*** (page 11) that you want to use.

Note: All procedures in this tutorial use the ***Vmart example database*** (page 67).

3 Log in to a terminal using the database administrator account that was created during product installation.

The default account name is `dbadmin`.

4 Create a directory for the example files on the Administration Host:

```
$ mkdir examples
```

Do not use the default data directory /`home/dbadmin`.

5 Copy the files to the sample directory.

If you installed the product rpm on a database server, the example databases are located in `/opt/vertica/examples` on the host.

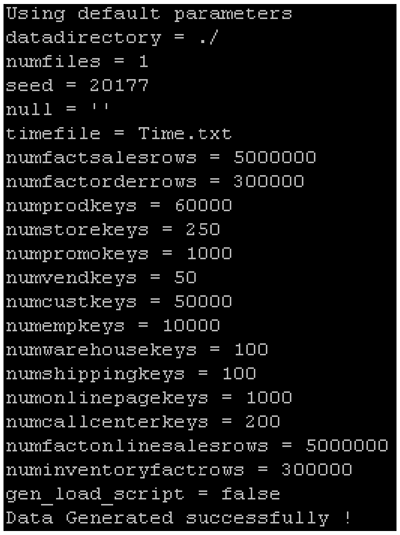
6 Set your current directory to the example database directory your created:

```
$ cd examples
```

Note: Do not change directories while following this tutorial. Some of the steps depend on being set to a specific directory.

7 Run the sample data generator program:

```
$ ./vmart_gen
Let the program run with the default parameters, which you can review
   in the README file.
```



```
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 !
```

If the `vmart_gen` executable does not work correctly, recompile it and run the sample data generator script (`./vmart_gen`) again.

This example uses the GNU C++ compiler, which is a ***free download*** (***http://gcc.gnu.org/***). You can use any other C++ compiler.

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

Tip: If you are using VMware, the fact table load could fail. Specify a smaller fact table size, such as 1000000 (1M) rows, as described in ***Generating Custom Data Files*** (page 103). The maximum size of a bulk load depends on the system resources and cannot be determined accurately.

Step 2: Create the Example Database

In this procedure, you create the example database.

1 Run the Administration Tools.

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

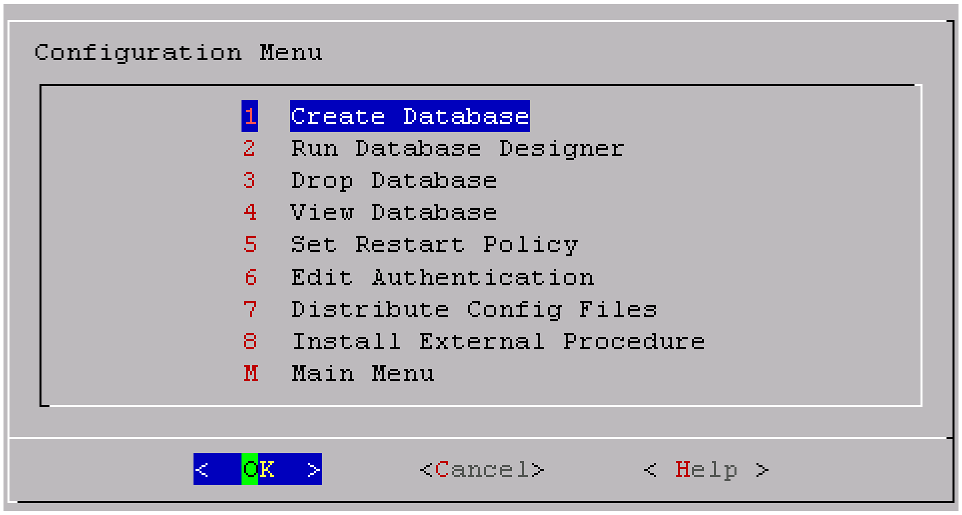
Note: See the ***Administration Tools Keystrokes*** (see "***Using the Graphical User Interface***" on page 107) for a quick reference. If you are using a remote terminal application, such as PuTTY or a Cygwin bash shell, see ***Notes for Remote Terminal Users*** (page 108).

2 Accept the license agreement (once only).

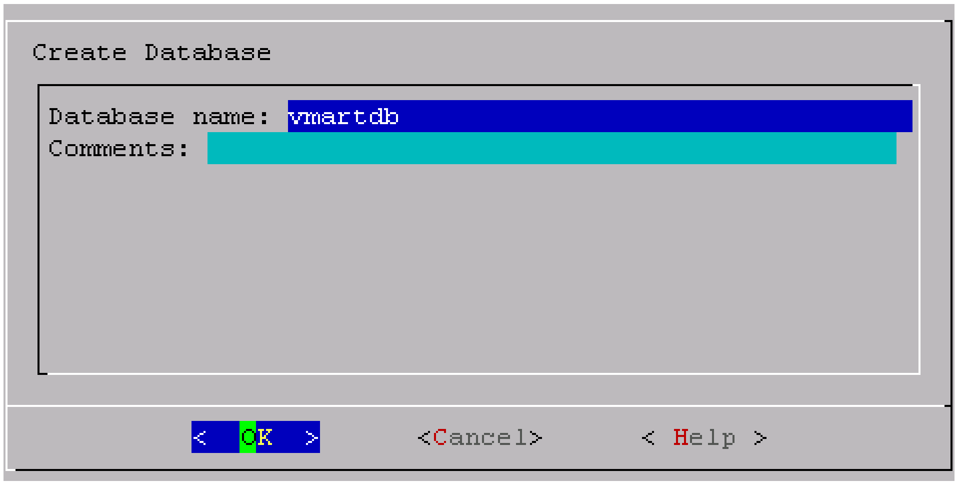
3 Specify the location of your license key file (once only).

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

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



6 Name the database **vmartdb** and click **OK**.



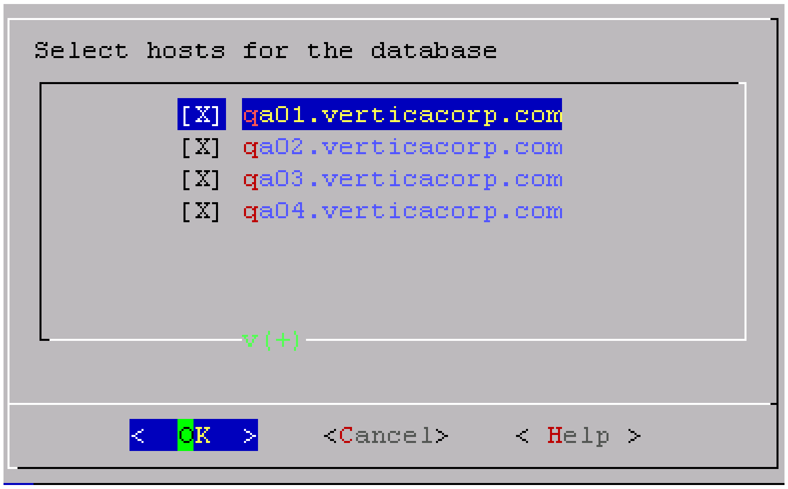
Create Database

Database name: vmartdb
Comments:

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

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

8 Select the hosts you want to include in the database cluster and click **OK**. This example creates the `vmartdb` database on a 4-host cluster.



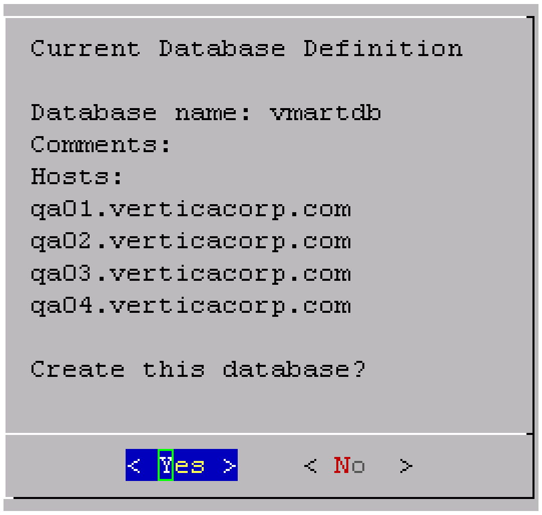
Select hosts for the database

[X] qa01.verticacorp.com
[X] qa02.verticacorp.com
[X] qa03.verticacorp.com
[X] qa04.verticacorp.com

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

- 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.

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



```
Current Database Definition

Database name: vmartdb
Comments:
Hosts:
qa01.verticacorp.com
qa02.verticacorp.com
qa03.verticacorp.com
qa04.verticacorp.com

Create this database?
```

During database creation, 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.

11 Click **OK** to close the message.

Step 3: Define the Database Schema

Now that you have created a database, define the schema.

1 On the Administration Tools Configuration Menu, click **Main Menu** and click **OK**.

2 Click **Connect to Database** and click **OK**.

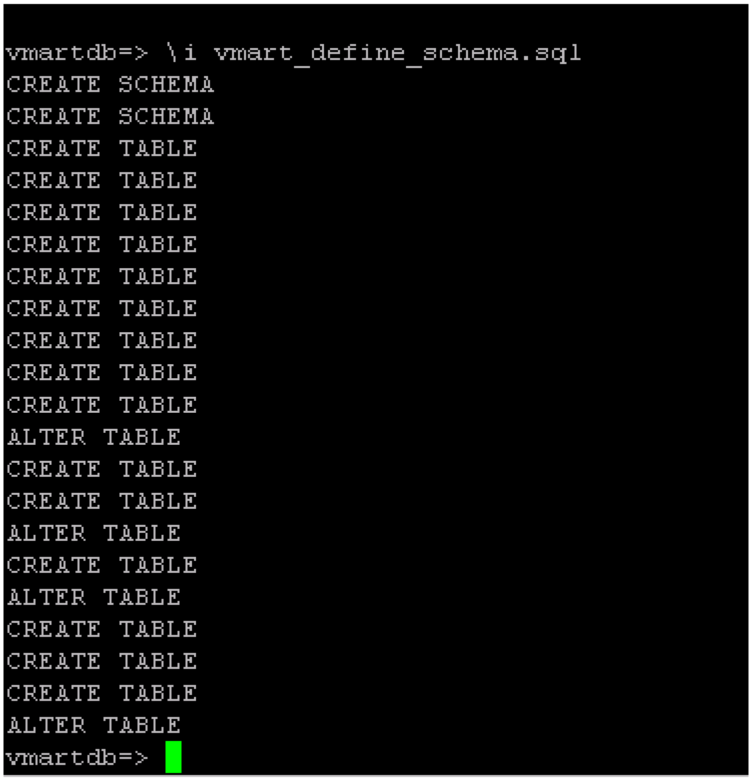
You'll see the following prompt:

```
Welcome to the vsql, Vertica_Database v4.1.x 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
vmartdb=>
```

3 To create the logical schema, run 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:



```
vmartdb=> \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
vmartdb=>
```

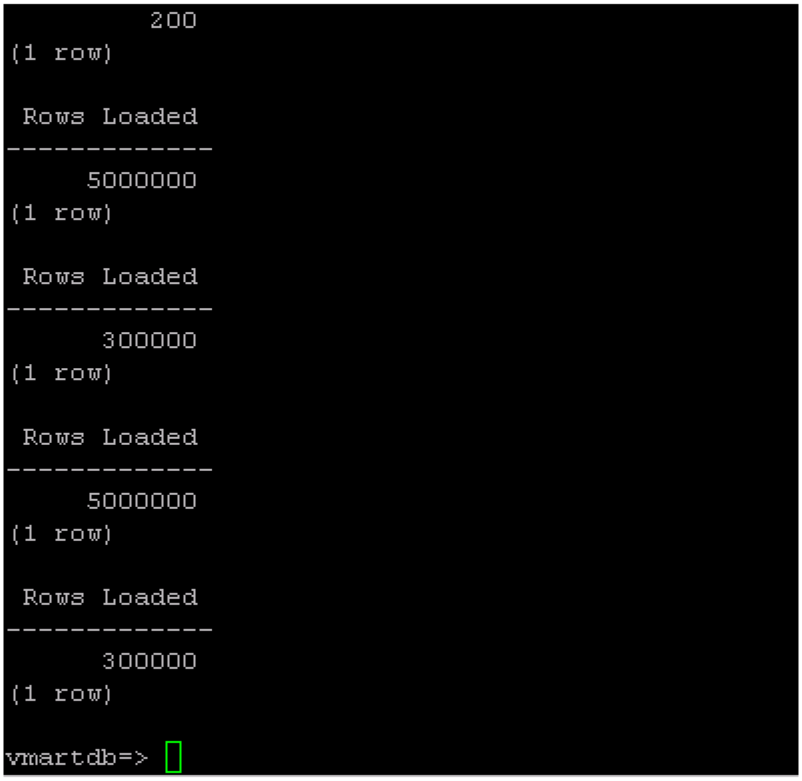
The `vmart_define_schema.sql` file creates the tables and referential integrity constraints that make up the logical schema.

Step 4: Load the Data

In this brief step, you'll load data into the schema you created in the previous step. Vertica automatically creates a superprojection for each table into which data is loaded.

1 Load data into the VMart database using the `vmart_load_data.sql` script.

```
vmartdb=> \i vmart_load_data.sql
```



```
         200
(1 row)

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

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

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

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

vmartdb=>
```

Note: It could take several minutes to load the default five-million row fact table on a typical hardware cluster. You can check the load by examining the *vertica*`.log` file, as described in Monitoring the Log Files in the Administrator's Guide.

Step 5: Create a Comprehensive Design

This procedure guides you through creating a comprehensive design and assumes you have already performed the following prerequisite steps:

- ***Set up the example environment*** (page 88)
- ***Created the example database*** (page 89)
- ***Defined the database schema*** (page 91)
- ***Loaded the data*** (page 92)

If you have not performed the above steps, refer to the ***Tutorial*** (page 87) in the Getting Started Guide.

1. Type `\q` to exit the vsql session and return to the Main Menu in the Administration Tools.

   Alternatively, restart the Administrative Tools:

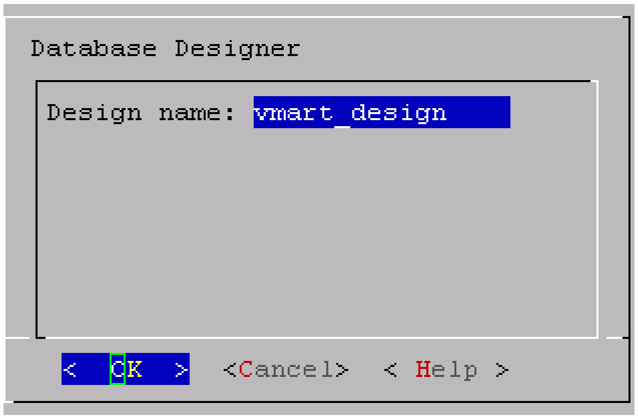
   ```
   $ /opt/vertica/bin/admintools
   ```
2. From the **Main Menu**, click **Configuration Menu** and click **OK.**
3. From the **Configuration Menu**, click **Run Database Designer**, and and click **OK.**
4. Select **vmartdb** as the database and click **OK**.

If you are asked to enter the password for the database, click **OK** to bypass. No password was assigned in ***Step 2: Create the Example Database*** (page 89), so you do not need to enter one now.

5 Click **OK** to accept the default directory for storing Database Designer output and log files. **Note this location**.

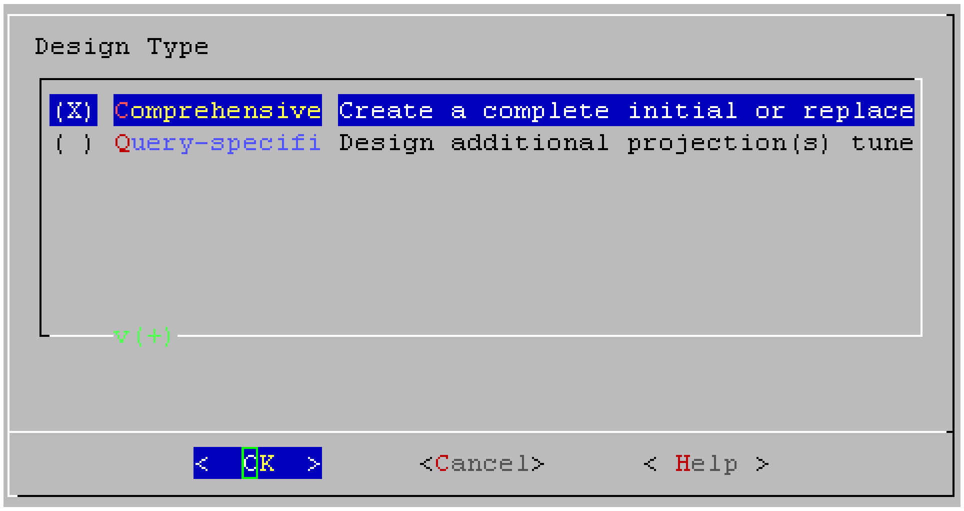
Note: If you choose to not deploy your design now, the Database Designer saves the SQL script to implement the design in this directory where you can review and manually deploy it later.

6 In the **Database Designer** window, enter a name for the design (this example uses **vmart_design**) and click **OK**.



```
Database Designer

Design name: vmart_design
```

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

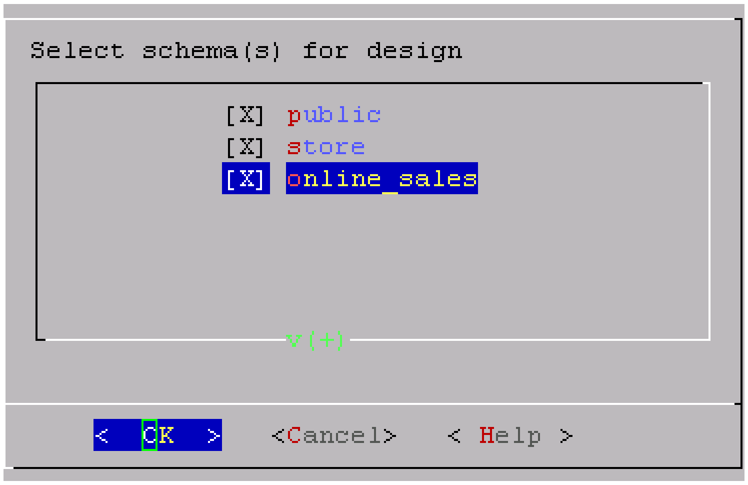


```
Design Type

(X) Comprehensive Create a complete initial or replace
( ) Query-specifi Design additional projection(s) tune
```

8 Select the schemas for your design, and click **OK**.

If you include a schema that contains tables without data, the Administration Tools returns a message notifying you that designing for tables without data could be suboptimal. You can choose to continue, but Vertica recommends that you click Cancel and deselect the schemas that contain empty tables before you proceed.

Note: In this example, the Vmart design is a multi-schema database, so be sure to select all three options: public, store, and online_sales



```
Select schema(s) for design

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

9 In the **Design Options** window, accept the default of all three options described below and click **OK**.

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

- *Optimize with queries*: Efficiency of the design is substantially improved if the Database Designer can access sample queries.

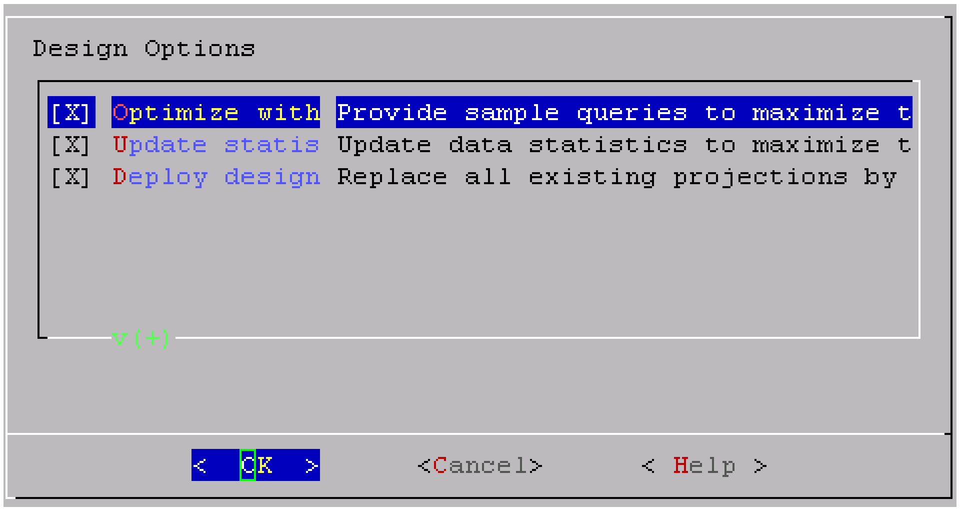
  Supplying the Database Designer with queries is especially important if you want to optimize the database design for query performance.

- *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.

  Note that updating statistics takes time and resources, so if the current statistics are up to date, this step is unnecessary. When in doubt, update statistics.

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

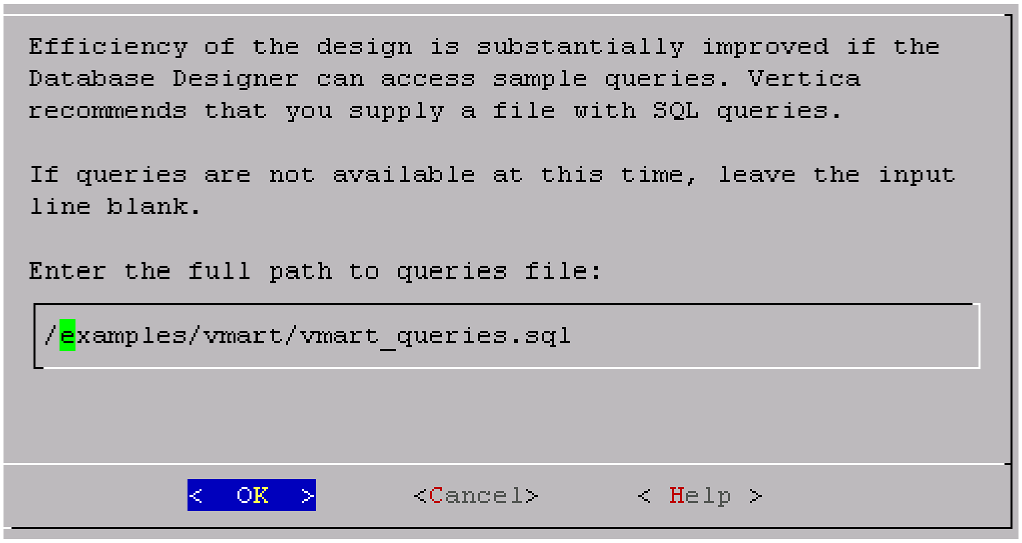
Note: For large databases, a full design session could take a long time, yet it is best to allow this process to complete uninterrupted. If the session must be canceled, use CTRL+C.



```
Design Options

[X] Optimize with Provide sample queries to maximize t
[X] Update statis Update data statistics to maximize t
[X] Deploy design Replace all existing projections by
```

10 If you selected the Optimize with queries option, you are prompted for the query file. Type the full path to the file containing the queries that will be run on your database. In this example it is:

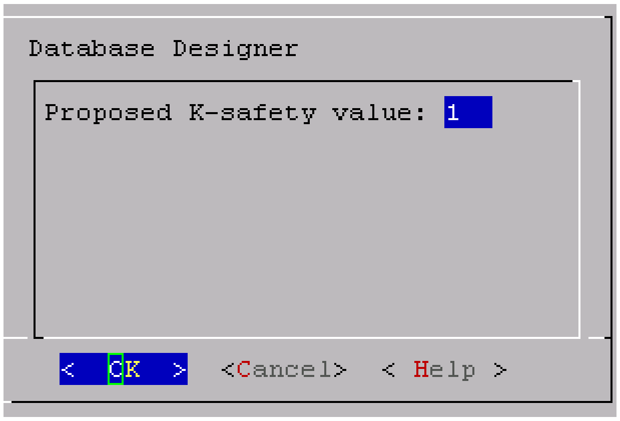
`/examples/VMart_Schema/vmart_queries.sql`



```
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:

/examples/vmart/vmart_queries.sql
```

11 Choose the **K-safety value** you want. In this example, it is 1. Click **OK**.

Note: There will be no K-safe form if you are creating a comprehensive design on a single node. In that case, you can skip this step.

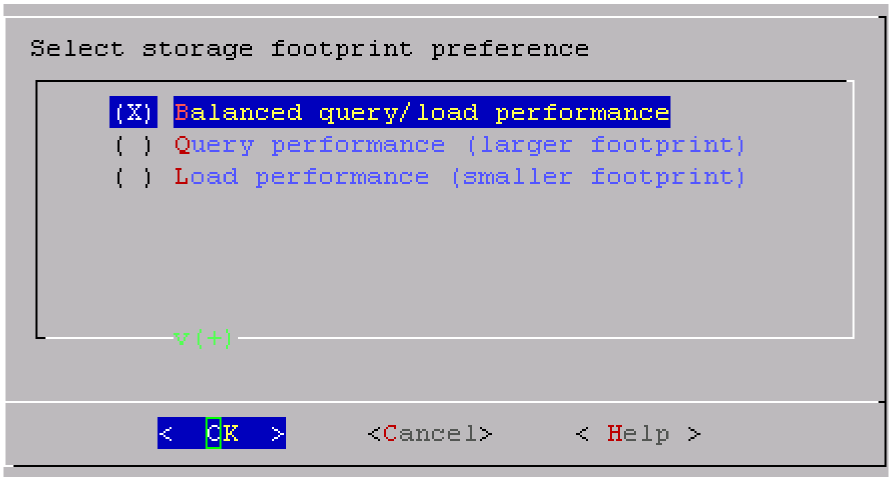


```
Database Designer

Proposed K-safety value: 1
```

12 Choose the Database Designer's priority for the design (in this procedure choose **Balanced**) and click **OK**.

The options are:

- *Balanced* query/load performance tells the Database Designer to create a design that is balanced between database size and query performance.
- *Query* load performance 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 is optimized for loads, minimizing size of the database, potentially at the expense of query performance.



```
Select storage footprint preference

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

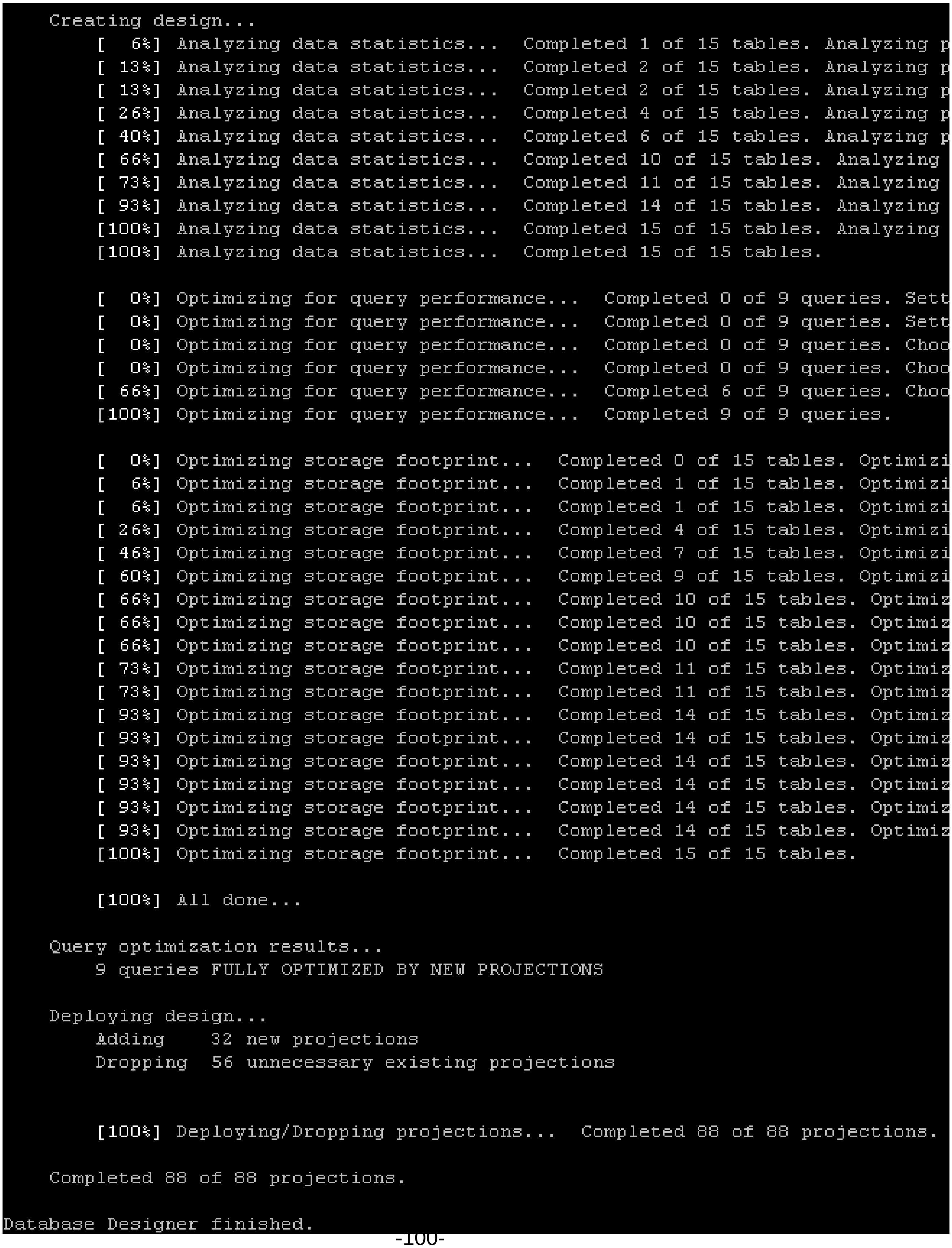
13 When the informational message displays, click **Proceed**.

The Database Designer:

- Sets up the design session
- Examines table data
- Loads queries from the query file you provided
- Creates the design

- Deploys the design or saves a SQL file containing the design, depending on what you selected for the Deploy design option in step 9.

You can watch the progress on the terminal window. The following image is just an example and might not match exactly what you see:



```
    Creating design...
        [  6%] Analyzing data statistics...  Completed 1 of 15 tables. Analyzing p
        [ 13%] Analyzing data statistics...  Completed 2 of 15 tables. Analyzing p
        [ 13%] Analyzing data statistics...  Completed 2 of 15 tables. Analyzing p
        [ 26%] Analyzing data statistics...  Completed 4 of 15 tables. Analyzing p
        [ 40%] Analyzing data statistics...  Completed 6 of 15 tables. Analyzing p
        [ 66%] Analyzing data statistics...  Completed 10 of 15 tables. Analyzing
        [ 73%] Analyzing data statistics...  Completed 11 of 15 tables. Analyzing
        [ 93%] Analyzing data statistics...  Completed 14 of 15 tables. Analyzing
        [100%] Analyzing data statistics...  Completed 15 of 15 tables. Analyzing
        [100%] Analyzing data statistics...  Completed 15 of 15 tables.

        [  0%] Optimizing for query performance...  Completed 0 of 9 queries. Sett
        [  0%] Optimizing for query performance...  Completed 0 of 9 queries. Sett
        [  0%] Optimizing for query performance...  Completed 0 of 9 queries. Choo
        [  0%] Optimizing for query performance...  Completed 0 of 9 queries. Choo
        [ 66%] Optimizing for query performance...  Completed 6 of 9 queries. Choo
        [100%] Optimizing for query performance...  Completed 9 of 9 queries.

        [  0%] Optimizing storage footprint...  Completed 0 of 15 tables. Optimizi
        [  6%] Optimizing storage footprint...  Completed 1 of 15 tables. Optimizi
        [  6%] Optimizing storage footprint...  Completed 1 of 15 tables. Optimizi
        [ 26%] Optimizing storage footprint...  Completed 4 of 15 tables. Optimizi
        [ 46%] Optimizing storage footprint...  Completed 7 of 15 tables. Optimizi
        [ 60%] Optimizing storage footprint...  Completed 9 of 15 tables. Optimizi
        [ 66%] Optimizing storage footprint...  Completed 10 of 15 tables. Optimiz
        [ 66%] Optimizing storage footprint...  Completed 10 of 15 tables. Optimiz
        [ 66%] Optimizing storage footprint...  Completed 10 of 15 tables. Optimiz
        [ 73%] Optimizing storage footprint...  Completed 11 of 15 tables. Optimiz
        [ 73%] Optimizing storage footprint...  Completed 11 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [ 93%] Optimizing storage footprint...  Completed 14 of 15 tables. Optimiz
        [100%] Optimizing storage footprint...  Completed 15 of 15 tables.

        [100%] All done...

    Query optimization results...
        9 queries FULLY OPTIMIZED BY NEW PROJECTIONS

    Deploying design...
        Adding    32 new projections
        Dropping  56 unnecessary existing projections


        [100%] Deploying/Dropping projections...  Completed 88 of 88 projections.

    Completed 88 of 88 projections.

Database Designer finished.
```

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

Note: The Database Designer creates a backup of the current design of your database before deploying the new design. This backup is stored in the output directory you entered in step 5, and is named `design_name_projection_backup_nnnnnnnnnn.sql`

Step 6: Connect to the Database and Run a Simple Query

Proceeding directly from Step 5:

1. Click **Main Menu** and click **OK**.
2. Click **Connect to Database** and click **OK**.

   The vsql welcome prompt displays:

   ```
   Welcome to vsql, the Vertica Analytic Database v4.0.11-20100427010202
     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
   vmartdb=>
   ```
3. Use the \i meta-command to execute your example query script:

   ```
   vmartdb=> \i vmart_query_01.sql
   ```

   See the following list for the example queries that Vertica supplies:

   - ***Clickstream Example Database*** (page 13)
   - ***Credit History Example Database*** (page 22)
   - ***Retail Sales Example Database*** (page 31)
   - ***Stock Exchange Example Database*** (page 45)
   - ***Telecom Example Database*** (page 58)
   - ***VMart Example Database*** (page 67)

See Also

Running Simple Queries (page 105)

Creating a Query-specific Design Using the Database Designer in the Administrator's Guide

Step 7: Test the Optimized Design

In order to test your optimized design, you can check query execution times:

1. Use the vsql `\timing` meta-command to enable the display of query execution time in milliseconds.

   Execute a SQL sample query script to test your schema and load scripts for errors.

   Note: Include a sample of queries your users are likely to run against the database. If you don't have any real queries, just write simple SQL that collects counts on each of your tables. Alternatively, you can skip this step.
2. Execute several ad hoc queries

1. Run Administration Tools and select Connect to Database.
2. Use the \i meta-command to execute the query script; for example:

```
vmartdb=> \i vmart_query_01.sql
```

Once the database has been optimized, it should run queries efficiently. However, you might discover additional queries that you want to optimize. If this is the case, modify and update the design. See Modifying Designs and Creating a Query-specific Design Using the Database Designer in the Administrator's Guide.

Step 8: (Optional) Generate Custom Data Files

Each example database provided with Vertica includes a sample data generator program that produces output files whose names correspond to the tables in the   logical schema. Each data generator has a similar set of input parameters that allow you to specify the number of rows of data to generate for any subset of the tables. To see a detailed list of the parameters for any example database, examine the README file in the example database directory.

Tip: You can repeat the tutorial using custom data files to test larger data sizes.

Syntax

```
./example_gen [ --files files ]
              [ --seed seed ]
              [ --time_file path ]
              [ --fact_table_name rows ]
              [ --dimension_table_name rows ] ...
```

Parameters

*example*_gen	Where *example* is one of the following: clickstream credithistory retail stock telecom vmart

files *files*	Splits the fact table data into the specified number of files. By default, the data generator produces a single, unnumbered fact table data file. If you specify a value of two (2) or more, the data generator numbers the files by appending an underscore character (_) and three digits to the file name, starting at _001. For example:
	`./retail_gen --files 3`
	produces:
	`Retail_Sales_Fact_001.tbl`
	`Retail_Sales_Fact_002.tbl`
	`Retail_Sales_Fact_003.tbl`
	Default: 1

seed *seed*	Is the seed for the pseudo-random number generator. If you use the same seed each time you run the data generator, you get the same data files (excluding external factors); for example, `seed 9999`. Default: 20177
time_file *path*	Is the pathname of the pre-computed time data input file used to generate the ***Date Dimension*** (see "***Date_Dimension***" on page 32) table. Default: `./Time.txt`

	This Vertica-supplied file is provided for each example database and the date range may vary; for example 2000-2004 or 2003-2007.
fact_table_name rows	Is the name of the fact table in *example* followed by the number of rows of data to generate for the fact table. Default: 5,000,000 (five million)
dimension_table_name rows	Is the name of a dimension table in *example* (other than the `Date_Dimension` table) followed by the number of rows of data to generate for that dimension table.

Notes

- The number of rows in `Date_Dimension` tables is determined by the time data input file supplied with the example database.
- If you are using multiple fact table data files, make sure that your fact table load script(s) contain the correct file names as described in Using Load Scripts.

Examples

```
./retail_gen
./retail_gen --files 3
/home/dbadmin/Retail_Schema/retail_gen \
--seed 9999
--time_file /home/dbadmin/Retail_Schema/Time.txt \
--retail_sales_fact 100000 \
--product_dimension 500 \
--store_dimension 50 \
--promotion_dimension 100
```

Running Simple Queries

Each example database includes example SQL queries that represent the kinds of queries you might use in a production database. If you copy the query files to a client system, you can connect to the example database and execute the queries using any of the methods described in the Programmer's Guide.

To run an example query using vsql on a cluster host:

1 Run Administration Tools and select Connect to Database.

```
Welcome to the vsql, Vertica_Database v4.1.x 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
vmartdb=>
```

2 Use the \i meta-command to execute the query script:

```
vmartdb=> \i vmart_query_01.sql
```

See the following list for the example queries that Vertica supplies:

- ***Clickstream Example Database*** (page 13)
- ***Credit History Example Database*** (page 22)
- ***Retail Sales Example Database*** (page 31)
- ***Stock Exchange Example Database*** (page 45)
- ***Telecom Example Database*** (page 58)
- ***VMart Example Database*** (page 67)

Cleanup Procedure

If you want to clean up your host and start over from scratch, use the following steps.

Drop the database

1. In a terminal window, log in to the database administrator account that was created by the installation script. The default account name is `dbadmin`.
2. Run the Administration Tools.

   ```
   $ /opt/vertica/bin/admintools
   ```
3. If necessary, stop any running database (Main Men **Stop Database**).
4. 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**.

Uninstall Vertica

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

Other

1. Optionally remove the dbadmin account on all cluster hosts.
2. Remove any example database directories you created.

See Also

For complete descriptions of each Admin Tools dialog, refer to the Administration Tools Reference in the Administrator's Guide.

Using the Graphical User Interface

This is only a quick reference. It is not a complete guide to keystroke usage. See Using the Administration Tools in the Administrator's Guide for full details.

Return	Run selected command.
Tab	Move cursor from OK to Cancel to Help to menu or to OK...
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.

Notes for Remote Terminal Users

The appearance of the graphical interface 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 Windows XP.

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, you can 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.

Index

Copyright Notice

Vertica Systems, Inc.
8 Federal Street
Billerica, MA 01821
Phone: (978) 600-1000
Fax: (978) 600-1001
E-Mail: info@vertica.com
Web site: http://www.vertica.com
(***http://www.vertica.com***)

Trademarks

Open Source Software Acknowledgments

Vertica makes no representations or warranties regarding any third party software. All third-party software is provided or recommended by Vertica on an AS IS basis.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

ASMJIT

Boost

bzip2

This file is a part of bzip2 and/or libbzip2, a program and library for lossless, block-sorting data compression.

Julian Seward, Cambridge, UK.

jseward@bzip.org <mailto:jseward@bzip.org>

bzip2/libbzip2 version 1.0 of 21 March 2000

This program is based on (at least) the work of:

Mike Burrows

David Wheeler

Peter Fenwick

Alistair Moffat

Radioed Neal

Ian H. Witten

Robert Sedgewick

Jon L. Bentley

Daemonize

Ganglia Open Source License

ICU (International Components for Unicode) License - ICU 1.8.1 and later

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

Keepalived Vertica IPVS (IP Virtual Server) Load Balancer

Copyright © 2007 Free Software Foundation, Inc.

http://fsf.org/

The keepalived software contained in the `VerticaIPVSLoadBalancer-4.1.x.RHEL5.x86_64.rpm` software package is licensed under the GNU General Public License ("GPL"). You are entitled to receive the source code for such software. For no less than three years from the date you obtained this software package, you may download a copy of the source code for the software in this package licensed under the GPL at no charge by visiting http://www.vertica.com/licenses/keepalived-1.1.17.tar.gz ***http://www.vertica.com/licenses/keepalived-1.1.17.tar.gz***. You may download this source code so that it remains separate from other software on your computer system.

jQuery

Copyright © 2009 John Resig, http://jquery.com/

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

Lighttpd Open Source License

MersenneTwister.h

MIT Kerberos

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

THIS SOFTWARE IS PROVIDED “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The implementation of the AES encryption algorithm in src/lib/crypto/aes has the following copyright:

LICENSE TERMS

The free distribution and use of this software in both source and binary form is allowed (with or without changes) provided that:

1 Distributions of this source code include the above copyright notice, this list of conditions and the following disclaimer.
2 Distributions in binary form include the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other associated materials.
3 The copyright holder's name is not used to endorse products built using this software without specific written permission.

DISCLAIMER

This software is provided 'as is' with no explicit or implied warranties in respect of any properties, including, but not limited to, correctness and fitness for purpose.

The implementations of GSSAPI mechglue in GSSAPI-SPNEGO in src/lib/gssapi, including the following files:

- lib/gssapi/generic/gssapi_err_generic.et
- lib/gssapi/mechglue/g_accept_sec_context.c
- lib/gssapi/mechglue/g_acquire_cred.c
- lib/gssapi/mechglue/g_canon_name.c
- lib/gssapi/mechglue/g_compare_name.c
- lib/gssapi/mechglue/g_context_time.c
- lib/gssapi/mechglue/g_delete_sec_context.c
- lib/gssapi/mechglue/g_dsp_name.c
- lib/gssapi/mechglue/g_dsp_status.c
- lib/gssapi/mechglue/g_dup_name.c
- lib/gssapi/mechglue/g_exp_sec_context.c
- lib/gssapi/mechglue/g_export_name.c
- lib/gssapi/mechglue/g_glue.c
- lib/gssapi/mechglue/g_imp_name.c

- lib/gssapi/mechglue/g_imp_sec_context.c
- lib/gssapi/mechglue/g_init_sec_context.c
- lib/gssapi/mechglue/g_initialize.c
- lib/gssapi/mechglue/g_inquire_context.c
- lib/gssapi/mechglue/g_inquire_cred.c
- lib/gssapi/mechglue/g_inquire_names.c
- lib/gssapi/mechglue/g_process_context.c
- lib/gssapi/mechglue/g_rel_buffer.c
- lib/gssapi/mechglue/g_rel_cred.c
- lib/gssapi/mechglue/g_rel_name.c
- lib/gssapi/mechglue/g_rel_oid_set.c
- lib/gssapi/mechglue/g_seal.c
- lib/gssapi/mechglue/g_sign.c
- lib/gssapi/mechglue/g_store_cred.c
- lib/gssapi/mechglue/g_unseal.c
- lib/gssapi/mechglue/g_userok.c
- lib/gssapi/mechglue/g_utils.c
- lib/gssapi/mechglue/g_verify.c
- lib/gssapi/mechglue/gssd_pname_to_uid.c
- lib/gssapi/mechglue/mglueP.h
- lib/gssapi/mechglue/oid_ops.c
- lib/gssapi/spnego/gssapiP_spnego.h
- lib/gssapi/spnego/spnego_mech.c

are subject to the following license:

Copyright © 2004 Sun Microsystems, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Npgsql-.Net Data Provider for Postgresql

Copyright © 2002-2008, The Npgsql Development Team

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE NPGSQL DEVELOPMENT TEAM BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE NPGSQL DEVELOPMENT TEAM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE NPGSQL DEVELOPMENT TEAM SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE NPGSQL DEVELOPMENT TEAM HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Open LDAP

The OpenLDAP Public License

Version 2.8, 17 August 2003

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions in source form must retain copyright statements and notices,
2. Redistributions in binary form must reproduce applicable copyright statements and notices, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution, and
3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use this Software under terms of this license revision or under the terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S) OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in advertising or otherwise to promote the sale, use or other dealing in this Software without specific, written prior permission. Title to copyright in this Software shall at all times remain with copyright holders.

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Open SSL

OpenSSL License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3 All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"

4 The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.

5 Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.

6 Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

PCRE LICENCE

PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language.

Release 8 of PCRE is distributed under the terms of the "BSD" licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself.

The basic library functions are written in C and are freestanding. Also included in the distribution is a set of C++ wrapper functions.

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel

Email local part: ph10

Email domain: cam.ac.uk

University of Cambridge Computing Service,

Cambridge, England.

Copyright (c) 1997-2010 University of Cambridge

All rights reserved.

THE C++ WRAPPER FUNCTIONS

Contributed by: Google Inc.

Copyright (c) 2007-2010, Google Inc.

All rights reserved.

THE "BSD" LICENCE

- Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF ERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

End

Perl Artistic License

Copyright © August 15, 1997

Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

Definitions

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1 You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2 You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3 You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

4 place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

   1. use the modified Package only within your corporation or organization.
   2. rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.
   3. make other distribution arrangements with the Copyright Holder.

5 You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

   1. distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.
   2. accompany the distribution with the machine-readable source of the Package with your modifications.

3. give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.
4. make other distribution arrangements with the Copyright Holder.

6 You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

7 The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package via the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

8 C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

9 Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

10 The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

Pexpect

Copyright © 2010 Noah Spurrier

Credits: Noah Spurrier, Richard Holden, Marco Molteni, Kimberley Burchett, Robert Stone, Hartmut Goebel, Chad Schroeder, Erick Tryzelaar, Dave Kirby, Ids vander Molen, George Todd, Noel Taylor, Nicolas D. Cesar, Alexander Gattin, Geoffrey Marshall, Francisco Lourenco, Glen Mabey, Karthik Gurusamy, Fernando Perez, Corey Minyard, Jon Cohen, Guillaume Chazarain, Andrew Ryan, Nick Craig-Wood, Andrew Stone, Jorgen Grahn (Let me know if I forgot anyone.)

Free, open source, and all that good stuff.

PHP License

THIS SOFTWARE IS PROVIDED BY THE PHP DEVELOPMENT TEAM ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PHP DEVELOPMENT TEAM OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the PHP Group.

The PHP Group can be contacted via Email at group@php.net.

For more information on the PHP Group and the PHP project, please see <http://www.php.net>.

PHP includes the Zend Engine, freely available at <http://www.zend.com>.

PostgreSQL

This product uses the PostgreSQL Database Management System(formerly known as Postgres, then as Postgres95)

Portions Copyright © 1996-2005, The PostgreSQL Global Development Group
Portions Copyright © 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Python Dialog

The Administration Tools part of this product uses Python Dialog, a Python module for doing console-mode user interaction.

Upstream Author:

Peter Astrand <peter@cendio.se>
Robb Shecter <robb@acm.org>

Sultanbek Tezadov <http://sultan.da.ru>
Florent Rougon <flo@via.ecp.fr>
Copyright © 2000   Robb Shecter, Sultanbek Tezadov
Copyright © 2002, 2003, 2004   Florent Rougon

License:

This package is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This package is distributed in the hope that it is useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this package; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

The complete source code of the Python dialog package and complete text of the GNU Lesser General Public License can be found on the Vertica Systems Web site at
http://www.vertica.com/licenses/pythondialog-2.7.tar.bz2

http://www.vertica.com/licenses/pythondialog-2.7.tar.bz2

RRDTool Open Source License

Note: rrdtool is a dependency of using the ganglia-web third-party tool. RRDTool allows the graphs displayed by ganglia-web to be produced.

RRDTOOL - Round Robin Database Tool
A tool for fast logging of numerical data graphical display of this data.
Copyright © 1998-2008 Tobias Oetiker
All rights reserved.

GNU GPL License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.   See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

FLOSS License Exception

(Adapted from http://www.mysql.com/company/legal/licensing/foss-exception.html)

I want specified Free/Libre and Open Source Software ("FLOSS") applications to be able to use specified GPL-licensed RRDtool libraries (the "Program") despite the fact that not all FLOSS licenses are compatible with version 2 of the GNU General Public License (the "GPL").

As a special exception to the terms and conditions of version 2.0 of the GPL:

You are free to distribute a Derivative Work that is formed entirely from the Program and one or more works (each, a "FLOSS Work") licensed under one or more of the licenses listed below, as long as:

1 You obey the GPL in all respects for the Program and the Derivative Work, except for identifiable sections of the Derivative Work which are not derived from the Program, and which can reasonably be considered independent and separate works in themselves

2 All identifiable sections of the Derivative Work which are not derived from the Program, and which can reasonably be considered independent and separate works in themselves

- are distributed subject to one of the FLOSS licenses listed below, and
- the object code or executable form of those sections are accompanied by the complete corresponding machine-readable source code for those sections on the same medium and under the same FLOSS license as the corresponding object code or executable forms of those sections.

3 Any works which are aggregated with the Program or with a Derivative Work on a volume of a storage or distribution medium in accordance with the GPL, can reasonably be considered independent and separate works in themselves which are not derivatives of either the Program, a Derivative Work or a FLOSS Work.

If the above conditions are not met, then the Program may only be copied, modified, distributed or used under the terms and conditions of the GPL.

FLOSS License List

License name	**Version(s)/Copyright Date**
Academic Free License	2.0
Apache Software License	1.0/1.1/2.0
Apple Public Source License	2.0
Artistic license	From Perl 5.8.0
BSD license	"July 22 1999"
Common Public License	1.0
GNU Library or "Lesser" General Public License (LGPL)	2.0/2.1
IBM Public License, Version	1.0
Jabber Open Source License	1.0
MIT License (As listed in file MIT-License.txt)	-
Mozilla Public License (MPL)	1.0/1.1
Open Software License	2.0
OpenSSL license (with original SSLeay license)	"2003" ("1998")
PHP License	3.0
Python license (CNRI Python License)	-
Python Software Foundation License	2.1.1
Sleepycat License	"1999"

W3C License	"2001"
X11 License	"2001"
Zlib/libpng License	-
Zope Public License	2.0/2.1

Spread

This product uses software developed by Spread Concepts LLC for use in the Spread toolkit. For more information about Spread see http://www.spread.org (***http://www.spread.org***).

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer and request.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer and request in the documentation and/or other materials provided with the distribution.
3. All advertising materials (including web pages) mentioning features or use of this software, or software that uses this software, must display the following acknowledgment: "This product uses software developed by Spread Concepts LLC for use in the Spread toolkit. For more information about Spread see http://www.spread.org"
4. The names "Spread" or "Spread toolkit" must not be used to endorse or promote products derived from this software without prior written permission.
5. Redistributions of any form whatsoever must retain the following acknowledgment:
6. "This product uses software developed by Spread Concepts LLC for use in the Spread toolkit. For more information about Spread, see http://www.spread.org"
7. This license shall be governed by and construed and enforced in accordance with the laws of the State of Maryland, without reference to its conflicts of law provisions. The exclusive jurisdiction and venue for all legal actions relating to this license shall be in courts of competent subject matter jurisdiction located in the State of Maryland.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, SPREAD IS PROVIDED UNDER THIS LICENSE ON AN AS IS BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT SPREAD IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. ALL WARRANTIES ARE DISCLAIMED AND THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE CODE IS WITH YOU. SHOULD ANY CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE COPYRIGHT HOLDER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

Although NOT REQUIRED, we at Spread Concepts would appreciate it if active users of Spread put a link on their web site to Spread's web site when possible. We also encourage users to let us know who they are, how they are using Spread, and any comments they have through either e-mail (spread@spread.org) or our web site at (http://www.spread.org/comments).

SNMP

Various copyrights apply to this package, listed in various separate parts below. Please make sure that you read all the parts. Up until 2001, the project was based at UC Davis, and the first part covers all code written during this time. From 2001 onwards, the project has been based at SourceForge, and Networks Associates Technology, Inc hold the copyright on behalf of the wider Net-SNMP community, covering all derivative work done since then. An additional copyright section has been added as Part 3 below also under a BSD license for the work contributed by Cambridge Broadband Ltd. to the project since 2001. An additional copyright section has been added as Part 4 below also under a BSD license for the work contributed by Sun Microsystems, Inc. to the project since 2003.

Code has been contributed to this project by many people over the years it has been in development, and a full list of contributors can be found in the README file under the THANKS section.

Part 1: CMU/UCD copyright notice: (BSD like)

Part 2: Networks Associates Technology, Inc copyright notice (BSD)

Part 3: Cambridge Broadband Ltd. copyright notice (BSD)

- The name of Cambridge Broadband Ltd. may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Part 4: Sun Microsystems, Inc. copyright notice (BSD)

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara,

California 95054, U.S.A. All rights reserved.

Use is subject to license terms below.

This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Sun Microsystems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Part 5: Sparta, Inc copyright notice (BSD)

Copyright © 2003-2006, Sparta, Inc

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Sparta, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Part 6: Cisco/BUPTNIC copyright notice (BSD)

Copyright © 2004, Cisco, Inc and Information Network Center of Beijing University of Posts and Telecommunications.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Cisco, Inc, Beijing University of Posts and Telecommunications, nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Part 7: Fabasoft R&D Software GmbH & Co KG copyright notice (BSD)

oss@fabasoft.com

Author: Bernhard Penz

Tecla Command-line Editing

Webmin Open Source License

xerces

NOTICE file corresponding to section 4(d) of the Apache License,
Version 2.0, in this case for the Apache Xerces distribution.

This product includes software developed by The Apache Software Foundation (http://www.apache.org/).

Portions of this software were originally based on the following:

Software copyright © 1999, IBM Corporation., http://www.ibm.com.

zlib

This is used by the project to load zipped files directly by COPY command. www.zlib.net/

zlib.h -- interface of the 'zlib' general purpose compression library version 1.2.3, July 18th, 2005

Copyright © 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1 The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

2 Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

3 This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu