Vertica Database 2.0.5-0

Glossary of Terms

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 .

Preface

This document describes the terms used in the Vertica documentation set..

Prerequsites

This document assumes that you are familiar with standard relational database terminology.

Audience

This document is intended for all users of Vertica.

Glossary of Terms

Please note that some of the HTML cross-reference links in this document jump to the target term rather than displaying it in an expanding block, as is common in the rest of the Vertica documentation. This is due to a limitation in the HTML generator program, which does not handle "circular references" correctly.

Administration Host

The ***host*** (page 25) on which the Vertica ***rpm*** (on page 30) package was manually installed. Always run the ***Administration Tools*** (page 19) on this host if possible.

Administration Tools

The tools needed for administering a Vertica ***database*** (page 22) are provided in the form of a graphical user interface that allows you to perform various tasks quickly and easily. The tools also provide a convenient way to connect to a database using ***vsql*** (on page 34). Always run the Administration Tools on the ***Administration Host*** (page 19) if possible.

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

See the Administration Tools Reference for a complete description.

Anchor Table

The anchor table of a join can be:

- the ***fact table*** (page 24) in a ***star schema*** (page 32)
- the central fact table in a ***snowflake schema*** (page 32)

- a ***dimension table*** (page 23) in a snowflake schema that functions as a fact table (with a restriction)

ATM

An abbreviation that provides an alternative name for the (Automatic) ***Tuple Mover*** (on page 33).

Authentication

Authentication is the process of attempting to verify the identity of a user attempting to connect to a database.

Authorization

Authorization is the process of verifying that a user has permission to perform a certain operation, such as query a specific table.

Blade Server

Blade servers consist of some number of small servers, called blades, in a common chassis that allows blades to share common components such as power supplies, cooling fans, and switching capabilities.

Buddy Projection

Buddy projections are required for ***K-Safety*** (page 26). Two ***projections*** (page 28) are considered to be buddies if they contain the same columns and have the same ***range segmentation*** (page 29) or ***hash segmentation*** (page 25) using different node ordering. Buddy projections usually have different sort orders for query performance purposes.

Bulk loading

A process of loading large amount of data, such as an initial load of historic data.

C-Store

A research project at MIT, Brandeis, Brown, and UMass (Boston) on which Vertica is based.

Catalog

In Vertica, the catalog is a set of files that contain information (metadata) about the objects in a ***database*** (page 22) (nodes, tables, constraints, projections, etc.) The catalog is replicated on all ***nodes*** (page 27) in a ***cluster*** (page 21).

Cluster

A cluster generally refers a collection of ***hosts*** (page 25) or a collection of ***nodes*** (page 27) bound to a ***database*** (page 22). A cluster is not part of a database definition and thus does not have a name.

Compression

Compression is the process of transforming data into a more compact format. Compressed data cannot be directly processed; it must first be decompressed. Vertica uses integer packing for unencoded integers and ***LZO*** (on page 26) for compressible data. Although compression is generally considered to be a form of ***encoding*** (page 24), the terms have different meanings in Vertica.

Connection

A SQL connection is an occurrence of an interactive user or an application program requesting and being granted access to a ***database*** (page 22) via a network connection. In Vertica, the scope of a connection is the same as that of a ***session*** (page 31).

Data Warehouse

A data warehouse is a relational database that is designed for query and analysis rather than transaction processing. Data warehouses:

- are often subjected to a heavy load of periodic and ad-hoc queries.
- contain historical information that enables analysis of correlations and trends over long periods of time.
- integrate data from various production (transactional) databases. Extraction, transformation, and loading (***ETL*** (on page 24)) software converts the data to a common format and copies it into a data warehouse at regular intervals.
- typically consist of one or more star or snowflake schemas.

Database

A database is a ***cluster*** (on page 21) of ***nodes*** (page 27) that, when active, can perform distributed data storage and SQL statement execution through administrative, interactive, and programmatic user interfaces.

Database Administrator

The database administrator (DBA) is the Linux user account that owns the database catalog and data storage on disk. The DBA can bypass all database authorization rules. However, the DBA must supply a password to connect to a running database and to use Administration Tools commands that affect a running database. The DBA can drop a stopped database without supplying a password.

Database Designer

The Database Designer is a tool that analyzes a logical schema definition, sample queries, and sample data and generates a set of ***projections*** (page 28) in the form of an SQL script to be executed after you create the tables but before you load any data. The script creates a minimal set of ***superprojections*** (see "Superprojection" on page 33) to ensure ***K-Safety*** (page 26), and optionally ***pre-join*** (page 28) projections. In most cases, the projections created by the Database Designer provide excellent query performance within physical constraints. You can, however, write a custom projection script should the Database Designer not meet your needs.

Database Superuser

The database superuser is the automatically-created database user who has the same name as the Linux ***database administrator*** (page 22) account and who can bypass all GRANT/REVOKE authorization. Superuser status cannot be granted to another user. (The concept of a database superuser should not be confused with Linux superuser (root) privilege. In fact, a database superuser cannot have Linux superuser privilege.)

DBA

The database administrator (DBA) is the Linux user account that owns the database catalog and data storage on disk. The DBA can bypass all database authorization rules. However, the DBA must supply a password to connect to a running database and to use Administration Tools commands that affect a running database. The DBA can drop a stopped database without supplying a password.

DDL

SQL Data Description (or Definition) Language generally consists of CREATE, ALTER, and DROP commands, which operate on the ***Logical Schema*** (page 26) metadata (tables, users, etc.)

DELTAVAL (Delta Encoding)

DELTAVAL (Delta Encoding) stores only the differences between sequential data values rather than the values themselves.

Dialog

Dialog is a Linux utility that creates user interfaces to shell scripts or other scripting languages, such as perl. In Vertica V2.0, Dialog is used to implement the Administration Tools, which run in a terminal window.

Dimension Table

A dimension table (sometimes called a lookup or reference table) is one of a set of companion tables to a ***fact table*** (page 24) in a ***star schema*** (page 32). It contains the actual data corresponding to the foreign keys in a fact table. For example, a business might use a dimension table to contain item codes and descriptions.

Dimension tables can be connected to other dimension tables to form a hierarchy of dimensions in a ***snowflake schema*** (page 32).

DML

SQL Data Manipulation Language generally consists of INSERT, UPDATE, and DELETE commands, which modify existing data (single ***tuples*** (see "Tuple" on page 33) or sets of tuples) in the ***Logical Schema*** (page 26).

Dynamic SQL

Dynamic SQL is a programmatic interface to a database management system that allows SQL statements to be defined and executed at run time, usually based on user input.

Encoding

Encoding is the process of transforming data from one format into another. In Vertica, encoded data can be processed directly, which distinguishes it from ***compression*** (page 21). Vertica uses a number of different encoding strategies, depending on column data type, table cardinality, and sort order.

Epoch

An epoch represents committed changes to the data stored in a ***database*** (page 22) between two specific points in time. In other words, an epoch contains all COPY, INSERT, UPDATE, and DELETE operations that have been executed and committed since the end of the previous epoch.

ETL

ETL (Extract, Transform, Load) is a process in data warehousing that involves extracting data from outside sources, transforming it to fit a specific schema, and ultimately loading it into the database.

Fact Table

A fact table is a table that represents quantitative or factual data. For example, in a business, a fact table might represent orders.

A fact table is often located at the center of a ***star schema*** (page 32) or ***snowflake schema*** (page 32). It typically has a large number of tuples and is surrounded by a collection of ***dimension tables*** (page 23), each with a lesser number of tuples. The fact table participates in a join with every dimension table. In other words, a fact table can contain data but generally contains many foreign keys, each of which matches the primary key of each in a dimension table.

Functional Dependency

A functional dependency is a relationship between two sets of column values in a table where one column value can determine the other. It represents knowledge of the data that cannot otherwise be expressed in the logical schema. In Vertica functional dependencies are expressed as CORRELATION constraints and are used by the ***Database Designer*** (page 22) to produce more efficient ***physical schema*** (page 27) designs.

Grid Computing

Grid computing is a software environment based on open standards and protocols that make it possible to share disparate, loosely coupled resources across organizations and geographies. Resources can potentially include almost any component: computer cycles, storage spaces, databases, applications, files, sensors or scientific instruments.

Hash Function

A hash function is a reproducible method of converting data into a number that can serve as a digital "fingerprint" of the data. In Vertica, the built-in HASH function is used in ***segmentation*** (page 31) to evenly distribute data over a ***cluster*** (page 21) of nodes.

Hash Segmentation

Hash segmentation allows you to segment a ***projection*** (page 28) based on a built-in ***hash function*** (page 25) that provides even distribution of data across some or all of the ***nodes*** (page 27) in a ***cluster*** (page 21), resulting in optimal query execution.

Historical Query

Vertica can execute a query from a snapshot of the database taken at a specific date and time. The syntax is:

```
AT TIME 'timestamp' SELECT...
```

The command queries all data in the database up to and including the ***epoch*** (on page 24) representing the specified date and time without holding a lock or blocking write operations.

Host

A host is a computer system with a 32-bit or 64-bit Intel or AMD processor, RAM, hard disk, and TCP/IP network interface (IP address and hostname). Hosts share neither disk space nor main memory with each other.

Instance

An instance of Vertica consists of the running Vertica process and disk storage (catalog and data) on a ***host*** (page 25). There can be only one instance of Vertica running on a host at any time.

JDBC

JDBC (Java Database Connectivity) is a call-level API that provides connectivity between Java programs and data sources (SQL databases and other non-relational data sources, such as spreadsheets or flat files). JDBC is included in the Java 2 standard and enterprise editions.

K-Safety

K represents the maximum number of ***nodes*** (page 27) in a ***database*** (page 22) that can fail and ***recover*** (page 29) with no loss of data. In Vertica V2.0, the value of K can be zero (0) or one (1). The value of K can be one (1) only when the ***Physical Schema*** (on page 27) design meets certain requirements. The designs generated by the ***Database Designer*** (page 22) are K-Safe.

Logical Schema

A logical schema consists of a set of tables and referential integrity constraints in a Vertica database. The objects in the logical schema are visible to SQL users. The logical schema does not include ***projections*** (page 28), which make up the ***physical schema*** (page 27).

LZO

LZO is an abbreviation for ***Lempel-Ziv-Oberhumer*** (http://www.oberhumer.com). It is a data compression algorithm that is focused on decompression speed. The algorithm is lossless and the reference implementation is thread safe.

Materialized View

A materialized view is similar to a standard SQL view with one major exception: the data is actually stored on disk rather than computed each time the view is used in a query. A materialized view, then, must be refreshed whenever the data in the underlying tables is changed. A ***projection*** (page 28) is a special case of a materialized view.

Mergeout

Mergeout is the process of consolidating ***ROS containers*** (see "ROS Container" on page 30).

Moveout

Moveout is the process of moving data from the ***WOS (Write Optimized Store)*** (on page 34) to the ***ROS (Read Optimized Store)*** (on page 30). It is performed by the ***Tuple Mover*** (on page 33).

Node

A node is a ***host*** (page 25) configured to run an ***instance*** (on page 25) of Vertica. It is a member of a database ***cluster*** (on page 21) (see ***Node Definition*** (page 27)). For a database to have the ability to recover from the failure of a node requires at least three nodes. Vertica Systems, Inc. recommends that you use a minimum of four nodes.

Node Definition

A node definition is a metadata object that binds a ***host*** (page 25) to a ***database*** (page 22). A node definition contains a symbolic ***node*** (page 27) name that is used to specify ***segmentation*** (on page 31) in ***projections*** (page 28).

ODBC

Open DataBase Connectivity is a standard application programming interface (API) for access to database management systems.

Out-of-Date

A ***projection*** (page 28) is out-of-date if it requires a ***refresh*** (page 29) in order to participate in query execution.

Physical Schema

A physical schema consists of a set of ***projections*** (page 28) used to store data on disk. The projections in the physical schema are based on the objects in the ***Logical Schema*** (page 26).

PostgreSQL

PostgreSQL is an open source database manager whose front-end components are used in Vertica. For more information, see the ***PostgreSQL*** (http://www.postgresql.org/) documentation.

Pre-Join Projection

A pre-join ***projection*** (page 28) stores the result set obtained by joining a single ***fact table*** (page 24) in the ***logical schema*** (page 26) (the ***anchor table*** (page 19)) with one or more ***dimension tables*** (page 23). The result set is typically sorted for optimal query execution performance. Most ***query-specific projections*** (page 28) are pre-joins.

Projection

A projection is a special case of a ***materialized view*** (on page 26) that provides physical storage for data. A projection can contain some or all of the columns of one or more tables. A projection that contains all of the columns of a table is called a ***superprojection*** (on page 33). A projection that joins one or more tables is called a ***pre-join projection*** (page 28). Most projections are used for ad-hoc query processing and K-safety but it is possible to have ***query-specific*** (page 28) projections.

Projection Set

A projection set is a group of ***buddy projections*** (page 20) that are safe for a given level of ***K-safety*** (page 26). (When K=1 there are two buddies in a set; when K=2 there are three buddies.) A ***projection*** (page 28) must be part of a projection set before it is ***refreshed*** (page 29). Once a projection set is created (by creating buddies) the set is refreshed in a single transaction.

psql

psql is a character-based, interactive, front-end that is part of ***PostgreSQL*** (http://www.postgresql.org/) and used by other database management systems. It allows you to type in SQL statements and see the results. It also provides a number of meta-commands and various shell-like features to facilitate writing scripts and automating a wide variety of tasks. On a Vertica host, psql is actually a symbolic link to ***vsql*** (page 34).

Query Optimizer

The query optimizer is the component that evaluates different strategies for executing a query and picks the best one.

Query-Specific Projection

A query-specific projection is usually a custom ***pre-join projection*** (page 28) designed to provide maximum performance for one or more specific queries. The Database Administrator's Guide (Advanced) describes how to write custom projections.

Range Segmentation

Range segmentation allows you to segment a ***projection*** (page 28) based on a known range of values stored in a specific column chosen to provide even distribution of data across a set of ***nodes*** (page 27), resulting in optimal query execution.

Recovery

Recovery is the process of restoring the ***database*** (page 22) to a fully-functional state after one or more nodes in the system has experienced a software or hardware related failure. Vertica has a unique approach to recovering a node that is based on querying replicas of the data stored on other nodes. For example, a hardware failure may cause a node to lose database objects or to miss changes made to the database (INSERTs, UPDATEs, etc.) while offline. When the node comes back on line, it recovers lost objects and catches up with changes by querying the other nodes.

Referential Integrity

Referential integrity in Vertica consists of a set of constraints (***logical schema*** (page 26) objects) that define primary key / foreign key relationships. In a ***star schema*** (page 32) or ***snowflake schema*** (page 32):

- Each ***dimension table*** (page 23) must have a primary key constraint.
- The ***fact table*** (page 24) must contain a foreign key constraint for each foreign key column that it contains.

Referential integrity constraints are required by Vertica but are not enforced when performing INSERT, UPDATE, DELETE, or COPY operations.

Refresh

A refresh operation ensures that all projections on a node are ***up-to-date*** (page 33) (can participate in query execution). This process may take a long time, depending on how much data is in the table(s).

RLE (Run Length Encoding)

RLE (Run Length Encoding) replaces sequences of the same data values within a column by a single value and a count number.

ROS (Read Optimized Store)

The ROS (Read Optimized Store) is a highly optimized, read-oriented, physical storage structure that is organized by ***projection*** (page 28) and that makes heavy use of ***compression*** (on page 21) and indexing. You can use the COPY...DIRECT and INSERT (with direct hint) statements to load data directly into the ROS.

ROS Container

ROS containers are subsets of the ***Read Optimized Store (ROS)*** (see "ROS (Read Optimized Store)" on page 30) that are created as the result of changes to the data stored within a ***projection*** (page 28) as a result of bulk loads and DML. The ***Tuple Mover*** (on page 33) periodically merges ROS containers in order to maximize performance. A ***segmented*** (see "Segmentation" on page 31) projection can be temporarily stored within several ROS containers on any node at any moment but never fewer than one.

rpm

rpm is a powerful package manager, which can be used to build, install, query, verify, update, and erase individual software packages. A package consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the package. Packages come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

Safe

A ***projection*** (page 28) is safe if it has enough ***buddy projections*** (page 20) for the current ***K-safety*** (page 26) level.

SAN

A SAN (Storage Area Network) is a dedicated hardware/software platform consisting of networked servers that provide access to storage-related resources such as such as disk array controllers. It provides high data transfer speeds similar to those used for internal disk drives (ATA, SCSI, etc.) and is highly scalable.

Schema

Schema has several related meanings in Vertica:

- In SQL statements, a schema refers to named namespace for a ***Logical Schema*** (page 26).

- ***Logical Schema*** (page 26) refers to a set of ***tables*** (page 33) and constraints
- ***Physical Schema*** (page 27) refers to a set of ***projections*** (page 28).
- ***Star Schema*** (page 32) and ***Snowflake Schema*** (page 32)

Secure Shell (SSH)

Secure Shell or SSH is a set of standards and an associated network protocol that establishes a secure TCP/IP data transmission channel between a local and a remote computer. It utilizes strong encryption and authentication to ensure confidentiality, integrity, and authenticity of the transferred data.

SSH is typically used to login to a remote machine and execute commands. Use SSH only between two devices that are both under your own administration, when both devices are trustworthy.

Segmentation

Segmentation is the horizontal partitioning of a ***projection*** (page 28) so that it can be stored on multiple ***nodes*** (page 27). The goal is to distribute physical data storage evenly across a database so that all nodes can participate in query execution. See also:

- ***Hash Segmentation*** (page 25)
- ***Range Segmentation*** (page 29)

Session

A SQL session is an occurrence of a user interacting with a database through the use of SQL statements. A session can be invoked using ***vsql*** (on page 34) or a ***JDBC*** (on page 26) application. In Vertica, the scope of a session is the same as that of a ***connection*** (page 21).

Site

Site is a deprecated term that is used to mean ***node*** (page 27) in some contexts, such as the CREATE PROJECTION statement.

Snapshot Isolation

Vertica can run any SQL query in snapshot isolation mode in order to obtain the fastest possible execution. To be precise, snapshot isolation mode is actually a form of ***historical query*** (on page 25). The syntax is:

```
AT EPOCH LATEST SELECT...
```

The command queries all data in the database *up to but not including the current epoch* (on page 24) without holding a lock or blocking write operations. This may cause the query to miss tuples loaded by other users up to (but no more than) a specific number of minutes before execution.

Snowflake Schema

A snowflake schema is the same as a ***star schema*** (page 32) except that a ***dimension table*** (page 23) can be normalized (hierarchically decomposed) into additional dimension tables. Every dimension table participates in a 1::n join with the ***fact table*** (page 24) or another dimension table. In other words, the primary key of a dimension table is a foreign key in a fact table or another dimension table.

Spread

Spread is an open source toolkit used in Vertica to provide a high performance messaging service that is resilient to network faults. Each running ***node*** (page 27) in a ***database*** (page 22) includes a Spread daemon in addition to a Vertica process. Spread daemons start automatically when a database starts up for the first time.

SQL

SQL (Structured Query Language) is a widely-used, industry standard data definition and data manipulation language for relational databases.

SSH

Secure Shell or SSH is a set of standards and an associated network protocol that establishes a secure TCP/IP data transmission channel between a local and a remote computer. It utilizes strong encryption and authentication to ensure confidentiality, integrity, and authenticity of the transferred data.

SSH is typically used to login to a remote machine and execute commands. Use SSH only between two devices that are both under your own administration, when both devices are trustworthy.

Star Schema

The star schema (sometimes called a star join schema) is the simplest ***data warehouse*** (on page 21) schema. In a star schema design there is a central ***fact table*** (page 24) with a large number of tuples, optionally surrounded by a collection of ***dimension tables*** (page 23), each with a lesser number of tuples. Every dimension table participates in a 1::n join with the fact table. In other words, the primary key of a dimension table is a foreign key in the fact table.

Superprojection

A superprojection is a ***projection*** (page 28) that contains every column of a ***table*** (page 33) in the ***Logical Schema*** (page 26). A table can have multiple superprojections with different sort orders.

Superuser

The database superuser is the automatically-created database user who has the same name as the Linux ***database administrator*** (page 22) account and who can bypass all GRANT/REVOKE authorization. Superuser status cannot be granted to another user. (The concept of a database superuser should not be confused with Linux superuser (root) privilege. In fact, a database superuser cannot have Linux superuser privilege.)

Table

In the relational model, a table (relation) is a set of data elements (values) organized into horizontal rows (tuples) and vertical columns. A table has a specified number of columns but can have any number of rows.

In Vertica, a table is a metadata-only entity referred to in queries. The physical representation of data is a ***projection*** (page 28).

Tuple

A tuple is a collection of data values corresponding conceptually to a row of a table, projection, or result set. A tuple in Vertica does not have the same physical representation as in a traditional RDBMS; it is a virtual entity that may have entirely different storage representations.

Tuple Mover

The tuple mover is the component of Vertica that moves the contents of the ***Write Optimized Store (WOS)*** (see "WOS (Write Optimized Store)" on page 34) into the ***Read Optimized Store (ROS)*** (see "ROS (Read Optimized Store)" on page 30). This data movement is known as a ***moveout*** (page 27). Normally, the tuple mover runs automatically in the background at preset intervals and is referred to as the ***ATM*** (page 20).

Up-To-Date

A ***projection*** (page 28) is up-to-date if it does not require a ***refresh*** (page 29) in order to participate in query execution.

User Agent

A user agent is a client application program used to access resources on networks such as the World Wide Web. User agents include web browsers, search engine crawlers, PDAs, cell phones, and so forth.

UTF-8

UTF-8 (8-bit UCS/Unicode Transformation Format) is a variable-length character encoding for Unicode created by Ken Thompson and Rob Pike. It is able to represent any universal character in the Unicode standard, yet the initial encoding of byte codes and character assignments for UTF-8 is coincident with ASCII (requiring little or no change for software that handles ASCII but preserves other values).

vsql

vsql is the Vertica implementation of ***psql*** (page 28), a character-based, interactive, front-end that is part of ***PostgreSQL*** (http://www.postgresql.org/) and used by other database management systems. It allows you to type in SQL statements and see the results. It also provides a number of meta-commands and various shell-like features to facilitate writing scripts and automating a wide variety of tasks.

WOS (Write Optimized Store)

The WOS (Write Optimized Store) is a memory-resident data structure into which INSERT, UPDATE, DELETE, and COPY (without DIRECT hint) actions are recorded. Like the ***ROS*** (see "ROS (Read Optimized Store)" on page 30), the WOS is arranged by ***projection*** (page 28) but it stores tuples without sorting, ***compression*** (on page 21), or indexing and thus supports very fast load speeds. The WOS organizes data by ***epoch*** (page 24) and holds uncommitted transaction data.

Index

P

Q

R

S

T

U

V

W