Vertica Database 2.0.5-0

Quick Start

Date of Publication: 1/8/2008

CONFIDENTIAL

Copyright Notice

Vertica Systems, Inc.

Three Dundee Park Drive, Suite 102

Andover, MA 01810-3723

Phone: (978) 475-1070

Fax: (978) 475-6855

E-Mail: info@vertica.com

Web site: http://www.vertica.com
(http://www.vertica.com)

Trademarks

Vertica™ and the Vertica Database™ are trademarks of Vertica Systems, Inc.

Adobe®, Acrobat®, and Acrobat® Reader® are registered trademarks of Adobe Systems Incorporated.

AMD™ is a trademark of Advanced Micro Devices, Inc. in the United States and other countries.

Fedora™ is a trademark of Red Hat, Inc.

Intel® is a registered trademark of Intel.

Linux® is a registered trademark of Linus Torvalds.

Microsoft® is a registered trademark of Microsoft Corporation.

Novell® is a registered trademark and SUSE™ is a trademark of Novell, Inc. in the United States and other countries.

Oracle® is a registered trademark of Oracle Corporation.

Red Hat® is a registered trademark of Red Hat, Inc.

VMware® is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

Other products mentioned may be trademarks or registered trademarks of their respective companies.

Open Source Software Acknowledgements

Boost

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

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, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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>

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 will be 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

On Vertica systems, 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 website at http://www.vertica.com/licenses/pythondialog-2.7.tar.bz2 http://www.vertica.com/licenses/pythondialog-2.7.tar.bz2

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:

"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"

6. 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.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY OTHER CONTRIBUTOR BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES FOR LOSS OF PROFITS, REVENUE, OR FOR LOSS OF INFORMATION OR ANY OTHER LOSS.

YOU EXPRESSLY AGREE TO FOREVER INDEMNIFY, DEFEND AND HOLD HARMLESS THE COPYRIGHT HOLDERS AND CONTRIBUTORS OF SPREAD AGAINST ALL CLAIMS, DEMANDS, SUITS OR OTHER ACTIONS ARISING DIRECTLY OR INDIRECTLY FROM YOUR ACCEPTANCE AND USE OF SPREAD.

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).

Contents

Technical Support

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

http://www.vertica.com/support (http://www.vertica.com/support)

You must be a registered user in order to access the page.

Before reporting a problem, please run the Diagnostics Utility described in the Troubleshooting Guide and attach the resulting .zip file.

About the Documentation

Where to Find the Vertica Documentation

Vertica Systems, Inc. recommends that you copy the Vertica documentation from the database server (any cluster host) to a client system on which you can use a browser and/or the Adobe Reader.

Database Server Systems

The Vertica Database Documentation Set is automatically installed in the **/opt/vertica/doc/** directory on all cluster hosts. If you have a browser and/or the Adobe Reader installed on a cluster host, you can access the documentation directly.

`/opt/vertica/doc/HTML/Master/index.htm`
`/opt/vertica/doc/JDBC/index.htm`
`/opt/vertica/doc/PDF/`*`book-name`*`.pdf`

Database Client Systems

To create a copy of the Vertica documentation on a client system, do one of the following:

- Download the documentation package (**.tar.gz** or **.zip**) from the Vertica Systems, Inc. Web site and extract the files to a directory on the client system, using the original pathnames
- Copy the documentation directories from a database server system to a convenient location in your client system. All cross-references within the HTML documentation are relative so there is no location dependency.

`<`*`your-location`*`>HTML/Master/index.htm`
`<`*`your-location`*`>JDBC/index.htm`
`<`*`your-location`*`>PDF/`*`book-name`*`.htm`

World Wide Web

You can read and/or download the Vertica documentation from the **Vertica Systems, Inc. Web site**:

http://www.vertica.com/v-zone/product_documentation http://www.vertica.com/v-zone/product_documentation.

You need a V-Zone login to access the Product Documentation page.

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, refer to the documentation on your database server or client systems.

Reading the HTML Files

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

The Vertica documentation has been tested on the following browsers:

- Internet Explorer 7
- FireFox
- Opera
- Safari

Please report any script, image rendering, or text formatting problems to ***Technical Support*** (on page 11).

The Vertica documentation may contain links to Web sites of other companies or organizations that Vertica does not own or control. If any of these links are broken, please inform us.

Printing the PDF Files

The documentation files are supplied in **Adobe Acrobat™ PDF** document format for the purpose of making printed copies as needed. The documents are designed to be printed on standard 8½ x 11 paper using full duplex (two sided printing).

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

HTML links to the PDF files are provided here for browser access.

- Database Administrator's Guide
- Database Administrator's Guide (Advanced)
- Glossary of Terms
- Installation Guide
- Product Overview
- Quick Start
- Release Notes
- SQL Programmer's Guide
- SQL Reference Manual
- Troubleshooting Guide

Suggested Reading Paths

This section provides a suggested reading path for various types of users. Read the manuals listed under All Users first.

All Users

- Product Overview (basic concepts critical to understanding Vertica)
- Quick Start (step-by-step guide to getting Vertica up and running)
- Glossary of Terms (glossary of terms)

System Administrators

- Installation Guide (platform configuration and software installation)
- Release Notes (release-specific information)
- Troubleshooting Guide (general troubleshooting information)

Database Administrators

- Installation Guide (platform configuration and software installation)
- Database Administrator's Guide (database configuration, loading, security, and maintenance)
- Troubleshooting Guide (general troubleshooting information)

Application Developers

- SQL Programmer's Guide (connecting to a database, queries, transactions, etc.)
- SQL Reference Manual (Vertica-specific language information)
- Troubleshooting Guide (general troubleshooting 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

Typographical Conventions

It is important to understand the terms and typographical conventions used in this document.

General Convention	Description
colored bold text	introduces new terms defined either in the text, the glossary, or both.
normal italic text	indicates emphasis and the titles of other documents.
UPPERCASE TEXT	indicates the name of an SQL command or keyword.
`monospace text`	indicates literal interactive or programmatic input/output.
`italic monospace text`	indicates user-supplied information in interactive or programmatic input/output.
`bold monospace text`	indicates literal interactive user input
↵	indicates the Return/Enter key; implicit on all user input that includes text

SQL Syntax Convention	Description
indentation	is an attempt to maximize readability; SQL is a free-form language.
backslash \	continuation character used to indicate text that is too long to fit on a single line.
braces { }	indicate required items.
brackets [ ]	indicate optional items.
ellipses ...	indicate an optional sequence of similar items.
vertical ellipses ⋮	indicate an optional sequence of similar items or that part of the text has been omitted for readability.
vertical line |	within braces or brackets, indicates a choice .

Overview

Welcome to the Quick Start. This document is presented as a tutorial that takes you through the process of configuring a Vertica database and running example queries.

> Vertica recommends that you read the Product Overview manual before using this tutorial, in order to get a minimal understanding of unfamiliar concepts.

User Interfaces

In following this tutorial, you will use the following user interfaces:

- The Linux command line (shell) interface
- The Vertica Administration Tools (see the Database Administrator's Guide for details)
- The vsql client interface (see the SQL 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 30).

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

The example databases are located in /opt/vertica/doc/ on all cluster hosts. The databases are provided as:

- directories containing files
- .tar.gz files
- .zip files

Tutorial Procedure

The Tutorial Procedure describes how to configure a Vertica database that you can use to execute 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 tutorial procedure on the Administration Host.

Example Queries

Each example database includes several queries that are intended to represent ones that might be used in a real business. Feel free to write and run queries of your own to get familiar with querying a star schema. Detailed instructions are provided in ***Running Simple Queries*** (page 27).

Cleanup Procedure

When you have finished with the tutorial, you can restore your host machines to their original state. Detailed instructions are provided in ***Cleanup Procedure*** (page 29).

Tutorial Procedure

This tutorial procedure assumes that you have already installed Vertica on a cluster of hosts as described in the Installation Guide.

This procedure omits screen captures of Administration Tools dialogs that are self-explanatory. For complete descriptions of each dialog, refer to the Administration Tools Reference in the Database Administrator's Guide.

1. **Choose an example database**. The tutorial procedure is the same for all of the ***example databases*** (page 30). The examples in this section use use the Stock Exchange Example Database. If you are using a different database, replace `stock` with `clickstream`, `credithistory`, `retail`, or `telecom` in each example.
2. **Log in to the database administrator account** that was created by the installation script. The default account name is **dbadmin**.
3. **Copy your chosen example database directory** to a directory on the Administration Host. Do not use the default data directory (`home/dbadmin`). For example:

   ```
   $ mkdir /scratch/examples
   $ cp -r /opt/vertica/doc/Stock_Schema /scratch/examples
   ```
4. **Set your current directory** to the example database directory:

   ```
   $ cd /scratch/examples/Stock_Schema
   ```

   Do not change directory while following this tutorial. Some of the steps depend on it being set to the example database directory.
5. **Run the sample data generator** program using the default parameters. The parameter names are listed in the README file.

   ```
   $ ./stock_gen
   Using default parameters
   numfiles = 1
   numfactrows = 5000000
   numstockkeys = 273
   numsplitkeys = 500
   numtraderkeys = 200
   random# = 20177
   timefile = Time.txt
   ```

   If the stock_gen executable file is not present or does not work correctly, recompile it. This example uses the GNU C++ compiler, which is a ***free download*** (http://gcc.gnu.org/). You can use any other C++ compiler.

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

If you are using VMware, the default 5M row fact table load will probably fail. Specify a smaller fact table size such as 1000000 (1M) rows as described in ***Generating Custom Data Files*** (page 78). The maximum size of a bulk load depends on the system resources and cannot be determined accurately.

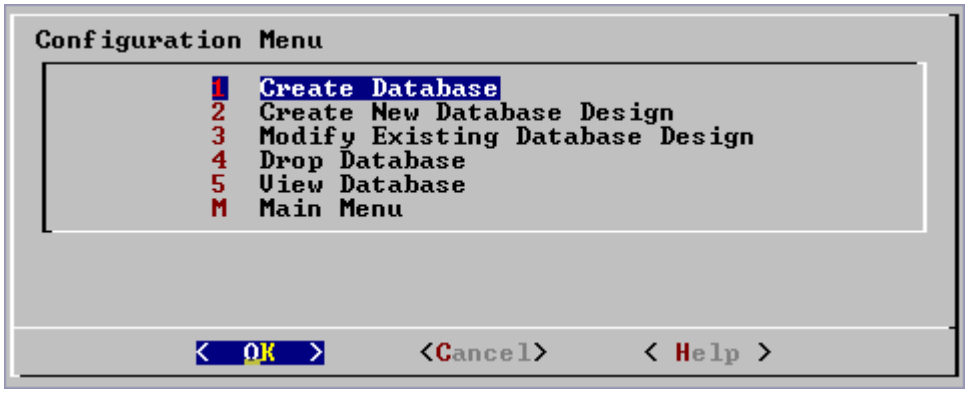
6. **Run the Administration Tools**.

   ```
   $ /opt/vertica/bin/adminTools
   ```

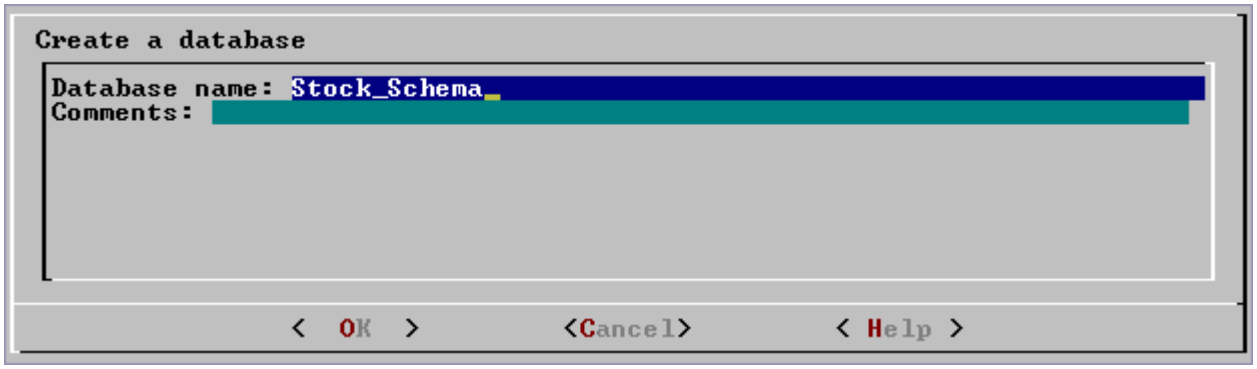
   A quick reference to the ***Administration Tools Keystrokes*** (see "Using the Graphical User Interface" on page 79) is available at the end of this document.

   If you are using a remote terminal application such as PuTTY or a Cygwin bash shell, see ***Notes for Remote Terminal Users*** (on page 80).

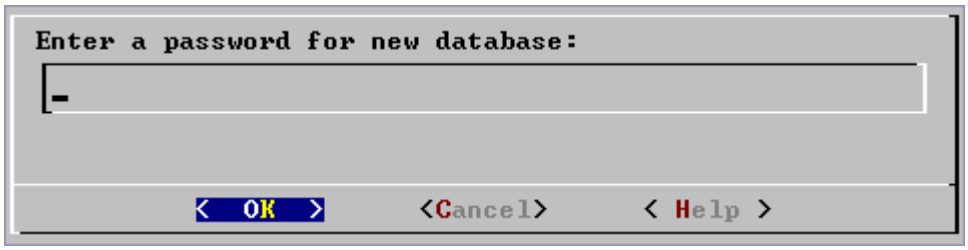
7. **Accept the license agreement** (once only).
8. Specify the location of your **license key file** (once only).
9. Go to the **Configuration** menu and select Create Database.



10. Create a database named **Stock_Schema.**



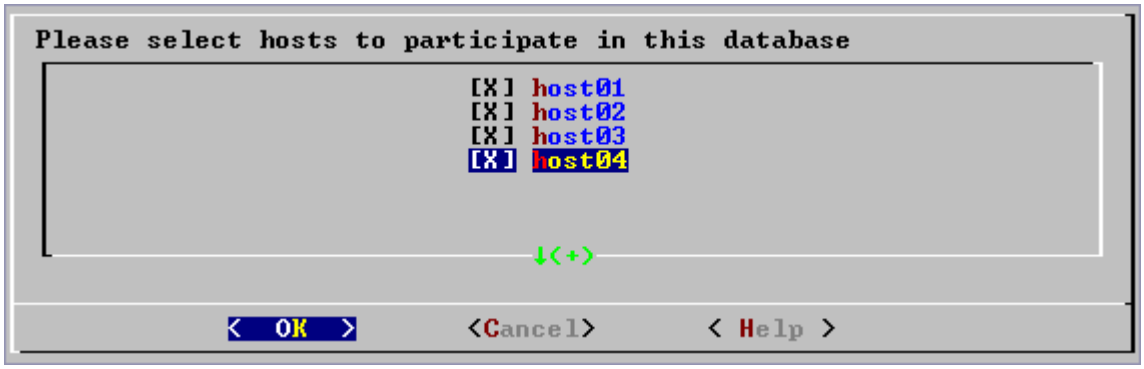
11. **Do not specify a password**.



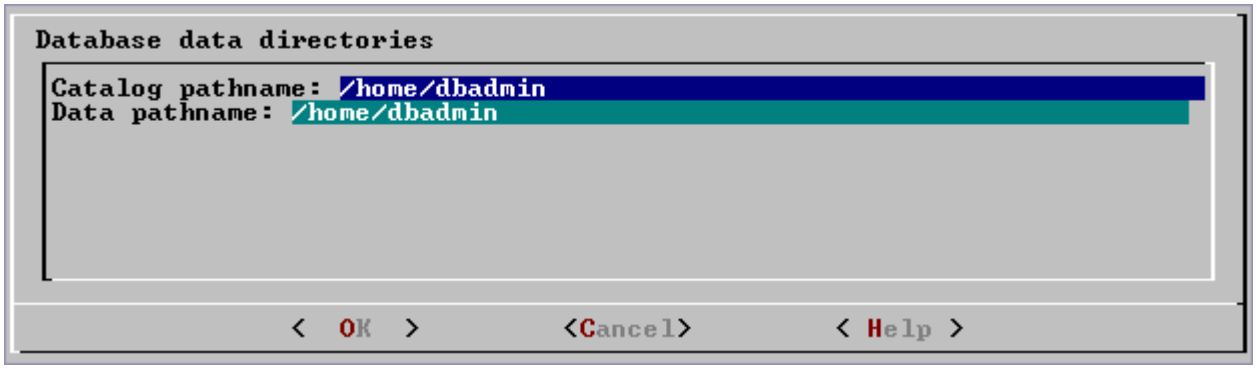
12. **Confirm** that you do not want a password.

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

13. **Select the hosts** that you want to include in the database cluster.

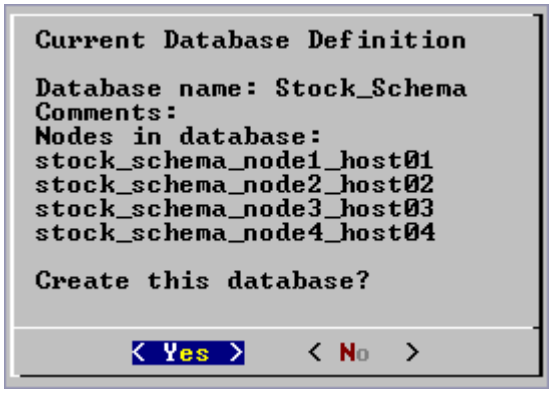


14. **Use the default pathnames** for the data and catalog directories. Catalog and data pathnames must contain only alphanumeric characters and cannot have leading space characters. Failure to comply with these restrictions may result in database creation failure.



When you create a production database, you will probably specify other locations. See Prepare Disk Storage Locations in the Database Administrator's Guide for more information.

15. **Create the database**.



Vertica automatically creates a set of node definitions based on the database name and the names of the hosts you selected.

16. Go back to the **Main Menu** and select Connect to Database.

```
Welcome to the vsql, Vertica_Database v2.0.5-0 interactive terminal.
Type:  \h for help with SQL commands
       \? for help with vsql commands
       \g or terminate with semicolon to execute query
       \q to quit
Stock_Schema=>
```

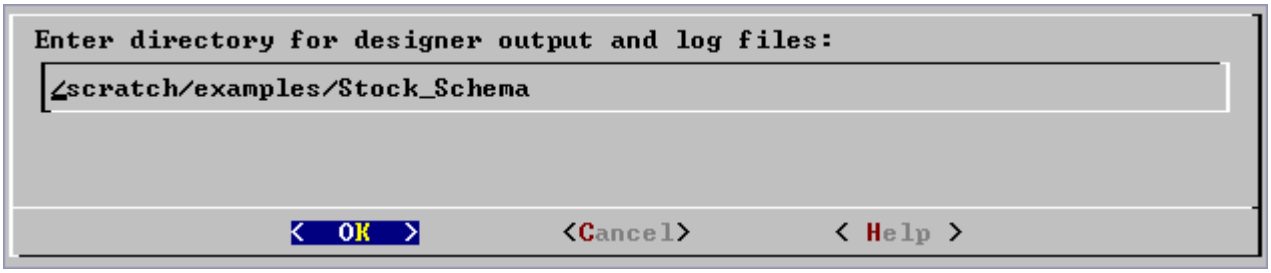
17. Execute the SQL script **stock_define_schema.sql** using the \i meta-command in vsql. This creates the tables and referential integrity constraints that make up the logical schema.

```
Stock_Schema=> \i stock_define_schema.sql
CREATE TABLE
ALTER TABLE
      ⋮
```

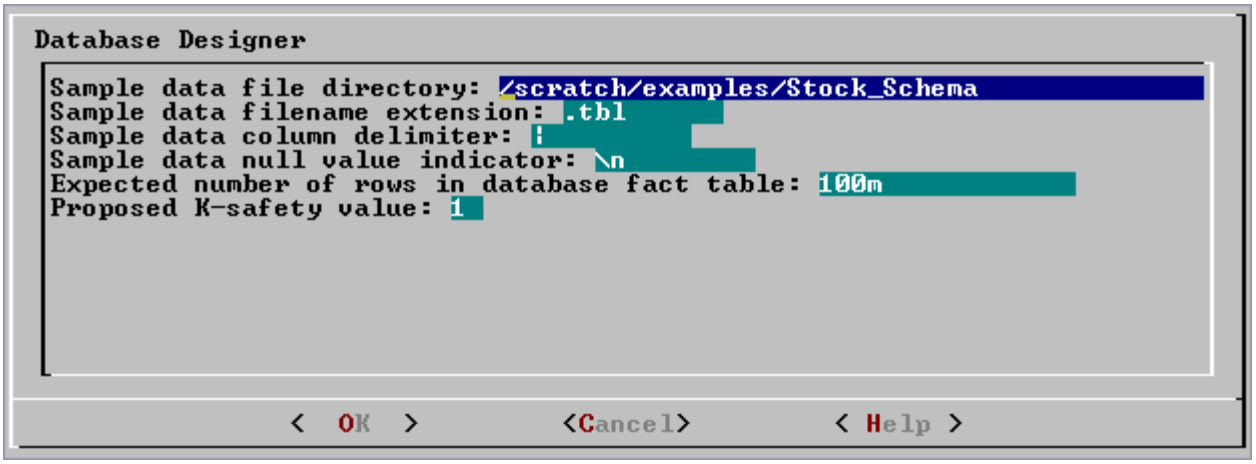
18. Return to the **Administration Tools** (quit vsql).

```
Stock_Schema=> \q
```

19. In the **Configuration** menu, select **Create New Database Design**.
20. In the **Select a database** form, select **Stock_Schema**.
21. In the **Enter directory** form, use the **default pathname** for output and log files.

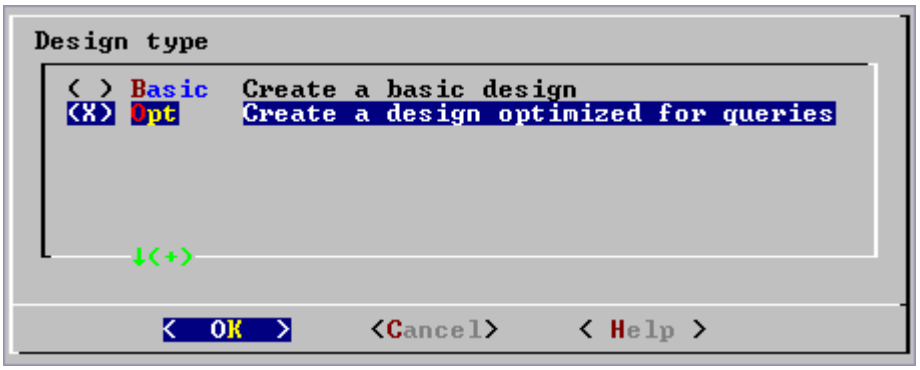


22. In the **Database Designer** form, use the **default parameters**.

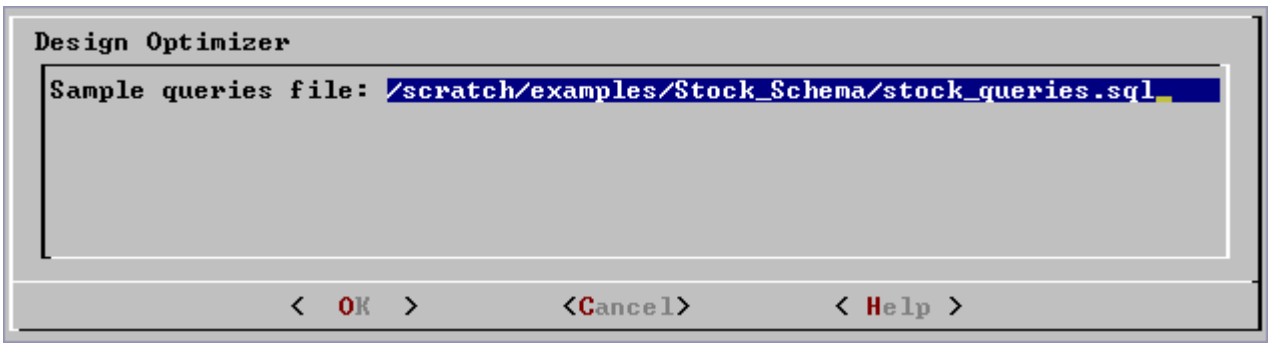


A complete description of the Database Designer dialog is provided in the Database Administrator's Guide.

23. In the **Design type** menu, select **Opt - Create a design optimized for queries**.



24. In the **Design Optimizer** form, enter the **pathname of the sample queries file**.



25. The Database Designer prints status messages to the standard output while it is working. Several minutes may pass before the following **success message** appears.

26. Go back to the **Main Menu** and select Connect to Database.

```
Welcome to the vsql, Vertica_Database v2.0.5-0 interactive terminal.
Type:  \h for help with SQL commands
       \? for help with vsql commands
       \g or terminate with semicolon to execute query
       \q to quit
Stock_Schema=>
```

27. Execute the **SQL script** generated by the Database Designer to create the projections that make up the physical schema.

```
Stock_Schema=> \i Stock_Schema_design_opt_1.sql
CREATE PROJECTION
CREATE PROJECTION
        ⋮
  mark_design_ksafe
----------------------
 Marked design 1-safe
(1 row)
```

28. Execute the SQL script **stock_load_data.sql** to load the example data.

```
Stock_Schema=> \i stock_load_data.sql
COPY
   COPY
  ⋮
```

It may take several minutes to load the default five-million row fact table on a typical hardware cluster. You can watch the load by examining the vertica.log file as described in the Monitoring the Log Files section of the Database Administrator's Guide.

29. Proceed to ***Running Simple Queries*** (page 27).

Running Simple Queries

Each example database includes several files containing SQL queries that are intended to represent queries that might be used 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 SQL Programmer's Guide.

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

1. In the **Administration Tools** select Connect to Database.

```
Welcome to the vsql, Vertica_Database v2.0.5-0 interactive terminal.
Type:  \h for help with SQL commands
       \? for help with vsql commands
       \g or terminate with semicolon to execute query
       \q to quit
Stock_Schema=>
```

2. Use the \i meta-command to **execute the query script**. For example:

```
Stock_Schema=> \i stock_query_01.sql
```

Clickstream Example Database

clickstream_query_01.sql (on page 38)

clickstream_query_02.sql (on page 38)

clickstream_query_03.sql (on page 39)

clickstream_query_04.sql (on page 39)

clickstream_query_05.sql (on page 40)

Credit History Example Database

credithistory_query_01.sql (on page 45)

credithistory_query_02.sql (on page 45)

credithistory_query_03.sql (on page 46)

credithistory_query_04.sql (on page 47)

credithistory_query_05.sql (on page 47)

Stock Exchange Example Database

stock_query_01 (on page 67)

Retail Sales Example Database

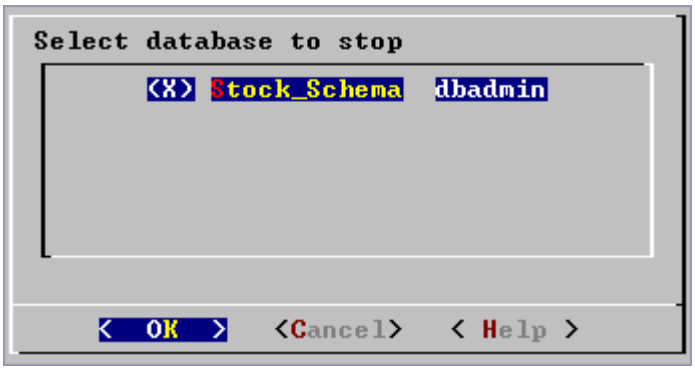
Telecom Example Database

Cleanup Procedure

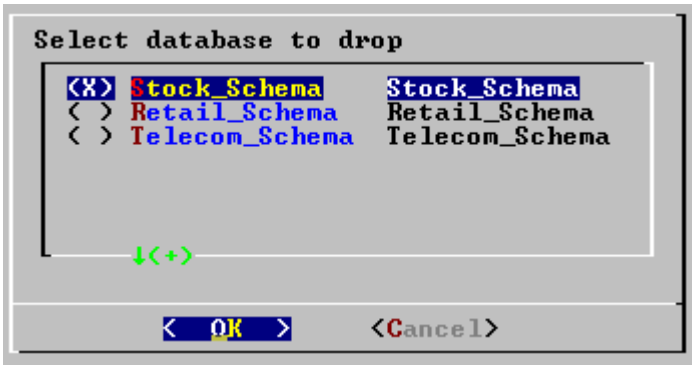
For complete descriptions of each dialog, refer to the Administration Tools Reference in the Database Administrator's Guide.

1. **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**.



4. Go to **Configuration** and select **Drop Database** for all existing databases.



5. Perform the steps in the Uninstalling the Vertica Software section of the Installation Guide.
6. Optionally remove the dbadmin account on all cluster hosts.
7. Remove any example database directories you created.

Example Databases

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

- ***Clickstream Example Database*** (page 33)
- ***Credit History Example Database*** (page 41)
- ***Retail Sales Example Database*** (page 49)
- ***Stock Exchange Example Database*** (page 59)
- ***Telecom Example Database*** (page 71)

You can perform this tutorial using any or all of the example databases.

> You can define multiple example databases within a single Vertica installation but Vertica Systems, Inc. strongly recommends that you start only one example database at a time. Otherwise, the results are unpredictable.

Example Database File Locations

The example databases are installed in:

```
/opt/vertica/doc/ClickStream_Schema
/opt/vertica/doc/CreditHistory_Schema
/opt/vertica/doc/Retail_Schema
/opt/vertica/doc/Stock_Schema
/opt/vertica/doc/Telecom_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:

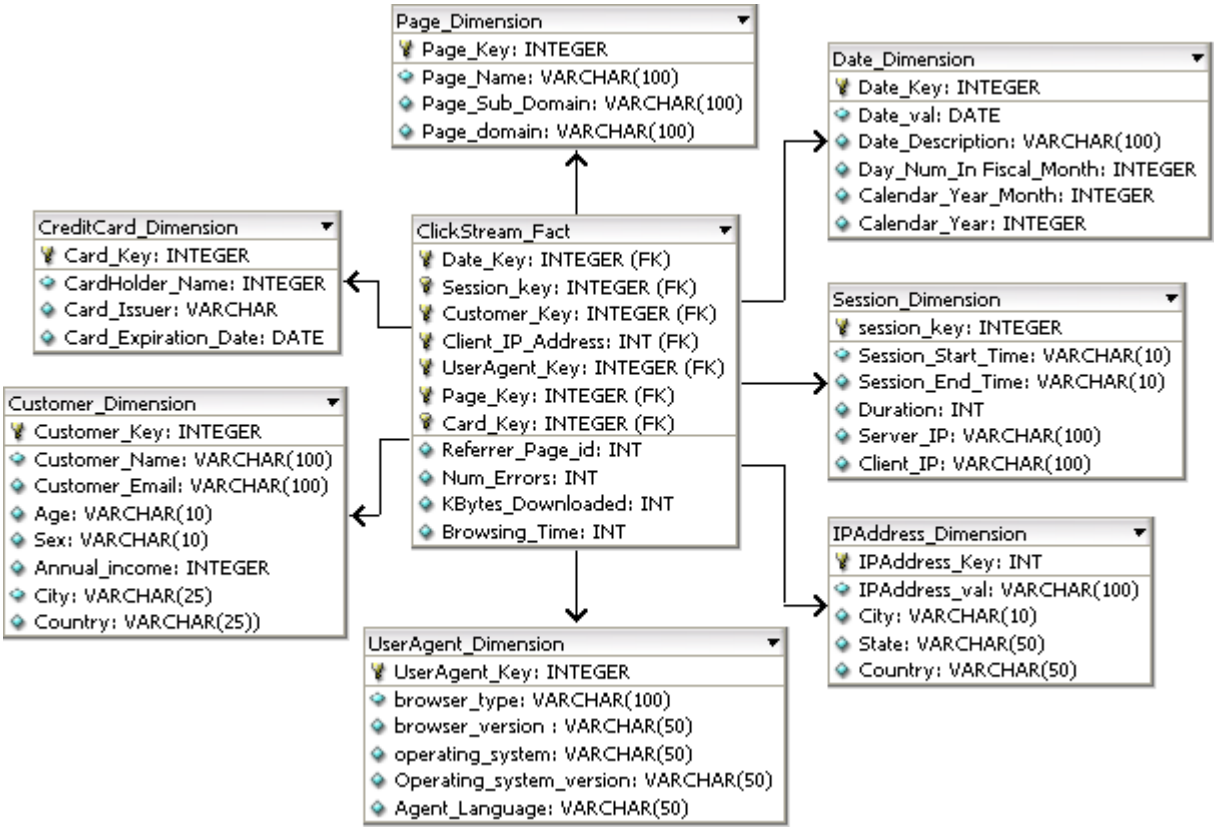
```
clickstream
credithistory
retail
stock
telecom
```

*example*_count_data.sql	SQL script that does 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 precomputed 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 website. 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 offline 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 will help 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 34)	5000000
Customer_Dimension (on page 35)	5000
Session_Dimension (on page 37)	50000
UserAgent_Dimension (on page 37)	500
IPAddress _Dimension (on page 36)	1000
Page_Dimension (on page 37)	5000
CreditCard_Dimension (on page 35)	5000

ClickStream_Fact

Each tuple 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
Email_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 e.g. 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	Primary Key
CardHolder_Name	Varchar	Varchar
Card_Type	Varchar	MasterCard/Visa/Amex
Card_Expiration_Date	Date	Date

Date_Dimension

The Date Dimension 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 e.g. January 1, 2000
Day_Num_In_Fiscal_Month	Integer	The day number in the month (1-31) e.g. 21 for 21st of any month.
Calendar_Year_Month	Integer	Calendar Month of the Date (1-12) e.g. 9 for September
Calendar_Year	Integer	Calendar year of the Date e.g. 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 e.g. 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
```

Credit History Example Database

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

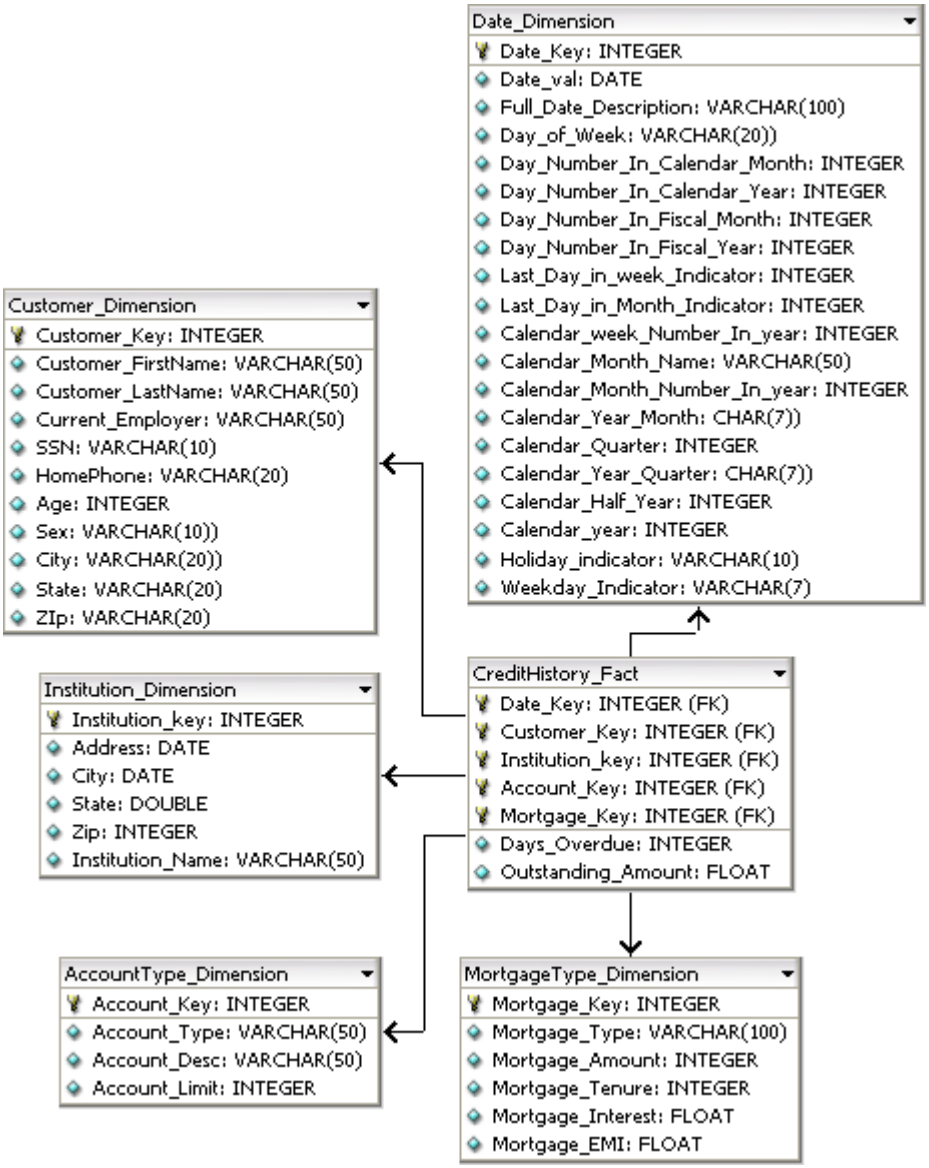


Table Name	Default Number of Rows
CreditHistory_Fact (on page 42)	5000000
Customer_Dimension (on page 42)	5000
Institution_Dimension (on	100

page 44)	
AccountType_Dimension (on page 42)	50
MortgageType_Dimension (on page 44)	1000

CreditHistory_Fact

Each tuple in the fact table represents a credit transaction done 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 Sate
Zip	Varchar	Zip code

Date_Dimension

The Date Dimension 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
Full_Date_Description	varchar(18)	Description of the Date e.g. January 1, 2000
Day_of_Week	varchar(9)	Calendar year of the Date e.g. 2001
Day_Number_in_Calendar_Month	integer	Calendar Month of the Date (1-12) e.g. 9 for September
Day_Number_in_Calendar_Year	integer	The day number in the month (1-31) e.g. 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, Like $1000, $10000 etc.
Mortgage_Tenure	Integer	Mortgage Tenure in Months like 12, 24, 36 etc
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
```

Retail Sales Example Database

The Retail Sales Example Database is based on an entirely 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.

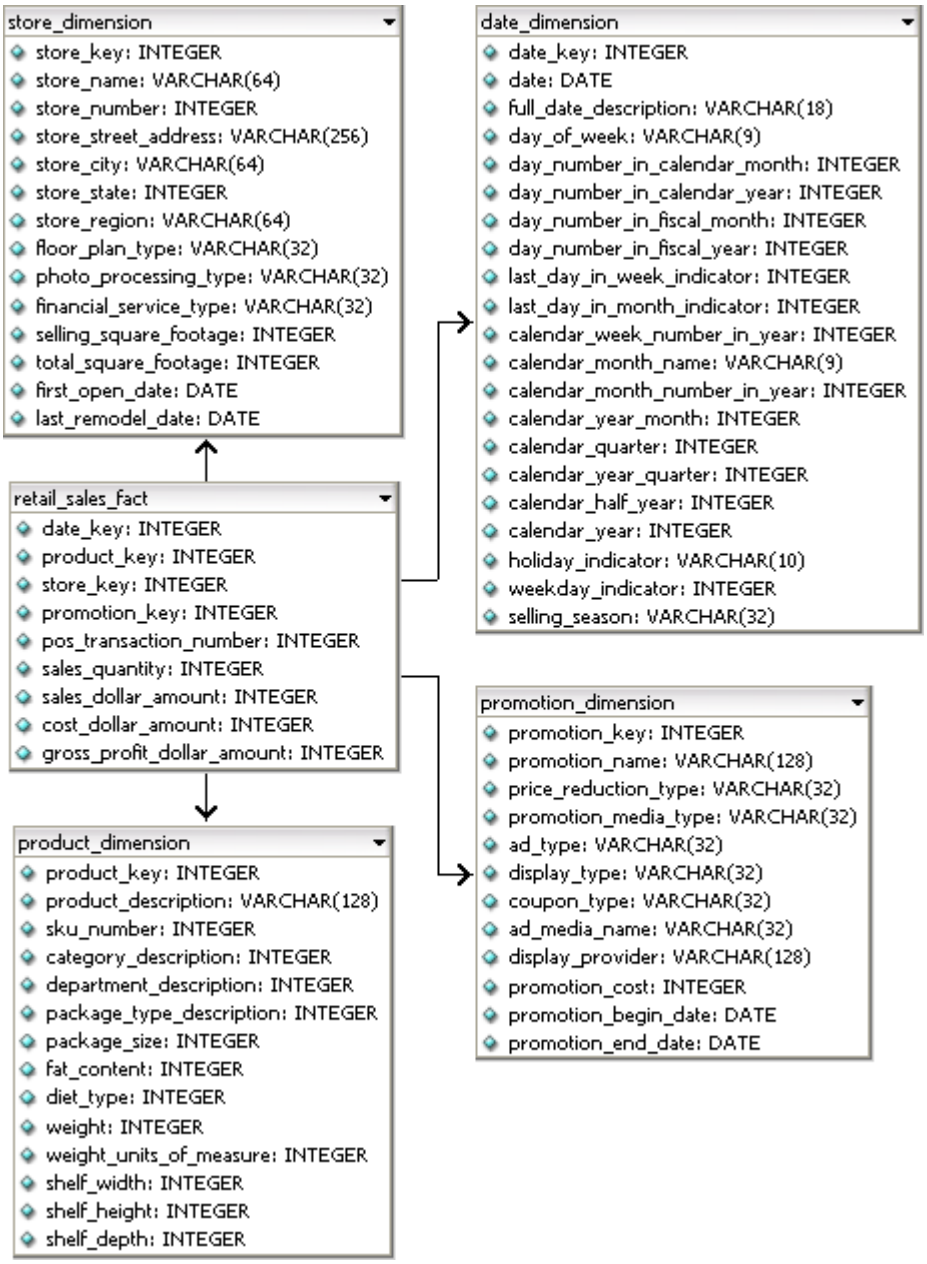


Table Name	Default Number of Rows
Retail_Sales_Fact (on page 50)	5000000
Product_Dimension (on page 51)	60000
Store_Dimension (on page 52)	250
Promotion_Dimension (on page 51)	1000

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)	ValentinesDay

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	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 ox
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	Example
Promotion_Key	integer	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_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
2000|January|West|Winter Cool Sale|88196
2000|December|West|Summer Cool Sale|88191
2002|October|West|July 4th Discount Sale|88081
2000|October|East|Summer Liquidation Promotion|87857
2002|October|West|Summer Liquidation Promotion|87836
2004|January|West|Winter Cool Sale|87751
2001|January|West|July 4th Discount Sale|87743
2003|March|West|July 4th Discount Sale|87475
2002|October|West|Thanksgiving Super Sellathon|87349
2002|June|West|Thanksgiving Super Sellathon|86971
2000|January|West|July 4th Discount Sale|86742
2000|January|West|Christmas Mega Promotion|86585
2003|March|West|Thanksgiving Discount Sellathon|86559
2003|March|West|Winter Cool Sale|86359
2000|December|West|July 4th Cool Sellathon|86148
2002|August|West|Thanksgiving Super Sellathon|85995
2004|January|West|July 4th Cool Sellathon|85840
2001|August|West|Summer Cool Sale|85612
2004|May|West|Thanksgiving Super Sellathon|85199
2003|December|West|July 4th Discount Sale|85069
2000|January|West|Thanksgiving Discount Sellathon|84933
2001|March|West|Thanksgiving Super Sellathon|84698
2001|January|West|Thanksgiving Super Sellathon|84629
2004|January|West|July 4th Discount Sale|84598
2002|October|West|July 4th Super Sale|84324
2000|October|West|July 4th Cool Sellathon|84106
2000|December|West|July 4th Discount Sale|83998
2000|January|East|Thanksgiving Super Sellathon|83824
2003|May|West|Thanksgiving Super Sellathon|83669
```

```
2003|May|West|Summer Cool Sale|83669
2004|June|West|Thanksgiving Super Sellathon|83584
2001|May|West|July 4th Discount Sale|83183
2002|October|West|Winter Cool Sale|83147
2000|July|West|Summer Cool Sale|82945
2002|May|West|Summer Liquidation Promotion|82845
2004|January|West|Thanksgiving Liquidation Promotion|82775
2000|January|West|Winter Mega Sale|82716
2004|January|West|Summer Mega Sellathon|82450
2000|December|West|July 4th Super Sale|82383
2000|December|West|Christmas Mega Promotion|82213
2003|May|West|Summer Liquidation Promotion|82160
2004|August|West|Thanksgiving Super Sellathon|82120
2002|October|West|Summer Cool Sale|82047
2000|January|West|Christmas Cool Promotion|81943
2004|January|East|July 4th Super Sale|81908
2004|January|West|July 4th Super Sale|81845
2003|March|West|July 4th Super Sale|81772
2004|August|West|Summer Liquidation Promotion|81650
2000|July|West|Winter Cool Sale|81510
2000|October|West|Winter Cool Sale|81444
2003|March|West|July 4th Cool Sellathon|81417
2001|August|West|July 4th Cool Sellathon|81341
2004|November|West|Thanksgiving Super Sellathon|81327
2000|January|East|Summer Cool Sale|81240
2001|March|West|Summer Liquidation Promotion|81190
2000|October|West|Thanksgiving Discount Sellathon|81172
2001|May|West|July 4th Cool Sellathon|81168
2004|June|West|July 4th Cool Sellathon|81037
2003|February|West|July 4th Discount Sale|80991
2003|December|West|July 4th Super Sale|80985
2003|February|West|Thanksgiving Super Sellathon|80969
2002|October|West|July 4th Cool Sellathon|80917
2000|January|West|Summer Discount Promotion|80890
2002|August|West|Summer Liquidation Promotion|80862
2003|June|West|Winter Cool Sale|80813
2002|December|West|Thanksgiving Super Sellathon|80761
2003|October|West|July 4th Discount Sale|80761
2003|December|West|Summer Liquidation Promotion|80724
2003|December|West|July 4th Discount Sellathon|80719
2003|August|West|July 4th Discount Sale|80700
2000|January|East|July 4th Cool Sellathon|80698
2000|December|East|Thanksgiving Super Sellathon|80656
2004|October|West|Winter Cool Sale|80616
2000|October|West|July 4th Super Sale|80555
2000|November|West|Thanksgiving Super Sellathon|80445
2003|May|West|July 4th Cool Sellathon|80401
2000|December|West|Winter Cool Sale|80394
2003|October|West|July 4th Super Sale|80346
2004|January|East|Summer Liquidation Promotion|80244
2001|January|West|July 4th Cool Sellathon|80191
2004|January|West|Christmas Cool Promotion|80134
2004|January|East|Thanksgiving Super Sellathon|80094
(101 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

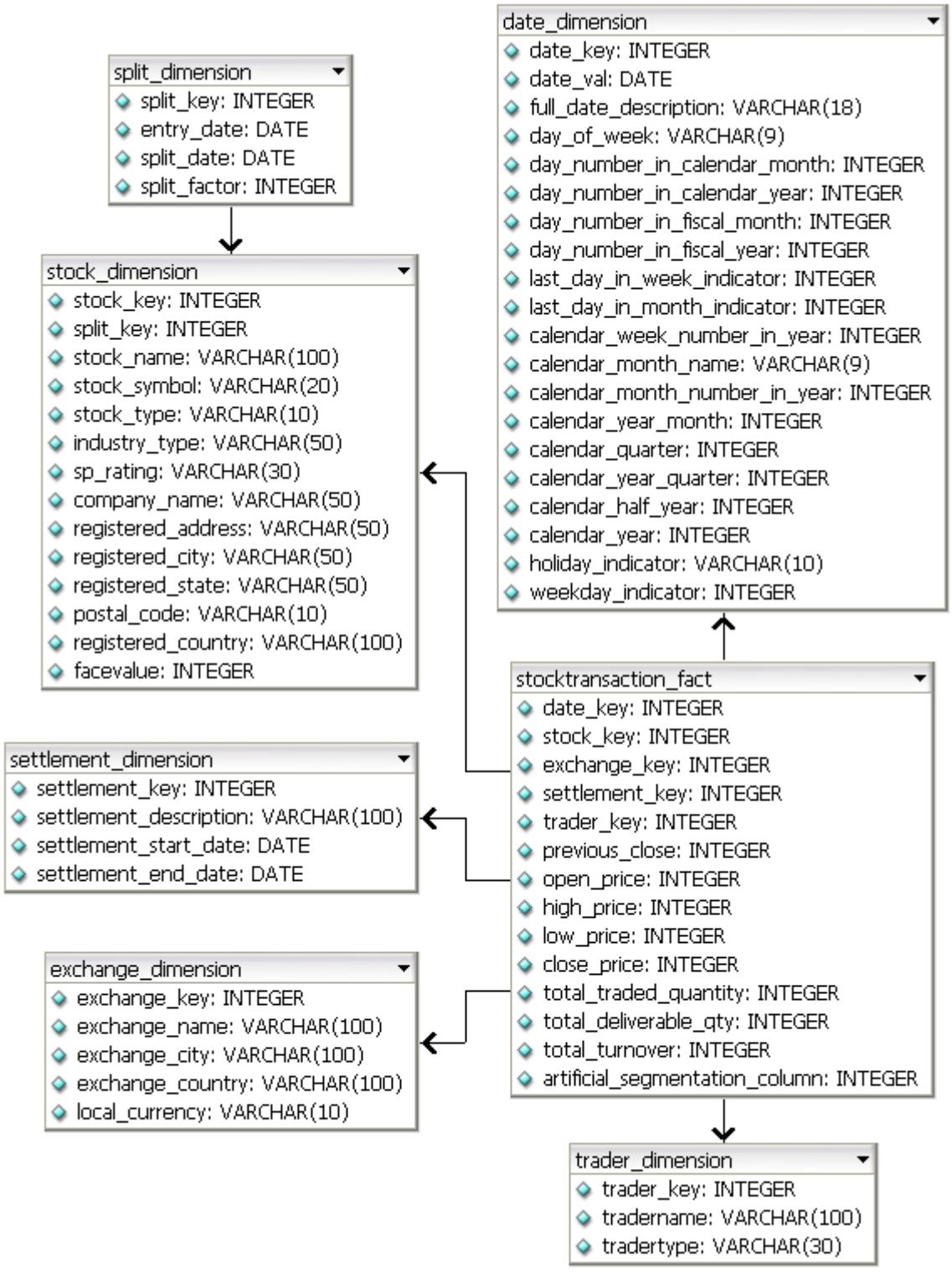
```
Retail_Single_Node=> \i retail_query_03.sql
-------------------------------------------------------

  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
```

Stock Exchange Example Database

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 etc. It is commonly known as a "tick store." Each table is described in a separate section.


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
split_dimension
split_key: INTEGER
entry_date: DATE
split_date: DATE
split_factor: 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 tuple 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

This table 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 Compnay
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 e.g. $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;
```

stock_query_02

Query

```
-------------------------------------------------------------------------
---                    QUERY #2
--- Stocks with maximum traded quantity and value 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;
```

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;
```

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;
```

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;
```

stock_query_06

Query

```
-------------------------------------------------------------------
---             Query 06
--- Get the closing price of a set of 10 stocks for a 10-year period
--- and group into weekly, monthly and yearly aggregates. For each
--- aggregate period determine the low, high and average closing price
--- value. Sort the output by id and trade date.
-------------------------------------------------------------------
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;
```

stock_query_07

Query

```
-------------------------------------------------------------------------
-----                  Query # 07
-- For each stock in a specified list of 1000 stocks, find the
-- differences between the daily high and daily low on the day of
-- each split event during a specified period.
-------------------------------------------------------------------------
SELECT A.High_Price - A.Low_Price,
       A.Close_Price,
       C.Split_Date
FROM     StockTransaction_Fact A,
         Stock_Dimension B,
         Split_Dimension C,
         Date_Dimension D
WHERE    A.Stock_Key = B.Stock_Key
         AND B.split_Key = C.Split_Key
         AND A.Date_Key = D.Date_Key
         AND D.Calendar_Year = 2002
         AND D.Calendar_Week_Number_in_Year = 10
         AND D.Day_of_Week = 'Monday'
ORDER BY A.Stock_Key;
```

Telecom Example Database

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.

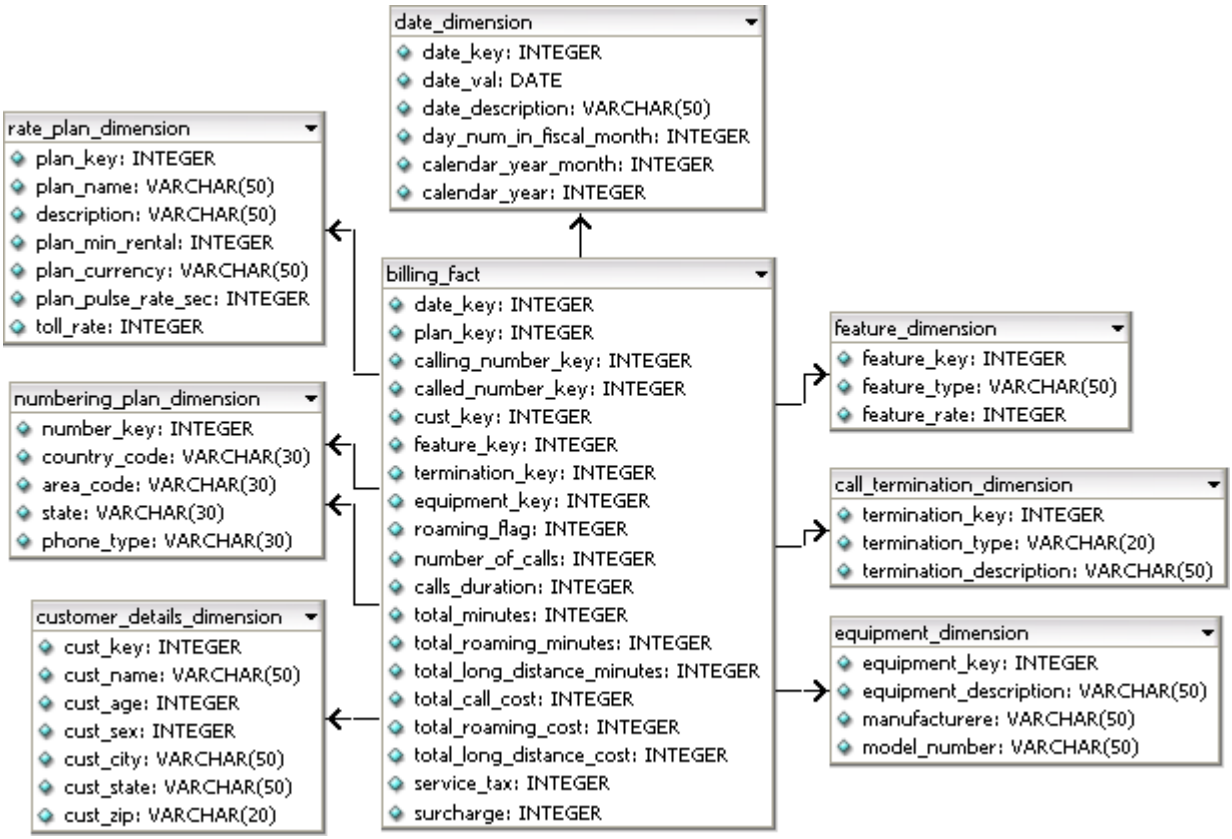


Table Name	Default Number of Rows
Billing_Fact (on page 72)	5000000
Customer_Details_Dimension (on page 73)	50000
Rate_Plan_Dimension (on page 75)	500
Numbering_Plan_Dimension (on page 74)	500
Equipment_Dimension (on page 74)	200
Feature_Dimension (on page 74)	20

Call_Termination_Dimension (on page 73)	20

Billing_Fact

Each tuple in the fact table represents a summary of the CDR records generated at tje 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 will be 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 i.e. normal or abnormal
Equipment_Key	Integer	Type of Equipment
Roaming_Flag	Bool	Whether this call is made/recd 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 e.g. January 1,

		2000
Calendar_Year	Integer	Calendar year of the Date e.g. 2001
Calendar_Year_Month	Integer	Calendar Month of the Date (1-12) e.g. 9 for September
Day_Num_in_Fiscal_Month	Integer	The day number in the month (1-31) e.g. 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)
```

Generating 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.

Syntax

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

Semantics

Parameter	Description

example	is one of the following:
	`clickstream`
	`credithistory`
	`retail`
	`stock`
	`telecom`

`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`*	the seed for the pseudo-random number generator. If you use the same seed each time you run the data generator, you will get the same data files (excluding external factors). Default: **20177**
`time_file` *`pathname`*	the pathname of the pre-computed time data input file used to generate the ***Date Dimension*** (see "Date_Dimension" on page 50) table. Default: **./Time.txt** (supplied by Vertica; contains data for the years 2000-2004).
`fact_table_name rows`	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`	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 \
   --time_file /home/dbadmin/Retail_Schema/Time.txt \
   --retail_sales_fact 100000 \
   --product_dimension 500 \
   --store_dimension 50 \
   --promotion_dimension 100
```

Using the Graphical User Interface

This is only a quick reference. It is not a complete guide to keystroke usage.

Return	Execute selected command.
Tab	Move cursor from OK to Cancel to Help to menu or form to OK...
Up/Down Arrow	Move cursor up and down in menu, form, 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 Cygwin Bash Shell running on Windows XP. If you are using a remote terminal application such as PuTTY or a Cygwin bash shell, make sure your window is at least:

23 characters wide and 81 characters high

If you are using PuTTY, you can make the Administration Tools look like the screen captures in this document:

1. In the PuTTY Configuration dialog, create or load a saved session.
2. Click Category: **Window** > **Appearance**.
3. In the **Font settings**, click the **Change...** button.
4. Select Font: **Terminal** Font Style: **Regular** Size: **10**
5. Click **OK**.
6. Click Category: **Session**
7. Click **Save**

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

Index

T

U

V

W