Vertica® Analytic Database 4.1, Revision 1

Concepts Guide

Copyright© 2006-2011 Vertica Systems, Inc.

Date of Publication: January 7, 2011

CONFIDENTIAL

Contents

Technical Support 1

About the Documentation 2

Where to Find the Vertica Documentation 2
Reading the Online Documentation 2
Printing Full Books 4
Suggested Reading Paths 4
Where to Find Additional Information 6
Typographical Conventions 7

Preface 9

Welcome to Vertica 10

The Vertica Approach 11

Vertica Components 12

Column Store Architecture with FlexStore 13
Data Encoding and Compression 15
High Availability and Recovery 16
Hybrid Storage Model 18
Physical Architecture 20
Logical Schema 21
Physical Schema 22
How Projections are Created 22
Anatomy of a Projection 23
Projection Concepts 24
Projection Performance 24
Projection Segmentation 26
High Availability Through Projections 27
Projection Naming 28

Database Setup....30
Database Connections....32
The Administration Tools....33
The Database Designer....35
K-Safety....35
Database Security....37
Data Loading and Modification....38
Workload Management....38

SQL Overview **40**

Query Execution **41**

Transactions **42**

Automatic Rollback....43
Savepoints....43
READ COMMITTED Isolation....43
SERIALIZABLE Isolation....45

International Languages and Character Sets **47**

Unicode Character Encoding....47
Locales....47
String Functions....48
Character String Literals....48

Get Started **49**

Glossary **50**

Access rank....50
ACID....50
Administration host....50
Administration Tools....50
AHM....51
Anchor table....51
Ancient History Mark....51
TM....51
Authentication....51
Authorization....51
Blade server....51
Bit....52
Bitstring....52
Byte....52
Buddy projection....52
Bulk loading....52
C-Store....52

CSV 52
Cardinality 52
Case-folded 53
Catalog 53
Catalog path 53
Checkpoint 53
Cluster 53
Cold backup 54
Comprehensive design 54
Compression 54
Connection 54
Current epoch 54
Data path 54
Data warehouse 54
Database 55
Database administrator 55
Database Designer 55
Database superuser 55
DBA 55
DDL 55
DELTAVAL (delta encoding) 55
Derived column 56
Design 56
Deterministic 56
Dialog 56
Dimension table 56
Dirty read 56
DML 56
Dynamic SQL 56
EOF 57
Encoding 57
Epoch 57
Epoch map 57
ETL 57
Executor node 57
Expression 57
External procedure 57
Fact table 57
FIFO 58
Foreign key 58
Flattening (subqueries and views) 58
Full backup 58
General pool 58
Grid computing 58
Grouped ROS 59
Hash function 59
Hash join 59
Hash segmentation 59
Hexadecimal 59
Histogram 59
Historical data 60
Historical query 60
Host 60
Hot backup 60

ICU 60
ISO Week-Year 60
Incremental backup 61
Initiator node 61
Instance 61
Interpolation 61
Immutable (invariant) functions 61
JDBC 61
K-Safety 61
Last epoch 62
Last Good Epoch 62
LGE 62
Locale 62
Logical schema 62
LZO 62
Man-in-the-middle attack 62
Manual recovery 63
Materialized view 63
Mergeout 63
Meta-functions 63
Metadata 63
Moveout 63
NaN 63
Nibble 64
Node 64
Node definition 64
Nondeterministic 64
Non-repeatable read 64
NUL 64
NULL 64
Octal 64
Octet 65
ODBC 65
OLAP 65
OLTP 65
Out-of-date 65
Partition key 65
Partitioning 65
Phantom read 66
Physical schema 66
Physical schema design 66
Pinned projection 66
Pre-join projection 66
Predicate 66
Primary column 66
Primary key 67
Projection 67
Projection set 67
Purge 67
Query cluster level 67
Query optimizer 67
Query-specific design 67
Query-specific projection 68

RDBMS 68
RLE (Run Length Encoding) 68
RM 68
Range segmentation 68
Recovery 68
Referential integrity 68
Refresh 69
Replication 69
Resource pool 69
Resource Manager 69
Rollback 70
ROS (Read Optimized Store) 70
ROS container 70
rpm 70
Safe 70
SAN 71
Savepoint 71
Schema 71
Secure Shell 71
Seed 71
Segmentation 71
Select list 72
Session 72
Site 72
Snapshot isolation 72
Snowflake schema 72
Spread 73
Sort-merge join 73
SQL 73
SSH 73
Stable functions 73
Star schema 73
Standalone resource pool 73
Storage location 74
Strict 74
Superprojection 74
Superuser 74
TM 74
Table 74
Temp location 74
Temp space 74
Transaction 75
Tuple 75
Tuple Mover 75
UDP 75
UTC 75
UTF-8 75
Unary operator 76
Up to date 76
Up-to-date 76
User agent 76
View 76
Volatile functions 76
Volatility 76

vsql....76
Window (analytic) 77
Workload management 77
WOS (Write Optimized Store)....77

Index **79**

Copyright Notice **83**

Technical Support

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

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

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

You can also email verticahelp@vertica.com.

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

About the Documentation

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

Where to Find the Vertica Documentation

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

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

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

See Installing Vertica Documentation in the Installation Guide.

Reading the Online Documentation

Reading the HTML documentation files

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

The Vertica documentation is supported on the following browsers:

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

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

Mozilla Firefox

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

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

Internet Explorer

Use one of the following methods:

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

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

Apple Safari

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

Opera

1. Open a browser window.
2. Position your cursor in the title bar and right click > **Customize** > **Appearance**, click the **Toolbar** tab and select **Main Bar**.
3. Choose one of the following methods to access the documentation:
   - Open a browser window and click Open, navigate to `..\HTML-WEBHELP\index.htm`, and click **Open**.
   - OR drag and drop `index.htm` into the browser window.
   - OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Google Chrome

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

1. Open a browser window.
2. Choose one of the following methods to access the documentation:

   In the address bar, type the location of the `index.htm` file on the server. For example: `file://<servername>//vertica41_doc//HTML/Master/index.htm`
   - OR drag and drop `index.htm` into the browser window.
   - OR press **CTRL+O**, navigate to `index.htm`, and click **Open**.

Notes

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

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

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

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

Printing Full Books

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

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

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

The following list provides links to the PDFs.

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

Suggested Reading Paths

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

All Users

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

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

System Administrators

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

Database Administrators

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

Application Developers

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

Where to Find Additional Information

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

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

Typographical Conventions

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

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

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

	be provided.You do not type the braces: `QUOTES ON`
Backslash `\`	Continuation character used to indicate text that is too long to fit on a single line.
Ellipses `...`	Indicate a repetition of the previous parameter. For example, `option[,...]` means that you can enter multiple, comma-separated options. Note: Showing an ellipses in code examples might also mean that part of the text has been omitted for readability, such as in multi-row result sets.
`Indentation`	Is an attempt to maximize readability; SQL is a free-form language.
`Placeholders`	Items that must be replaced with appropriate identifiers or expressions are shown in italics.
Vertical bar `|`	Is a separator for mutually exclusive items. For example: `[ASC | DESC]` Choose one or neither. You do not type the square brackets.

Preface

This document presents a high-level overview of the Vertica® Analytic Database, an innovative system that represents a vast improvement in performance over products currently available from major DBMS software and appliance vendors. It introduces the basic concepts that get you started in effectively designing, building, operating, and maintaining a Vertica database.

Prerequisites

The Concepts Guide assumes that you are familiar with the basic concepts and terminology of relational database management systems and SQL.

Audience

This guide is intended for all users of Vertica® Analytic Database.

Welcome to Vertica

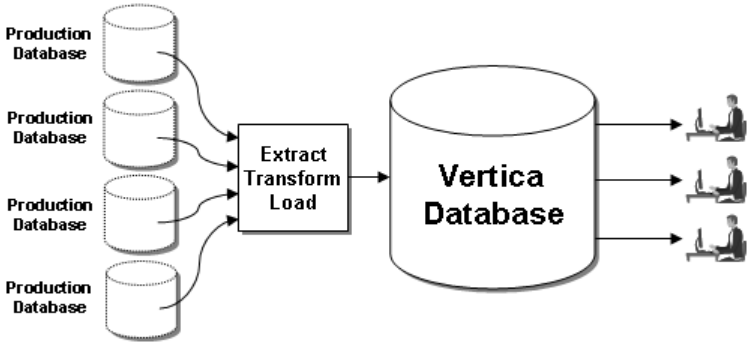
The Vertica® Analytic Database is an innovative, ground-up implementation of a relational database management system optimized for read-intensive workloads. The Vertica® Analytic Database (referred to as Vertica from this point) provides extremely fast ad hoc SQL query performance, even for very large databases, making it well suited for:

- Data warehousing
- Data marts
- Clickstream analysis
- Fraud detection
- Call detail analysis
- Consumer customer analytics
- Business intelligence
- Other query-intensive applications

Data Warehousing

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) 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.



The Vertica Approach

Vertica's primary key differentiators can be listed as the 4 Cs:

- Column storage — Stores data the way it is typically queried for best performance
- Compression — Stores more data, provides more views, and uses less hardware, letting you keep much more historical data in physical storage
- Clustering — Lets you scale out your database cluster infinitely by adding more hardware
- Continuous performance — Queries and loads data 24x7 with virtually no database administration

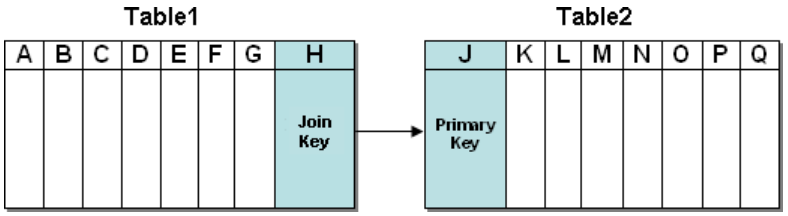
Vertica Components

This section describes the unique components that make up Vertica.

Column Store Architecture with FlexStore

Traditionally databases were designed for ***OLTP*** (on page 65) and used a row-store architecture. To process a query, a row store reads all of the columns in all of the tables named in the query, regardless of how wide the tables might be or how many columns are actually needed. Quite often, analytic queries use only two or three columns from tables containing up to several hundred columns, resulting in a great deal of unnecessary data retrieval.

Unlike other ***RDBMS*** (on page 68), Vertica reads the columns from database objects called ***projections*** (see "***Projection***" on page 67), which are described in the ***Physical Schema*** (page 22) section of this guide. No resources are wasted by reading large numbers of unused columns. Every byte of data is used by the execution engine. For example, consider this simple two-table schema:



Suppose you want to run this query:

```
SELECT A, C, N
FROM Table1 JOIN Table2
ON H = J;
```

A row store must read 16 columns (A through H and J through Q) from physical storage for each record in the result set. A column store with a query-specific projection reads only three columns: A, C, and N.

How FlexStore™ enhances your column-based architecture

FlexStore™ is a combination of physical design, database storage, and query execution techniques that Vertica applies to the database to optimize it for the analytic workload supports at the time. These techniques include:

- **Column grouping.** Multiple columns can be grouped into a single disk file. This grouping minimizes file input/output (I/O) for workloads that read a large percentage of the columns in a table, that do single row look-ups, that query against many small columns, or that frequently update data in these columns. Grouped columns also benefit from special compression and retrieval techniques.

  By Default, Vertica automatically groups all columns together for small loads, such as trickle loads. If you have highly correlated data that is always accessed together and it is not used in predicates, you can increase query performance by grouping these columns. See CREATE PROJECTION in the SQL Reference Manual for details.

- **Intelligent disk use.** Vertica supports the use of multiple storage locations on each node. By default, data is striped across all storage locations based on available disk space such that all locations are uniformly used. However, you can control disk usage and increase I/O performance further by isolating files that have different I/O or access patterns in different storage locations. For example, you can isolate execution engine temporary files from data files and create a tiered-disk architecture, in which columns are stored on different disks, based on predicted or measured access patterns.

  See Creating and Configuring Storage Locations in the Administrator's Guide for details.
- **Fast Deletes.** Vertica can be optimized to enhance delete performance for applications that frequently purge data. See Optimizing Deletes and Updates for Performance and Best Practices for DELETE and UPDATE in the the Administrator's Guide for details.

Data Encoding and Compression

Encoding

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

The query executor in Vertica operates on the encoded data representation whenever possible to avoid the cost of decoding. It also passes encoded values to other operations, saving memory bandwidth. In contrast, row stores and most other column stores typically decode data elements before performing any operation.

Compression

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 for compressible data. Although compression is generally considered to be a form of encoding, the terms have different meanings in Vertica.

The size of a database is often limited by the availability of storage resources. Typically, when a database exceeds its size limitations, the administrator archives data that is older than a specific historical threshold.

The extensive use of compression allows a column store to occupy substantially less storage than a row store. In a column store, every value stored in a column of a projection has the same data type. This greatly facilitates compression, particularly in sorted columns. In a row store, each value of a row can have a different data type, resulting in a much less effective use of compression.

Vertica's efficient storage allows the database administrator to keep much more historical data in physical storage. In other words, the archiving threshold can be set to a much earlier date than in a less efficient store.

High Availability and Recovery

Vertica's unique approach to failure recovery is based on the distributed nature of a database. A Vertica database is said to be ***K-Safe*** (see "***K-Safety***" on page 61) if a node can fail at any given time without causing the database to shut down. When the lost node comes back online and rejoins the database, it recovers its lost objects by querying the other nodes. See Managing Nodes and Monitoring Recovery in the Administrator's Guide.

In Vertica, the value of K can be 0, 1, or 2. If a database that has a K-Safety of one (K=1) loses a node, the database continues to run normally. Increasing K-Safety to 2 allows Vertica to run normally if up to two nodes fail. When the failed node or nodes return and successfully recover, they can participate in database operations again. However if more than 2 nodes fail on a K=2 database, the system is no longer K-Safe and shuts down to prevent data inconsistencies.

If you want the performance of two nodes but are not concerned about K-safety failover, Vertica also supports two-node clusters with no fault tolerance (K=0 safety).

K-Safety Requirements

To determine the number of nodes required to meet your K-Safety needs, use the formula: K<N/2

where :

- K is the K-Safety value.
- N is the number of nodes.

The value of K can be 1 or 2 only when the physical schema design meets certain redundancy requirements. See ***Physical Schema*** (page 22). To create designs that are K-Safe, Vertica recommends that you use the ***Database Designer*** (on page 55).

By default, Vertica creates K-Safe superprojections when the database has a K-Safety greater than 0 (K>0). When creating projections with the Database Designer, projection definitions that meet K-Safe design requirements are recommended and marked with the K-Safety level. Note the output from running the optimized design script generated by the Database Designer in the following example:

```
VMart_Schema=> \i VMart_Schema_design_opt_1.sql
CREATE PROJECTION
CREATE PROJECTION
mark_design_ksafe
----------------------
Marked design 1-safe
(1 row)
```

Determining K-Safety

To determine the K-Safety state of a running database, run the following SQL command:

```
SELECT current_fault_tolerance FROM system;
current_fault_tolerance
----------------
               1
```

```
(1 row)
```

Monitoring K-Safety

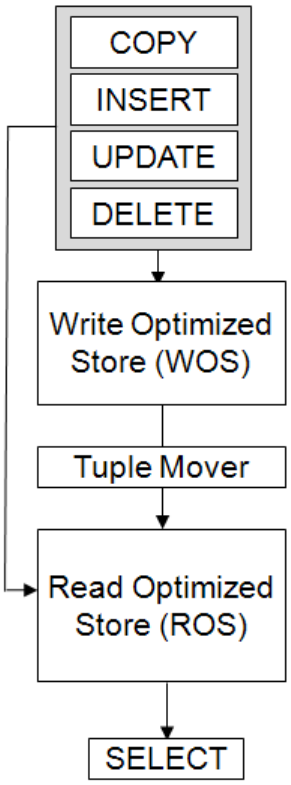
Monitoring tables can be accessed programmatically to enable external actions, such as alerts. You monitor the K-Safety level by polling the SYSTEM table and checking the value. See SYSTEM in the SQL Reference Manual.

Loss of K-Safety

If you lose a node and your cluster's K-Safety level drops to 0, all DML operations remain available; however, performance is affected.

Hybrid Storage Model

To support Data Manipulation Language commands (INSERT, UPDATE, and DELETE) and bulk load operations (COPY) intermixed with queries in a typical data warehouse workload, Vertica implements the storage model shown in the illustration below. This model is the same on each Vertica node.



An abbreviation for Write Optimized Store, WOS is a memory-resident data structure into which INSERT, UPDATE, DELETE, and COPY (without DIRECT hint) actions are recorded. Like the ROS, the WOS is arranged by projection but it stores records without sorting, compression, or indexing and thus supports very fast load speeds. The WOS organizes data by epoch and holds uncommitted transaction data.

Runs in the background, optimizing the data layout. Tasks include moving data from memory (WOS) to disk (ROS), combining small ROS containers into larger ones, and purging deleted data.

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

Like a ROS, a Grouped ROS is a highly-optimized, read-oriented physical storage structure organized by projection that makes heavy use of compression and indexing. However, a Grouped ROS stores data for two or more grouped columns in one disk file.

The COPY command is designed for bulk load operations and can load data into the WOS or the ROS.

Physical Architecture

Terminology

In Vertica, the physical architecture is designed to distribute physical storage and and to allow parallel query execution over a potentially large collection of computing resources.

The most important terms to understand are host, instance, node, cluster, and database:

Host — A computer system with a 32-bit (non-production use only) 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. There can be only one instance of Vertica running on a host at any time.

Node — A host configured to run an instance of Vertica. It is a member of a database cluster. For a database to have the ability to recover from the failure of a node requires at least three nodes. Vertica recommends that you use a minimum of four nodes.

Cluster — Refers a collection of hosts (nodes) bound to a database. A cluster is not part of a database definition and does not have a name.

Database — A cluster of nodes that, when active, can perform distributed data storage and SQL statement execution through administrative, interactive, and programmatic user interfaces.

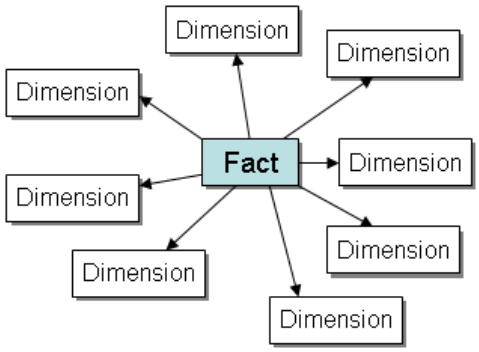
Logical Schema

From a SQL user's point of view, Vertica supports the standard relational data model, in which a schema consists of a collection of tables (relations), each with a collection of columns (attributes) and rows (tuples), and other SQL objects.

As in most relational systems, specific columns in Vertica tables can be designated as a primary key, or as a foreign key that references a primary key in another table. The tables in a Vertica database are assumed to obey the Dimensional Modeling concepts of a star schema or snowflake schema design. These are typical schema designs used in data warehouses.

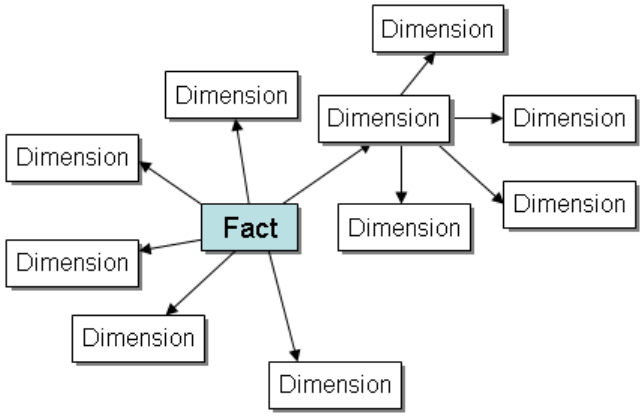
Star Schema

Sometimes called a star join schema, a star schema is the simplest data warehouse schema. In a star schema design there is a central fact table with a large number of records, optionally surrounded by a collection of dimension tables, each with a lesser number of records. Every dimension table participates in a 1::n join with the fact table.



Snowflake Schema

The same as a star schema except that a dimension table can be normalized (hierarchically decomposed) into additional dimension tables. Every dimension table participates in a 1::n join with the fact table or another dimension table.



For more information, see Designing a Logical Schema in the Administrator's Guide.

Physical Schema

In traditional database architectures, data is primarily stored in tables. Additionally, secondary tuning structures such as index and materialized view structures are created for improved query performance. In contrast, tables do not occupy any physical storage at all in Vertica. Instead, physical storage consists of collections of table columns called projections.

Projections store data in a format that optimizes query execution. They are similar to materialized views in that they store result sets on disk rather than compute them each time they are used in a query. The result sets are automatically refreshed whenever data values are inserted or loaded.

Using projections provides the following benefits:

- Projections compress and encode data to greatly reduce the space required for storing data. Additionally, Vertica operates on the encoded data representation whenever possible to avoid the cost of decoding. This combination of compression and encoding optimizes disk space while maximizing query performance. See ***Projection Performance*** (page 24).
- Projections for fact and large dimension tables are distributed across database nodes to distribute the work load. See ***Projection Performance*** (page 24).
- Projections are transparent to end-users of SQL. The Vertica query optimizer automatically picks the best projections to use for any query.
- Projections also provide high availability and recovery. To ensure high availability and recovery, Vertica duplicates table columns on all nodes within the cluster. Thus, if one machine fails in a ***K-Safe*** (see "***K-Safety***" on page 61) environment, the database continues to operate normally using duplicate data on the remaining nodes. Once the node resumes its normal operation, it automatically recovers its data and lost objects by querying other nodes. See ***High Availability and Recovery*** (page 16) for an overview of this feature and ***High Availability Through Projections*** (page 27) for an explanation of how Vertica uses projections to ensure high availability and recovery.

How Projections are Created

For each table in the database, Vertica requires a minimum of one projection, called a ***superprojection*** (on page 74). A superprojection is a projection for a single table that contains all the columns in the table.

To get your database up and running quickly, Vertica automatically creates a default superprojection for each table created through the CREATE TABLE and CREATE TEMPORARY TABLE statements. See these statements for additional details about the superprojections created through these statements.

By creating a superprojection for each table in the database, Vertica ensures that all SQL queries can be answered. However, default superprojections do not optimize database performance. Vertica recommends that you start with the default projections and then use ***Database Designer*** (on page 55) to optimize your database. Database Designer creates new projections that optimize your database based on its data statistics and the queries you use. Database Designer:

1 Analyzes your ***logical schema*** (on page 62), sample data, and sample queries (optional).

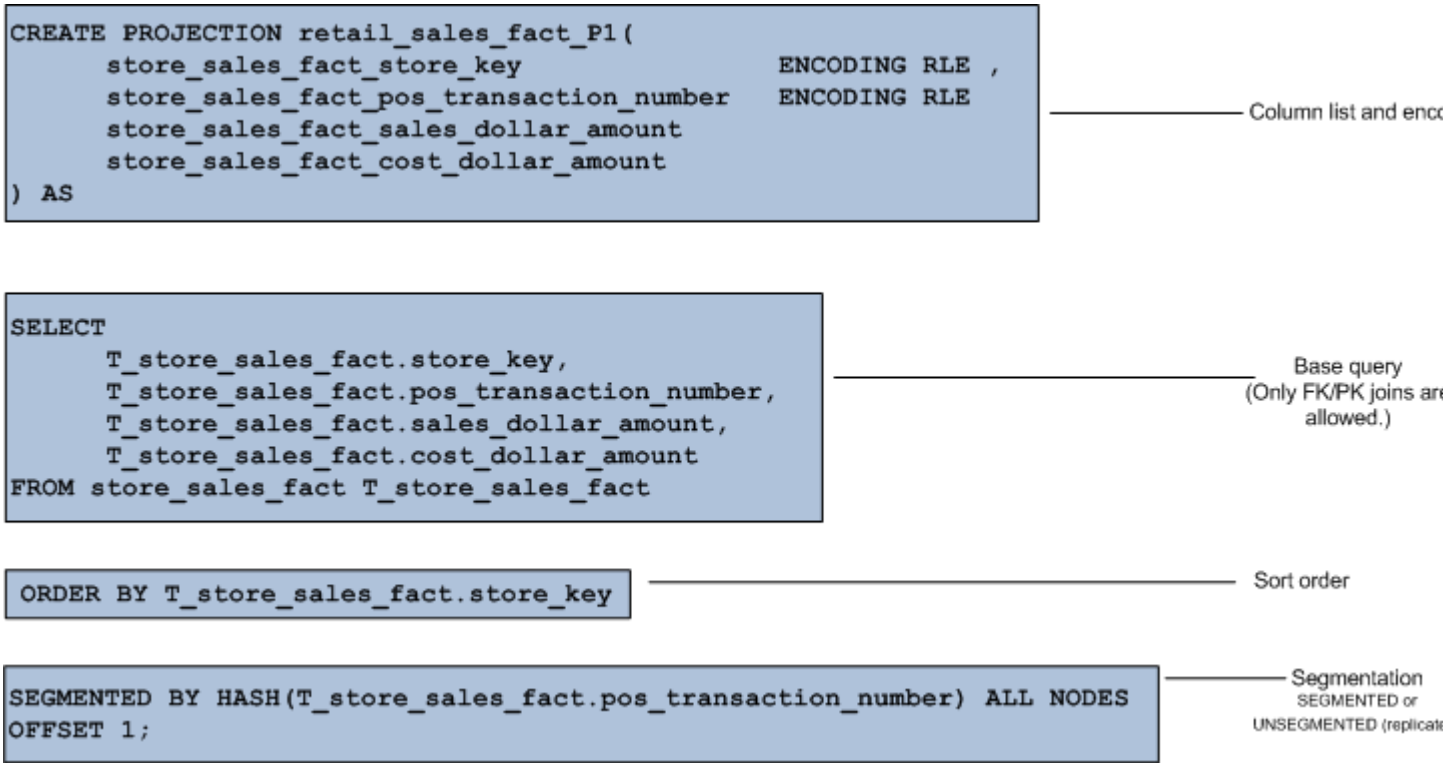
2 Creates a ***physical schema*** (on page 66) design (projections) in the form of a SQL script that can be deployed automatically or manually.

In most cases, the designs created by the Database Designer provide excellent query performance within physical constraints. The Database Designer uses sophisticated strategies to provide excellent ad-hoc query performance while using disk space efficiently. If you prefer, you can design custom projections. This option is intended for advanced users only, under the guidance of ***Technical Support*** (on page 1).

For more information about creating projections, see Creating a Physical Design in the Administrator's Guide.

Anatomy of a Projection

The CREATE PROJECTION statement specifies individual projections. The following sample depicts the elements of a projection.



The previous example contains the following significant elements:

Column list and encoding

Lists every column within the projection and defines the encoding for each columns. Unlike traditional database architectures, Vertica operates on encoded data representations. Therefore, Vertica encourages you to use data encoding because it results in less disk I/O.

Base query

Identifies all the columns to incorporate in the projection through column name and table name references. The base query for fact table projections can contain PK/FK joins to dimension tables.

Sort order

The ORDER BY clause specifies a projection's sort order, which localizes logically-grouped values so that a disk read can pick up many results at once. The sort order optimizes for a specific query or commonalities in a class of queries based on the query predicate. The best sort orders are determined by the WHERE clauses. For example, if a projection's sort order is `(x, y)`, and the query's WHERE clause specifies `(x=1 AND y=2)`, all of the needed data is found together in the sort order, so the query runs almost instantaneously.

You can also optimize a query by matching the projection's sort order to the query's GROUP BY clause. If you do not specify a sort order, Vertica uses the order in which columns are specified in the column definition as the projection's sort order.

Segmentation

The segmentation clause determines whether a projection is segmented or replicated across nodes within the database. Replication stores identical copies of projections for dimension tables across all nodes in the cluster. This ensures high availability and recovery. Segmentation distributes contiguous pieces of projections, called segments, for fact and large dimension tables across database nodes. This maximizes database performance by distributing the load.

Projection Concepts

For each table in the database, Vertica requires a projection, called a ***superprojection*** (on page 74). A superprojection is a projection for a single table that contains all the columns in the table. By creating a superprojection for each table in the database, Vertica ensures that all SQL queries can be answered.

In addition to superprojections, you can create one or more single-table projections that each contain a subset of columns from a table. These projections are used to optimize queries and they require only the columns necessary to process these queries. They are referred to as ***query-specific projections*** (see "***Query-specific projection***" on page 68).

Both superprojections and query-specific projections for fact tables support PK/FK joins to dimension tables to store the results of a join between a single fact table in the logical schema (the anchor table) with one or more dimension tables. Projections that contain joins are called ***pre-join projections*** (see "***Pre-join projection***" on page 66) because they store the results of a join. Pre-join projections can have only inner joins between tables on their primary and foreign key columns. Outer joins are not allowed. Pre-join projections provide a significant performance advantage over joining tables when queries are run.

Projection Performance

Vertica provides the following methods for maximizing the performance of all projections:

Encoding and Compression

Vertica operates on encoded data representations. Therefore, Vertica Systems, Inc. encourages you to use data encoding whenever possible because it results in less disk I/O and requires less disk space. For a description of the available encoding types, see encoding-type in the SQL Reference Manual.

Sort Order

The sort order optimizes for a specific query or commonalities in a class of queries based on the query predicate. For example, if the WHERE clause of a query is (X=1 AND Y=2) and a projection is sorted on (X,Y), the query runs almost instantaneously. It is also useful for sorting a projection to optimize a group by query. Simply match the sort order for the projection to the query group by clause.

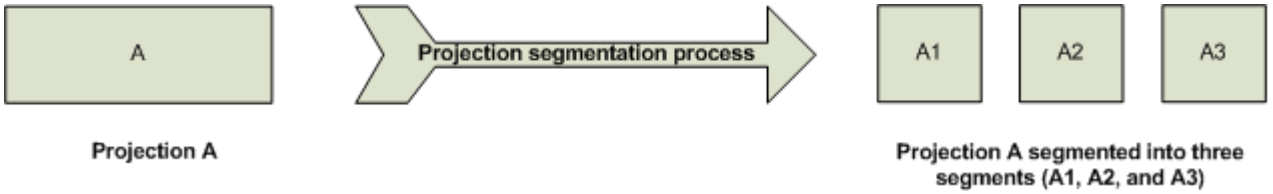
Segmentation

Segmentation distributes contiguous pieces of projections, called segments, for fact and large dimension tables across database nodes. This maximizes database performance by distributing the load. See ***Projection Segmentation*** (page 26).

In many cases, the performance gain for ***superprojections*** (see "***Superprojection***" on page 74) provided through these methods is sufficient enough that creating additional query-specific projections is unnecessary.

Projection Segmentation

Projection segmentation splits individual projections into chunks of data of similar size, called segments. One segment is created for and stored on each node. The following illustration shows the segmentation for a projection called *A* for a three node cluster.



Projection segmentation provides high availability and recovery and optimizes query execution. Specifically, it:

- Ensures high availability and recovery through K-Safety.
- Spreads the query execution workload across multiple nodes.
- Allows each node to be optimized for different query workloads.

Projections for fact tables are typically very large, so they are segmented to spread the query execution workload across multiple nodes. Projections for dimension tables are smaller so they are usually replicated. However, projections for large dimension tables can be segmented. A dimension table is considered to be large when it is approximately the same size as a fact table.

Types of Projection Segmentation

Vertica provides two ways in which to determine how data is segmented: hash and range segmentation. By default, Vertica uses hash segmentation.

- **Hash Segmentation**

  Hash segmentation is the preferred method of segmentation. It allows you to segment a projection based on a built-in hash function that provides even distribution of data across multiple nodes, resulting in optimal query execution. In a projection, the data to be hashed consists of one or more column values, each having a large number of unique values and an acceptable amount of skew in the value distribution. Primary key columns that meet the criteria could be an excellent choice for hash segmentation.

- **Range Segmentation**

  Range segmentation is accomplished by dividing up the range of possible values of an expression into disjoint sub-ranges, which are then allocated to specific nodes. The expression used to determine segmentation is known as the segmentation expression or segmentation criteria and can consist of a single projection column or an expression or function involving multiple projection columns. Range segmentation is useful when you have an artificial segmentation key.

  Note: Range segmentation based on dates or timestamps is not recommended because it results in a skewed data distribution. For example, suppose you segment by year using a DATE column. All rows inserted within the current year would be loaded into the same segments on specific nodes. Thus, a query asking for rows inserted during any specific year would be run only on those nodes. The rest of the nodes would be idle.

High Availability Through Projections

To ensure high availability and recovery for database clusters of three or more nodes, Vertica copies projections and stores them on multiple nodes within the database. This is feasible because the data in a Vertica database is compressed and encoded. Vertica provides two ways for storing copies of projections: projection replication and the creation of buddy projections.

Projection Replication

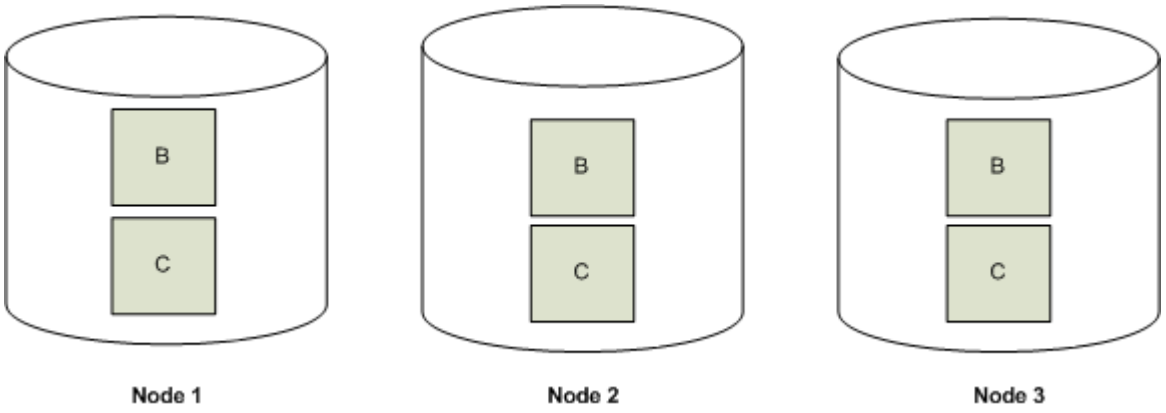
Replication consists of creating duplicates of unsegmented projections on all nodes in a database.

Replication has two purposes:

- Distributed query execution across multiple nodes.
- High availability and recovery. In a K-Safe database, replicated projections serve as buddy projections. This means that a replicated projection on any node can be used for recovery.

Typically projections for dimension tables are replicated because they are small enough that segmenting them provides no significant advantage. However, projections for large dimension tables can be segmented. (See ***Projection Segmentation*** (page 26).)

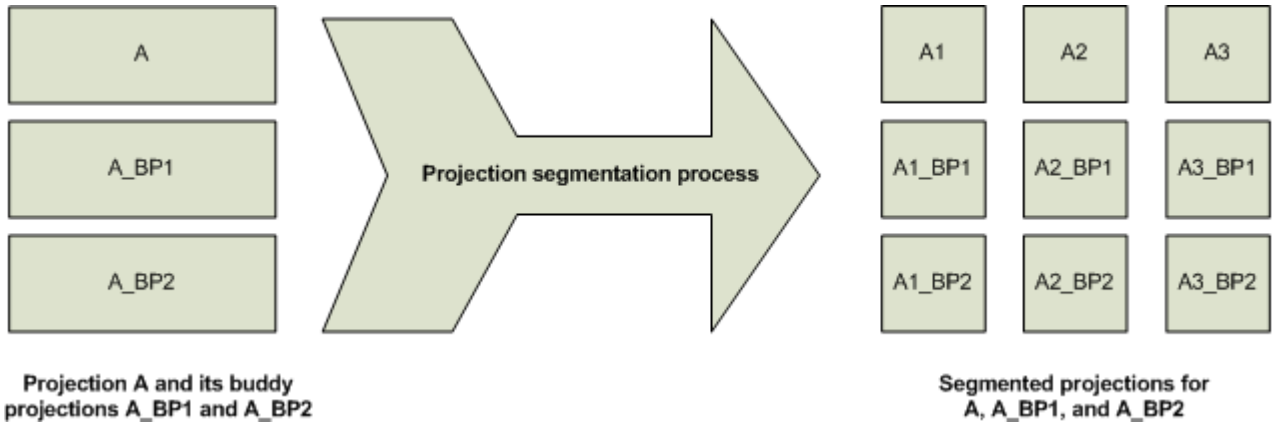
The following illustration shows two projections, B and C, replicated across a three node cluster.



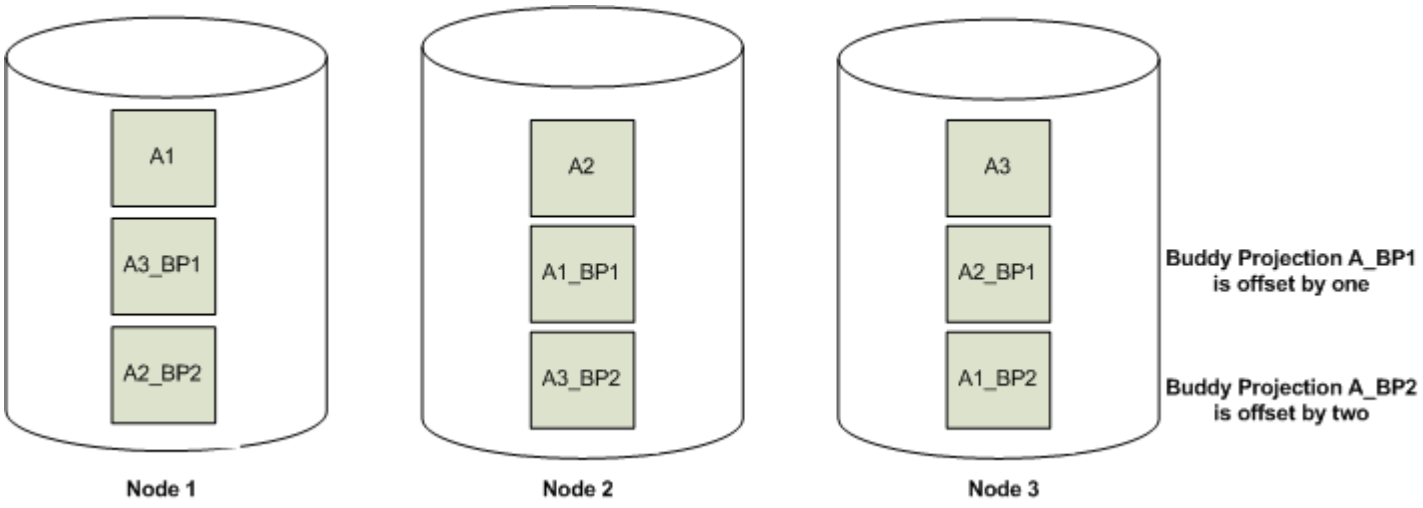
Buddy Projections

Buddy projections are copies of segmented projections, and they are distributed across database nodes. (See ***Projection Segmentation*** (page 26).) However the distribution is modified so that segments that contain the same data are distributed to different nodes. This ensures that if a node goes down, all the data is available on the remaining nodes. Vertica distributes segments to different nodes by using offsets. For example, segments that comprise the first buddy projection (A_BP1) would be offset from projection A by one node and segments from the second buddy projection (A_BP2) would be offset from projection A by two nodes.

The following illustration shows the segmentation for a projection called *A* and its buddy projections, *A_BP1* and *A_BP2,* for a three node cluster.



The following illustration shows how Vertica uses offsets to ensure that every node has a full set of data for the projection.



This example illustrates how one projection and its buddies are segmented across nodes. However, each node can easily store a collection of segments from various projections.

Projection Naming

Whether a projection is created automatically or you use Database Designer, Vertica uses a standard set of naming conventions which include the base name of the table with a suffix appended to it that indicates the type of projection. If the projection name would not be unique, Vertica automatically appends either v1 or v2 to it.

Projection Type	Suffix	Examples	Unique Name Example
Replicated (unsegmented) or unsegmented on one node	`_super`	`customer_dimension_vmart_super`	`customer_dimension_vmart_super_v1`

Replicated (unsegmented) on all nodes	_<nodename>	customer_dimension_vmart_node01 customer_dimension_vmart_node02 customer_dimension_vmart_node03	customer_dimension_vmart_v1_node01 customer_dimension_vmart_v1_node02 customer_dimension_vmart_v1_node03
Segmented on more than one node	_b<offset>	customer_dimension_vmart_b0 customer_dimension_vmart_b1	customer_dimension_vmart_v1_b0 customer_dimension_vmart_v2_b1

Note: Vertica might truncate the base table name to ensure the projection name fits the length limit.

Database Setup

The process of setting up a Vertica database is described in detail in the Administrator's Guide. It involves the following tasks:

Prepare SQL scripts and data files

The first part of the setup procedure can be done well before Vertica is installed. It consists of preparing the following files:

- Logical schema script
- Loadable data files (dimension table and fact tables)
- Load scripts
- Sample query script (training set)

Create the database

This part requires that Vertica be installed on at least one host. The following tasks are not in sequential order.

- Use the ***Administration Tools*** (on page 50) to:
  - Create a database
  - Connect to the database
- Use the ***Database Designer*** (on page 55) to design the physical schema.
- Use the ***vsql*** (on page 76) interactive interface to run SQL scripts that:
  - Create tables and constraints
  - Create ***projections*** (see "***Projection***" on page 67)

Test the empty database

- Test for sufficient projections using the sample query script
- Test the projections for ***K-Safety*** (on page 61)

Test the partially-loaded database

- Load the dimension tables
- Partially load the fact table
- Check system resource usage
- Check query execution times
- Check projection usage

Complete the fact table load

- Monitor system usage
- Complete the fact table load

Set up security

For security-related tasks, see Implementing Security.

- [Optional] Set up SSL
- [Optional] Set up client authentication
- Set up database users and privileges

Set up incremental loads

Set up periodic ("trickle") loads.

Database Connections

You can connect to a Vertica database in the following ways:

- Interactively using the **vsql** client, as described in Using vsql in the Administrator's Guide.

  vsql is a character-based, interactive, front-end utility that lets you type SQL statements and see the results. It also provides a number of meta-commands and various shell-like features that facilitate writing scripts and automating a variety of tasks.

  You can run vsql on any node within a database. To start vsql, use the ***Administration Tools*** (on page 50) or the shell command described in Using vsql.
- Programmatically using the **JDBC** driver provided by Vertica, as described in Using JDBC in the Programmer's Guide.

  An abbreviation for Java Database Connectivity, JDBC is a call-level application programming interface (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.
- Programmatically using the **ODBC** driver provided by Vertica, as described in Using ODBC in the Programmer's Guide.

  An abbreviation for Open DataBase Connectivity, ODBC is a standard application programming interface (API) for access to database management systems.
- Programmatically using the **ADO.NET** driver provided by Vertica, as described in Using ADO.NET in the Programmer's Guide.

  The Vertica driver for ADO.NET allows applications written in C# and Visual Studio 2008 to read data from, update, and load data into Vertica databases. It provides a data adapter that facilitates reading data from a database into a data set, and then writing changed data from the data set back to the database. It also provides a data reader (VerticaDataReader) for reading data and autocommit functionality for committing transactions automatically.
- Programmatically using **Perl** and the DBI driver, as described in Using Perl in the Programmer's Guide.

  Perl is a free, stable, open source, cross-platform programming language licensed under its Artistic License, or the GNU General Public License (GPL).
- Programmatically using **Python** and the pyodbc driver, as described in Using Python in the Programmer's Guide.

  Python is a free, agile, object-oriented, cross-platform programming language designed to emphasize rapid development and code readability.

Vertica Systems, Inc. recommends that you deploy Vertica as the only active process on each machine in the cluster and connect to it from applications on different machines. Vertica expects to use all available resources on the machine, and to the extent that other applications are also using these resources, suboptimal performance could result.

The Administration Tools

Vertica provides a set of tools that allows you to perform administrative tasks quickly and easily. Most of the database administration tasks in Vertica can be done using the Administration Tools.

Note: Always run the Administration Tools using the ***Database Administrator*** (on page 55) account on the ***Administration Host*** (on page 50) if possible. Make sure that no other Administration Tools processes are running.

If the Administration Host is down, run the Administration Tools on a different node in the cluster. That node permanently takes over the role of Administration Host.

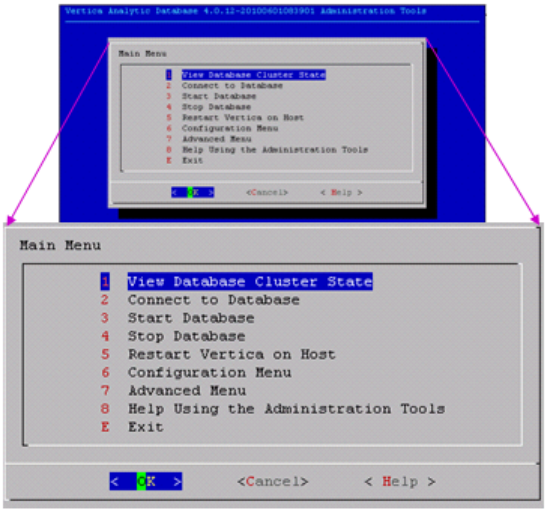
Running the Administration Tools

At the Linux command line:

```
$ /opt/vertica/bin/admintools [ -t | --tool ] toolname [ options ]
```

`toolname`	Is one of the tools described in the Administration Tools Reference.	
`options`	`-h` `--help`	Shows a brief help message and exits.
	`-a` `--help_all`	Lists all command-line subcommands and options as described in Writing Administration Tools Scripts.

If you omit the toolname and options, the Main Menu dialog box appears inside your console or terminal window with a dark blue background and a title on top. The screen captures used in this documentation set are cropped down to the dialog box itself, as shown below.



If you are unfamiliar with this type of graphical user interface, read Using the Graphical User Interface before you do anything else.

First Time Only

The first time you log in as the ***Database Administrator*** (on page 55) and run the Administration Tools, the user interface displays.

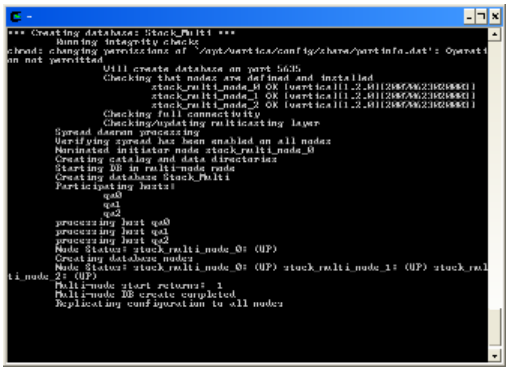
1 In the EULA (license agreement) window, type **`accept`** to proceed.

   A window displays, requesting the location of the license key file you downloaded from the Vertica Web site. The default path is `/tmp/vlicense.key`.

2 Type the absolute path to your license key (for example, `/tmp/vlicense.txt`) and click **OK**.

Between Dialogs

While the Administration Tools are working, you see the command line processing in a window similar to the one shown below. Do not interrupt the processing.



The Database Designer

The Vertica ***Database Designer*** (on page 55) is a UI-based tool that :

1. Analyzes your ***logical schema*** (on page 62), sample data, and sample queries
2. Creates a ***physical schema*** (on page 66) design (projections) in the form of a SQL script that you can redirect to a file and run.

In most cases, the designs created by the Database Designer provide excellent query performance within physical constraints. The Database Designer uses sophisticated strategies to provide excellent ad-hoc query performance while using disk space efficiently.

You can also design custom projections as described in the Administrator's Guide. This option is intended for advanced users only and should be done only under the guidance of Vertica ***Technical Support*** (on page 1).

See Also

- ***Physical Schema*** (page 22)
- Creating a Physical Design in the Administrator's Guide

K-Safety

A Vertica database is said to be K-Safe if a node can fail at any given time without causing the database to shut down. In Vertica, the value of K can be 0, 1, or 2. The value of K can be one 1 or two 2 only when the physical schema design meets certain requirements. To create designs that are K-Safe, Vertica recommends using the ***Database Designer*** (on page 55).

If, for example, a database having K=1 loses a node, the database continues to run normally. Increasing K-Safety to 2 allows Vertica to run normally if up to two nodes fail. When the failed node or nodes return and successfully recover, they can participate in database operations again. However if more than 2 nodes fail on a K=2 database, the system is no longer K-Safe and shuts down to prevent data inconsistencies. The entire database must then be started up again, as described in Restarting the Database.

K-Safety Requirements

When creating projections with the Database Designer, projection definitions that meet K-Safe design requirements are recommended and marked with the K-Safety level. Note the output from running the optimized design script generated by the Database Designer in the following example:

```
VMart_Schema=> \i VMart_Schema_design_opt_1.sql
CREATE PROJECTION
CREATE PROJECTION
mark_design_ksafe
----------------------
Marked design 1-safe
(1 row)
```

Determining K-Safety

To determine the K-Safety state of a running database, execute the following SQL command:

```
SELECT current_fault_tolerance FROM system;
current_fault_tolerance
-----------------------
                      1
(1 row)
```

Monitoring K-Safety

Monitoring tables can be accessed programmatically to enable external actions, such as alerts. You monitor the K-Safety level by polling the SYSTEM table column and checking the value. See SYSTEM in the SQL Reference Manual.

Loss of K-Safety

If you lose a node and your cluster's K-Safety level drops to 0, all DML operations remain available; however, performance is affected.

Database Security

Vertica secures access to the database and its resources by enabling you to control who has access to the database and what they are authorized to do with database resources once they have gained access. See Implementing Security.

Data Loading and Modification

SQL data manipulation language (DML) commands INSERT, UPDATE, and DELETE perform the same functions in Vertica as they do in row-oriented databases. These commands follow the SQL-92 transaction model and can be intermixed.

In Vertica, the COPY command is designed for bulk loading data into the database. The COPY operation reads data from a delimited text file and inserts records either into the ***WOS*** (see "***WOS (Write Optimized Store)***" on page 77) (memory) or directly into the ***ROS*** (see "***ROS (Read Optimized Store)***" on page 70) (disk). COPY automatically commits itself and any current transaction but is not atomic; some rows could be rejected. Note that COPY does not automatically commit when copying data into temporary tables.

Notes:

Use `COPY` with the `NO COMMIT` key words to prevent the current transaction from committing automatically (default behavior for all but temporary tables). This option is useful for executing multiple `COPY` commands in a single transaction. For example, all the rows in the following sequence commit in the same transaction.

```
=> COPY... NO COMMIT;
=> COPY... NO COMMIT;
=> COPY... NO COMMIT;
=> COMMIT;
```

`NO COMMIT` can be combined with any other existing `COPY` option, and all the usual transaction semantics apply.

If there is a transaction in progress initiated by a statement other than `COPY` (for example, `INSERT`), `COPY... NO COMMIT` adds rows to the same transaction as the earlier statements. The previous statements are NOT committed.

Tip: Use the `NO COMMIT` keywords to incorporate detection of constraint violations into the load process. Vertica checks for violations when queries are run, not when data is loaded. To avoid constraint violations, load data without committing it and then perform a post-load check of your data using the ANALYZE_CONSTRAINTS function. If the function finds constraint violations, you can easily roll back the load because you have not committed it.

You can use multiple, simultaneous database connections to load and/or modify data.

Workload Management

Vertica provides a sophisticated workload management scheme to enable diverse concurrent workloads to run efficiently against the database. For basic operations, Vertica provides a built-in GENERAL pool that is pre-configured based on RAM and machine cores and can be customized further to handle specific concurrency requirements.

For more advanced needs, the database administrator can establish new resource pools configured with limits on memory usage, concurrency, and priority. Each database user can optionally be restricted to use a specific resource pool and also be given quotas on memory usage to control resources used by their requests.

User-defined pools are useful when there are competing resource requirements across different classes of workloads. Some examples of scenarios that can make use of user-defined pools include:

- A large batch job takes up all of the server resources, leaving small jobs that update a web page to starve, thereby degrading user experience. To solve this problem, a dedicated resource pool can be created for web page requests to ensure they always get resources. Alternatively, a limited resource pool can be created for the batch job, so it cannot use up all the resources in the system.
- A certain application has lower priority and you would like to limit the amount of memory and number of concurrent users of this application. To solve this problem, a resource pool with an upper limit on the memory usage of a query can be created and associated with users of this application.

For more information, best practices, and additional scenarios, see Managing Workloads in the Administrator's Guide.

SQL Overview

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

Note: In Vertica, use a semicolon to end a statement or to combine multiple statements on one line.

Vertica Support for ANSI SQL Standards

Vertica SQL supports a subset of ANSI SQL-99.

See ***BNF Grammar for SQL-99*** (***http://savage.net.au/SQL/sql-99.bnf.html***)

Support for Historical Queries

Unlike most databases, the DELETE command in Vertica does not delete data; it marks records as deleted. The UPDATE command performs an INSERT and a DELETE. This behavior is necessary for ***historical queries*** (see "***Historical data***" on page 60). See Historical (Snapshot) Queries in the Programmer's Guide.

Joins

Vertica supports typical data warehousing query joins. For details, see Joins in the Programmer's Guide.

Transactions

Session-scoped isolation levels determine transaction characteristics for ***transactions*** (see "***Transaction***" on page 75) within a specific user ***session*** (on page 72). You set them through the SET SESSION CHARACTERISTICS command. Specifically, they determine what data a transaction can access when other transactions are running concurrently. See ***Transactions*** (page 42) in the Concepts Guide.

Query Execution

This topic provides a brief overview of how queries are executed in Vertica. For detailed information about writing and executing queries, see the Programmer's Guide.

Snapshot Isolation Mode

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 a historical query. The syntax is:

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

The command queries all data in the database up to but not including the current epoch without holding a lock or blocking write operations, which could cause the query to miss rows loaded by other users up to (but no more than) a specific number of minutes before execution.

Historical Queries

The DELETE command in Vertica does not actually delete data; it marks records as deleted. (The UPDATE command is actually a combined INSERT and a DELETE.) Thus, Vertica can run 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 representing the specified date and time, without holding a lock or blocking write operations. The specified TIMESTAMP value must be greater than or equal to the Ancient History Mark epoch.

You can control how much deleted data is stored on disk. See Managing Disk Space in the Administrator's Guide for details.

Parallel Query Execution

The query optimizer chooses the projections to use for any given incoming query and forms an optimized query plan for the ones chosen. The query plan that the optimizer produces is decomposed into “mini-plans,” which are run on various nodes in parallel, interspersed with data movement operations. A local executor at each node runs its assigned mini-plan.

Transactions

Session-scoped isolation levels determine transaction characteristics for ***transactions*** (see "***Transaction***" on page 75) within a specific user ***session*** (on page 72). You set them through the `SET SESSION CHARACTERISTICS` command. Specifically, they determine what data a transaction can access when other transactions are running concurrently.

A transaction retains its isolation level until it completes, even if the session's transaction isolation level has changed mid-transaction. Vertica internal processes (such as the Tuple Mover and Refresh operations) and DDL operations are run at `SERIALIZABLE` isolation to ensure consistency.

Although the Vertica query parser understands all four standard SQL isolation levels (`READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ`, and `SERIALIZABLE`) for a user session, internally Vertica uses only `READ COMMITTED` and `SERIALIZABLE`. Vertica automatically translates `READ UNCOMMITTED` to `READ COMMITTED` and `REPEATABLE READ` to `SERIALIZABLE`. Therefore, the isolation level Vertica uses could be more strict than the isolation level you choose.

By default, Vertica uses the `READ COMMITTED` isolation level. However, you can change the isolation level for the database or individual transactions. See Change Transaction Isolation Levels.

The following table highlights the behaviors of transaction isolation. For specific information see, ***SERIALIZABLE Isolation*** (page 45) and ***READ COMMITTED Isolation*** (page 43).

Isolation Level	*Dirty Read* **(on page 56)**	*Non Repeatable Read* **(see "*Non-repeatable read*" on page 64)**	*Phantom Read* **(on page 66)**
`READ COMMITTED`	Not Possible	Possible	Possible
`SERIALIZABLE`	Not Possible	Not Possible	Not Possible

Implementation Details

Vertica supports conventional SQL transactions with standard ***ACID*** (on page 50) properties:

- ANSI SQL 92 style-implicit transactions. You do not need to run a `BEGIN` or `START TRANSACTION` command.
- No redo/undo log or two-phase commits.
- The `COPY` command automatically commits itself and any current transaction (except when loading temporary tables). Vertica recommends that you `COMMIT` or `ROLLBACK` the current transaction before you use `COPY`.

Automatic Rollback

Reverts data in a database to an earlier state by discarding any changes to the database state that have been performed by a transaction's statements. In addition, it releases any locks that the transaction might have held. A rollback can be done automatically in response to an error or through an explicit `ROLLBACK` transaction.

Vertica supports transaction-level and statement-level rollbacks. A transaction-level rollback discards all modifications made by a transaction. A statement-level rollback reverses just the effects made by a particular statement. Most errors caused by a statement result in a statement-level rollback to undo the effects of the erroneous statement. Vertica uses `ERROR` messages to indicate this type of error. DDL errors, systemic failures, dead locks, and resource constraints result in transaction-level rollback. Vertica uses `ROLLBACK` messages to indicate this type of error.

To implement automatic, statement-level rollbacks in response to errors, Vertica automatically inserts an implicit savepoint after each successful statement one at a time. This marker allows the next statement, and only the next statement, to be rolled back if it results in an error. If the statement is successful, the marker automatically rolls forward. Implicit savepoints are available to Vertica only and cannot be referenced directly.

To explicitly roll back an entire transaction, use the `ROLLBACK` statement. To explicitly roll back individual statements, use explicit savepoints.

Savepoints

Vertica supports using savepoints. A savepoint is a special mark inside a transaction that allows all commands run after the savepoint was established to be rolled back, restoring the transaction to its former state in which the savepoint was established.

Savepoints are useful when creating nested transactions. For example, a savepoint could be created at the beginning of a subroutine. That way, the result of the subroutine could be rolled back, if necessary.

Use the `SAVEPOINT` statement to establish a savepoint, the `RELEASE SAVEPOINT` statement to destroy it, or the `ROLLBACK TO SAVEPOINT` statement to roll back all operations that occur within the transaction after the savepoint was established.

READ COMMITTED Isolation

A `SELECT` query sees a snapshot of the committed data at the start of the transaction. It also sees the results of updates run within its transaction, even if they have not been committed. This is standard ANSI SQL semantics for ***ACID*** (on page 50) transactions. Any `SELECT` query within a transaction should see the transactions's own changes regardless of isolation level.

DML statements acquire write locks to prevent other READ COMMITTED transactions from modifying the same data. SELECT statements do not acquire locks, which prevents read and write statements from conflicting.

READ COMMITTED is the default isolation level used by Vertica. For most general querying purposes, the READ COMMITTED isolation effectively balances database consistency and concurrency. However, data can be changed by other transactions between individual statements within the current transaction. This can result in ***nonrepeatable*** (see "***Non-repeatable read***" on page 64) and ***phantom reads*** (see "***Phantom read***" on page 66). Applications that require complex queries and updates might need a more consistent view of the database. If this is the case, use SERIALIZABLE isolation.

The following example illustrates reads and writes using READ COMMITTED isolation.

Session A	Session B	Description
SELECT C1 FROM T1; C1 -- (0 rows) COMMIT;		The SELECT statement in Session A reads committed data from T1.
INSERT INTO T1 (C1) VALUES (1);		Session A inserts a row, but does not yet commit.
SELECT C1 FROM T1; C1 -- 1 (1 rows)	SELECT C1 FROM T1; C1 -- (0 rows)	Session A reads the inserted row because it was inserted during the same transaction. However, Session B does not see the inserted value because it can only read committed data.
COMMIT;		Session A commits the INSERT and ends the transaction.
	SELECT C1 FROM T1; C1 -- 1 (1 rows)	The SELECT statement in Session B now observes the insert that session A committed. This is an example of a ***non-repeatable read*** (on page 64).
SELECT C1 FROM T1; C1 -- 1 (1 rows) COMMIT;		The SELECT statement in Session A begins a new transaction. It sees the previous inserted value because it was committed.

READ COMMITTED isolation uses exclusive (X) write locks that are maintained until the end of the transaction. The following example illustrates this.

Session A	Session B	Description
INSERT INTO T1 (C1) VALUES (2); (1 rows)		The transaction in session A acquires an insert (I) lock to insert row 2 into table T1.
	DELETE FROM T1 WHERE C1 >1;	The DELETE statement in Session B is blocked because the transaction cannot acquire an exclusive lock (X lock) until the entire transaction in Session A is completed and the insert (I) lock is released.

INSERT INTO T1 (C1) VALUES (3); (1 rows)		The transaction in session A inserts row 3. (It already has an insert (I) lock.)
COMMIT;		The COMMIT statement ends the transaction in Session A and releases its insert (I) lock.
	(2 rows)	The transaction in Session B obtains its X lock and deletes rows 2 and 3.

See Also

LOCKS and SET SESSION CHARACTERISTICS in the SQL Reference Manual

Configuration Parameters in the Administrator's Guide

SERIALIZABLE Isolation

SERIALIZABLE is the strictest level of SQL transaction isolation. Although this isolation level permits transactions to run concurrently, it creates the effect that transactions are running in serial order. It acquires locks for both read and write operations, which ensures that successive SELECT commands within a single transaction always produce the same results. SERIALIZABLE isolation establishes the following locks:

- Table-level read locks are acquired on selected tables and released at the end of the transaction. This prevents a transaction from modifying rows that are currently being read by another transaction.
- Table-level write locks are acquired on update and are released at the end of the transaction. This prevents a transaction from reading uncommitted changes to rows made within another transaction.

A SELECT sees, in effect, a snapshot of the committed data at the *start of the transaction*. It also sees the results of updates run within its transaction, even if they have not been committed.

The following example illustrates locking within concurrent transactions running with SERIALIZABLE isolation.

Session A	Session B	Description
SELECT C1 FROM T1; C1 -- (0 rows)	SELECT C1 FROM T1; C1 -- (0 rows)	Transactions in sessions A and B acquire shared locks. Both transactions can read from table T1.
	COMMIT;	The COMMIT statement in Session 2 ends the transaction and releases its read lock.
INSERT INTO T1 (C1) VALUES (1);		The transaction in Session A acquires an exclusive lock (X lock) and inserts a new row.
COMMIT;		The COMMIT statement in Session 1 ends the transaction and releases its X lock.

SELECT C1 FROM T1; C1 -- 1 (1 rows)	SELECT C1 FROM T1; C1 -- 1 (1 rows)	New transactions in Sessions A and B use a SELECT statement to see the new row previously created in Session A. They acquire shared locks.
INSERT INTO T1 (C1) VALUES (2);		The transaction in Session A is blocked from inserting a row because it cannot upgrade to an X lock.

The advantage of SERIALIZABLE isolation is that it provides a consistent view. This is useful for applications that require complex queries and updates. However, it reduces concurrency. For example, it is not possible to perform queries during a bulk load.

Additionally, applications using SERIALIZABLE must be prepared to retry transactions due to serialization failures. Serialization failures can occur due to deadlocks. When a deadlock occurs, the transaction that is waiting for the lock automatically times out after five (5) minutes. The following example illustrates a condition that can create a deadlock.

Session A	Session B	Description
SELECT C1 FROM T1; C1 -- (0 rows)	SELECT C1 FROM T1; C1 -- (0 rows)	Transactions in sessions A and B acquire shared locks on table T1. Both transactions can read from T1.
INSERT INTO T1 (C1) VALUES (1);		The transaction in Session A is blocked because it cannot upgrade to an exclusive lock (X lock) on T1 unless the transaction in Session B releases its lock on T1.
	INSERT INTO T1 (C1) VALUES (2);	The transaction in Session B is blocked because it cannot acquire an exclusive lock (X lock) unless the transaction in Session A releases its lock on T1. Neither session can proceed because each one is waiting for the other.
	ROLLBACK	Vertica automatically breaks the deadlock by rolling back the transaction in Session B and releasing the locks.
(1 rows) COMMIT;		The transaction in session A is able to upgrade to an X lock on T1 and insert the row.

Note: SERIALIZABLE isolation does not acquire locks on temporary tables, which are isolated by their transaction scope.

International Languages and Character Sets

Vertica provides the following support for international languages and character sets:

- ***Unicode character encoding*** (page 47)
- ***Locales*** (page 47)
- ***String functions*** (page 48)
- String literals (character)

Unicode Character Encoding

Vertica stores character data in UTF-8. UTF-8 is an abbreviation for Unicode Transformation Format-8 (where 8 equals 8-bit) and is a variable-length character encoding for Unicode created by Ken Thompson and Rob Pike. UTF-8 can 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).

All input data received by the database server is expected to be in UTF-8, and all data output by Vertica is in UTF-8. The ODBC API operates on data in UCS-2, and JDBC and ADO.NET APIs operate on data in UTF-16. The client drivers automatically convert data to (from) UTF-8 when sending to (receiving from) the database server.

See Implement Locales for International Data Sets in the Administrator's Guide for details.

Locales

The locale is a parameter that defines the user's language, country, and any special variant preferences, such as collation. Vertica uses the locale to determine the behavior of various string functions as well for collation for various SQL commands that require ordering and comparison; for example, GROUP BY, ORDER BY, joins, the analytic ORDER BY clause, and so forth.

By default, the locale for the database is `en_US@collation=binary` (English US). You can establish a new default locale that is used for all sessions on the database, as well as override individual sessions with different locales. Additionally the locale can be set through ODBC, JDBC, and ADO.net.

See the following topics in the Administrator's Guidefor details:

- Implement Locales for International Data Sets
- Supported Locales in the Appendix

String Functions

Vertica provides string functions to support internationalization. Unless otherwise specified, these string functions can optionally specify whether VARCHAR arguments should be interpreted as ***octet*** (on page 65) (byte) sequences, or as (locale-aware) sequences of characters. This is accomplished by adding "USING OCTETS" and "USING CHARACTERS" (default) as a parameter to the function.

See String Functions in the SQL Reference Manual for details.

Character String Literals

By default, string literals (`'...'`) treat back slashes literally, as specified in the SQL standard.

Tip: If you have used previous releases of Vertica and you do not want string literals to treat back slashes literally (for example, you are using a back slash as part of an escape sequence), you can turn off the `StandardConformingStrings` configuration parameter. See Internationalization Parameters in the Administrator's Guide. You can also use the `EscapeStringWarning` parameter to locate back slashes which have been incorporated into string literals, in order to remove them.

See String Literals (Character) in the SQL Reference Manual for details.

Get Started

To get started using Vertica, follow the Tutorial presented in the Getting Started Guide. The tutorial requires that you install Vertica on one or more hosts as described in the Installation Guide.

Glossary

Many HTML links in the Glossary 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.

Access rank

Determines the speed at which a column can be accessed. Columns are stored on disk from the highest ranking to the lowest ranking in which the highest ranking columns are placed on the fastest disks and the lowest ranking columns are placed on the slowest disks.

ACID

ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably, as follows:

- Atomicity — All of the tasks of a transaction are performed or none of them are. This means that if the entire transaction succeeds, it is committed. If any part of the transaction fails, the entire transaction fails and is rolled back.
- Consistency — Data integrity constraints are maintained. This ensures that the database remains in a consistent state before the start of the transaction and after the transaction is over (whether successful or not). Vertica provides limited constraint checking.
- Isolation — Concurrent transactions do not interact with one another.
- Durability — Short of catastrophic failure, the effects of committed transactions persist.

Administration host

The host on which the Vertica rpm package was manually installed. Always run the Administration Tools on this host if possible.

Administration Tools

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

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

Note: Throughout the Vertica documentation, you might see Administration Tools referred to as Admin Tools or admintools or adminTools. They all refer to the same utility.

AHM

Is the epoch prior to which historical data can be purged from physical storage. Abbreviated throughout the documentation as AHM.

Anchor table

The anchor table of a join can be:

- A large (fact) table in a star schema
- The central fact table in a snowflake schema
- A dimension table in a snowflake schema that functions as a fact table (with a ***restriction*** (page 72))

Ancient History Mark

Is the epoch prior to which historical data can be purged from physical storage. Abbreviated throughout the documentation as AHM.

TM

An abbreviation for the Tuple Mover. Runs in the background, optimizing the data layout. Tasks include moving data from memory (WOS) to disk (ROS), combining small ROS containers into larger ones, and purging deleted data.

Authentication

The process of verifying the identity of a user attempting to connect to a database.

Authorization

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

Blade server

Consists 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.

Bit

An abbreviation for **bi**nary dig**it** and is the smallest unit of data in a computer. Bits have a single binary value, 1 or 0, and these settings can be implemented as logical values (true or false) or as a switch (on or off).

Computers generally store data and execute instructions in bit multiples called bytes. There are eight bits in a byte.

Note: In some systems, the term octet is used for an eight-bit unit instead of byte.

Bitstring

A sequence of bits.

Byte

A basic unit of measurement of information storage in computers. In most systems a byte is made up of of eight bits. Half a byte (four bits) is called a nibble.

Buddy projection

Required for K-Safety. Two projections are considered to be buddies if they contain the same columns and have the same range segmentation or hash segmentation, using different node ordering. Buddy projections can have different sort orders for query performance purposes.

Bulk loading

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

C-Store

A research project at MIT, Brandeis, Brown, and the University of Massachusetts (Boston), on which Vertica is based.

CSV

Short for comma-separated values, another name for the comma-delimited format of data representation. For example: Smith,Joe,555-4615.

Cardinality

Refers to the number of unique values for a given column in a relational table:

- *High cardinality* — Refers to columns containing values that are unique, such as a customer ID or an employee e-mail address. For example, in the Vertica VMart schema, the `employee_dimension` table contains an `employee_key` column. This column contains values that uniquely identify each employee. Since the values in this column are unique and could be numerous, the column's cardinality type is referred to as high cardinality.
- *Normal cardinality* — Refers to columns containing values that are less unique, such as job titles and street addresses. An example of a normal-cardinality column would be `employee_title` or `employee_first_name` in the `employee_dimension` table, where many employees could share the same job title or same first name.
- *Low cardinality* — Refers to a low number of unique values, relative to the overall number of records in a table. For example, in the `employee_dimension` table, the column called `employee_gender` would contain two unique values: 'Male' or 'Female'. Since there are only two values possible in this column, cardinality is low.

Case-folded

Uppercase is converted to lowercase or lowercase is converted to uppercase.

Catalog

A set of files that contains information (metadata) about the objects in a database (nodes, tables, constraints, projections, etc.) Vertica maintains a catalog on each node in the cluster.

Catalog path

A storage location used to store the database catalog.

Checkpoint

Every time the Tuple Mover performs a move-out operation for a given projection, it records a Checkpoint Epoch for that projection, representing an epoch up to which the projection has no data in the WOS. The minimum checkpoint epoch across all projections on that node is called the node's **Checkpoint Epoch**. It represents a point in time up to which all the data was moved out to disk. If a K-safe=1 database experiences a single-node failure, the node's recovery process attempts to rebuild the data beyond the Checkpoint Epochs from other nodes. If all nodes fail, such as during a power outage, Vertica recovers the database back to the minimum Checkpoint Epochs across all the nodes, known as the Last Good Epoch (LGE).

Cluster

Refers a collection of hosts (nodes) bound to a database. A cluster is not part of a database definition and does not have a name.

Cold backup

Involves shutting down the Vertica database completely before backing it up. The advantage of a cold backup is that it is the fastest way to perform a backup because it ensures that there are no competing processes that require resources on the server. The disadvantage of a cold backup is that the database must be shut down. Cold backups can be full or incremental.

Comprehensive design

Using the Database Designer, a comprehensive design creates a complete initial or replacement design. If you have reasonably representative data, you can create a comprehensive design, optimized for storage and load. If you provide queries then the design is optimized for those queries.

Compression

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 for compressible data. Although compression is generally considered to be a form of encoding, 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 through a network connection. In Vertica, the scope of a connection is the same as that of a session.

Current epoch

The epoch into which data (COPY, INSERT, UPDATE, and DELETE operations) is currently being written. The current epoch advances automatically every three minutes.

Data path

A storage location that contains actual database data files.

Data warehouse

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) 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 cluster of nodes that, when active, can perform distributed data storage and SQL statement execution through administrative, interactive, and programmatic user interfaces.

Database administrator

Is the Linux user account that owns the database catalog and data storage on disk. The The database administrator (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

A tool that analyzes a logical schema definition, sample queries, and sample data, and creates a physical schema (a set of projections) in the form of a SQL script that can be deployed automatically or manually. The script creates a minimal set of superprojections to ensure K-Safety, and, optionally, pre-join projections. In most cases, the projections created by the Database Designer provide excellent query performance within physical constraints. You also have the option to create custom projections if the Database Designer does not meet your needs.

Database superuser

The automatically-created database user who has the same name as the Linux database administrator account and who can bypass all GRANT/REVOKE authorization. Superuser status cannot be granted to another user. Do not confuse the concept of a database superuser with Linux superuser (root) privilege. A database superuser cannot have Linux superuser privilege.

DBA

Is the Linux user account that owns the database catalog and data storage on disk. The The database administrator (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 Definition Language generally consists of CREATE, ALTER, and DROP commands, which operate on the logical schema metadata (tables, users, and so on)

DELTAVAL (delta encoding)

Stores only the differences between sequential data values instead of the values themselves.

Derived column

A column whose values are calculated by an expression at load time. The expression is specified within the COPY statement, and the column exists in the target database.

Design

Specific to the Database Designer, a design is made up of parameters (K-safety, design policy, and design type), projections, and queries. The design contains the design queries for which designs are wanted, including query information tables and the design output tables.

Deterministic

Deterministic functions return the same result every time they are called if given the same input values. For example, DATE_DIFF, ABS, and CEILING are deterministic functions.

Dialog

A Linux utility that creates user interfaces to shell scripts or other scripting languages, such as perl. In Vertica 4.1, Dialog is used to implement the Administration Tools, which run in a terminal window.

Dimension table

Sometimes called a lookup or reference table, a dimension table is one of a set of companion tables to a large (fact/anchor) table in a star schema. It contains the PRIMARY KEY column corresponding to the join columns in fact tables. 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.

Dirty read

Occurs when one transaction reads uncommitted changes made by another concurrent transaction. Vertica prevents dirty reads from occurring.

DML

SQL Data Manipulation Language generally consists of INSERT, UPDATE, and DELETE commands, which modify existing data (single records or sets of records) in the logical schema.

Dynamic SQL

A programmatic interface to a database management system that lets SQL statements be defined and run at run time, usually based on user input.

EOF

Is a condition in a computer operating system where no more data can be read from a data source. It refers to end-of-file. The data source is usually called a file or stream.

Source: Wikipedia

Encoding

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

Epoch

A logical unit of time in which a single change is made to data in the system.

Epoch map

A catalog object that provides mapping between time and epochs. Specifically, an epoch map contains a list of epoch numbers and their associated timestamps.

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.

Executor node

Any node that participates in executing a specific SQL statement. The initiator node can, and usually does, also function as an executor node.

Expression

The components of a query (columns, constants, functions) that compare one or more values against other values. Expressions can perform calculations when they are combined using arithmetic and/or logical operators and are generally used in a conditional statement.

External procedure

A procedure external to Vertica that you create, maintain, and store on the server.

Fact table

The 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 or snowflake schema and might also be referred to as the anchor table. It typically has a large number of records and is surrounded by a collection of dimension tables, each with fewer records. The fact table participates in a join with every dimension table. It can contain data but generally contains many join columns (with optional FOREIGN KEY constraints), each of which corresponds to the primary key column of a dimension table.

FIFO

An acronym for First In/First Out, FIFO is an abstraction for organizing and manipulating data relative to time and prioritization. The expression describes the principle of a queue processing technique or servicing conflicting demands by ordering process by first-come, first-served (FCFS) behavior: what comes in first is handled first, what comes in next waits until the first is finished, and so on.

Source: Wikipedia

Foreign key

A column that is used to join a table to other tables to ensure referential integrity of the data. A FOREIGN KEY constraint is a rule that states that a column cannot be null and can contain only values from the PRIMARY KEY column on a specific table.

Flattening (subqueries and views)

Occurs when a subquery or named view is internally rewritten so the subquery is combined with the outer query block. The result sets of the original and flattened queries are exactly the same, but the flattened query usually benefits from significant performance improvements.

Full backup

Consists of copying each catalog and all the data files (ROS containers) on each node as well as the complete `/opt/vertica/config` directory.

General pool

A special built-in pool that represents the total amount of RAM available to the resource manager for use by queries. Other pools can borrow memory from the GENERAL pool. See also Built-in Pools.

Grid computing

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.

Grouped ROS

Like a ROS, a Grouped ROS is a highly-optimized, read-oriented physical storage structure organized by projection that makes heavy use of compression and indexing. However, a Grouped ROS stores data for two or more grouped columns in one disk file.

Hash function

A reproducible method of converting data into a number that can serve as a digital fingerprint of the data. Hash functions take up to 32 arguments, usually column names, and select a specific node for each row, based on the values of the columns for that row. Use one of the built-in hash functions to segment projections. These functions provide even distribution of data over a set of nodes in a cluster.

Hash join

Are used only for equi-joins where hashed values are compared for equality, not for other relationships.

Hash segmentation

Allows you to segment a projection based on a built-in hash function that provides even distribution of data across some or all of the nodes in a cluster, resulting in optimal query execution.

Hexadecimal

A numeral system with a base of 16. It uses sixteen distinct symbols: 0–9 (representing values zero to nine) and A-F (representing values ten to fifteen). For example:

```
00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10 ...
```

Each hexadecimal digit represents four binary digits (bits)—also called a nibble.

To convert the base-16 hexadecimal value 0xA0FF to its base-10 decimal equivalent, hexadecimal progresses from right to left using the following places that represent a power of 16, starting with 16^0:

```
4096   256   16   1
A      0     F    F
```

Given that A=10 and F=15 in hex, the decimal result is calculated as follows:

```
(10 x 4096) + (0 x 256) + (15 x 16) + (15 x 1) = 41215
```

Source: Wikipedia

Histogram

In statistics, a histogram is a graphical display of numerical data in the form of vertical bars, with the area of each bar representing frequency.

Historical data

Refers to any data in memory or physical storage other than the current epoch. It includes all COPY, INSERT, UPDATE, and DELETE operations, including deleted rows. This allows Vertica to run a historical query on data written up to and including the epoch representing the specified date and time.

Historical query

Vertica can run 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 representing the specified date and time, without holding a lock or blocking write operations. The specified TIMESTAMP value must be greater than or equal to the Ancient History Mark epoch.

Host

A computer system with a 32-bit (non-production use only) 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.

Hot backup

Allows you to make a backup while the database is running and available to users. The advantage of a hot backup is that the database can continue to handle normal requests while the database is being backed up. The disadvantage of a hot backup is that it is slightly more complex to implement. Hot backups can be full or incremental.

ICU

An abbreviation for International Components for Unicode, ICU is an open source library that provides Unicode support.

ISO Week-Year

By definition, the ISO-8601 week starts on Monday, and the first week of a year contains January 4 of that year. In other words, the first Thursday of a year is in week 1 of that year.

Because of this, it is possible for early January dates to be part of the 52nd or 53rd week of the previous year. For example, `2005-01-01` is part of the 53rd week of year 2004, and `2006-01-01` is part of the 52nd week of year 2005.

Incremental backup

Backs up only those files that have been added to the database since the last backup was performed. Vertica supports incremental backups through file management utilities such as rsync. This is possible because Vertica never modifies data files, it adds files to and deletes them from the database. Once you have created a full backup, you can use your file management utility to synchronize the backup with your database.

Initiator node

In the context of a client connection, the initiator node is the node associated with the specific host to which the connection was made. The initiator node can, and usually does, also function as an executor node.

Instance

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

Interpolation

The process of constructing new data points within a range of known data points.

Immutable (invariant) functions

When run with a given set of arguments, immutable functions always produces the same result. The function is independent on any environment or session settings, such as locale. For example, 2+2 always equals 4. Another immutable function is AVG(). Some immutable functions can take an optional stable argument; in this case they are treated as stable functions.

JDBC

An abbreviation for Java Database Connectivity, JDBC is a call-level application programming interface (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

Vertica uses the concept of K-Safety for failure recovery. The K value represents the maximum number of nodes in a database that can fail and recover with no loss of data. In Vertica, the value of K can be zero (0), one (1), or two (2). The Physical Schema design must meet certain requirements. To create designs that are K-Safe, Vertica recommends using the Database Designer.

Last epoch

The last epoch is the current epoch minus one. SELECT statements under READ COMMITTED isolation read from the last epoch.

Last Good Epoch

A term used in manual recovery, LGE (Last Good Epoch) refers to the most recent epoch that can be recovered.

LGE

An abbreviation for Last Good Epoch. A term used in manual recovery, LGE (Last Good Epoch) refers to the most recent epoch that can be recovered.

Locale

The locale is a parameter that defines the user's language, country, and any special variant preferences, such as collation. Vertica uses the locale to determine the behavior of various string functions as well for collation for various SQL commands that require ordering and comparison; for example, GROUP BY, ORDER BY, joins, the analytic ORDER BY clause, and so forth.

By default, the locale for the database is `en_US@collation=binary` (English US). You can establish a new default locale that is used for all sessions on the database, as well as override individual sessions with different locales. Additionally the locale can be set through ODBC, JDBC, and ADO.net.

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, which make up the physical schema.

LZO

An abbreviation for Lempel-Ziv-Oberhumer, LZO is a data compression algorithm that focuses on decompression speed. The algorithm is lossless, and the reference implementation is thread safe.

Man-in-the-middle attack

A network security breach in which a third party makes independent connections with a client and server and relays messages between them, intercepting data throughout the process. Vertica supports using SSL to avoid this type of attack.

Manual recovery

The process of recovery after an unclean shutdown of the database, where the administrator must accept recovery from the ***Last Good Epoch*** (on page 62), which could lead to loss of some recently loaded data. See Failure Recovery.

Materialized view

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 changes. A projection is a special case of a materialized view.

Mergeout

A mergeout is the process of consolidating ROS containers and purging deleted records. Subsets of the Read Optimized Store (ROS) and sometimes referred to as ROSs, ROS containers are created by changes to the data stored within a projection as a result of bulk loads and DML. The Tuple Mover periodically merges ROS containers to maximize performance.

Meta-functions

Used to query or change the internal state of Vertica and are not part of the SQL standard. Since meta-functions access internal data structures, they cannot be called in DML, DDL, or SELECT queries. See Vertica Functions in the SQL Reference Manual.

Metadata

Data that describes the data in a database, such as data type, compression, constraints, and so forth, for each column in the tables. Metadata is stored in the Vertica catalog.

Moveout

The process of moving data from the WOS (Write Optimized Store), which is kept in memory, to the ROS (Read Optimized Store), which is kept on disk. This operation is performed by the Tuple Mover.

NaN

An abbreviation for Not A Number, "NaN" (disregarding case) is optionally followed by a sequence of characters enclosed in parentheses. The character string specifies the value of NaN in an implementation-dependent manner. The Vertica internal representation of NAN, for example, is 0xfff8000000000000LL on x86 machines.

Nibble

A four-bit aggregation (half a byte or half an octet) that can represent the decimal values 0 to 15 (16 distinct values). Since there are sixteen (24) possible values, a nibble corresponds to a single hexadecimal digit and is sometimes referred to as a "hex digit" or "hexit".

The following is a 4-bit nibble:

```
1 0 0 1
```

Note: In some systems, the term octet is used for an eight-bit unit instead of byte

Source: Wikipedia

Node

A host configured to run an instance of Vertica. It is a member of a database cluster. For a database to have the ability to recover from the failure of a node requires at least three nodes. Vertica recommends that you use a minimum of four nodes.

Node definition

A metadata object that binds a host to a database. A node definition contains a symbolic node name that is used to specify segmentation in projections.

Nondeterministic

Nondeterministic functions could return different results each time they are called, even when the same input values are provided. For example, GETDATE and CURRENT_TIMESTAMP are nondeterministic functions.

Non-repeatable read

Occurs in a READ COMMITTED isolation level when two identical queries in the same transaction produce different results. This occurs when another transaction commits changes that alter the results of the query after the first query has completed and before the second query has begun.

NUL

Represents a character whose ASCII/Unicode code is zero, sometimes qualified "ASCII NUL".

NULL

Means *no value*, and is true of a field (column) or constant but not of a character.

Octal

The base-8 number system that uses eight unique symbols (0, 1, 2, 3, 4, 5, 6, and 7), where each digit represents three binary digits, starting from the right.

For example, the binary representation for decimal 74 is 0b1001010, which groups into 001 001 010 — so the octal representation is 112.

Whereas in decimal systems, each decimal place is a base of 10

```
74 = 7 x 10^1 + 4 x 10^0
```

in octal numerals, each place is a power with base 8:

```
112 = 1 x 8^2 + 1 x 8^1 + 2 x 8^0
```

By performing the octal calculation, you can see 112 (octal) = 64 + 8 + 2 = 74 (in decimal).

Though octal is sometimes used in computing, instead of hexadecimal, hexadecimal is generally the more straightforward format and is used throughout the Vertica documentation.

Source: Wikipedia

Octet

A series of eight bits to form a single byte.

ODBC

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

OLAP

An abbreviation for Online Analytical Processing, OLAP answers multi-dimensional SQL analytical queries.

OLTP

An abbreviation for Online Transaction Processing, OLTP facilitates and manages real-time, transaction-oriented applications, such as data entry and retrieval transactions, for decision-support servers. OLTP technology is used in industries, such as financial services (e-trading and e-banking), manufacturing and mail order (e-commerce and order processing)

Out-of-date

A projection is out-of-date if it requires a refresh to participate in query execution.

Partition key

A table column that specifies how the table is partitioned.

Partitioning

Specifies how data is organized within individual nodes. Logically, the partition clause is applied after the segmented by clause.

Phantom read

Occurs in a READ COMMITTED isolation level when two identical queries in the same transaction produce different collections of rows. This occurs because a table lock is not acquired on SELECT during the initial query.

Physical schema

Consists of a set of projections used to store data on disk. The projections in the physical schema are based on the objects in the logical schema.

Physical schema design

A usable K=1 design based on an analysis of the sample data files and queries (if available). The physical schema design contains segmented (fact table) superprojections and replicated (dimension table) superprojections. It might also contain pre-join projections.

Pinned projection

A projection for a temporary table that is only on the initiator node. It is neither segmented nor replicated on other nodes.

Pre-join projection

A subset of query specific projections that are used to store the results of a join between a single large (fact/anchor) table in the logical schema with one or more dimension tables. As with other query-specific projections, Vertica calculates and stores a new result set each time data is inserted or loaded into these tables. This provides a significant performance advantage over joining tables when queries are run.

The result set of a pre-join projection is typically sorted for a specific query or commonalities in a class of queries based on the query predicate. This provides optimal query performance.

Pre-join projections can have only inner joins between tables on their primary and foreign key columns. Outer joins are not allowed in pre-join projections.

Predicate

Specify conditions that can be evaluated to SQL ***three-valued logic http://en.wikipedia.org/wiki/Ternary_logic*** (3VL) Boolean truth values. Predicates are also used to limit the effects of statements and queries.

Primary column

A column that is loaded from raw data, and not derived from an expression.

Primary key

A single column or combination of columns (called a compound key) that uniquely identifies each row in a table. A PRIMARY KEY constraint contains unique, non-null values.

Projection

A special case of a materialized view 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. A projection that joins one or more tables is called a pre-join projection.

Projection set

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

Purge

Permanently removes deleted data from physical storage so that the disk space can be reused. You can purge historical data up to and including the epoch in which the Ancient History Mark is contained.

Query cluster level

Determines the number of sets used to group similar queries. The query cluster level can be any integer from one (1) to the number of queries to be included in the physical schema design.

Queries are generally grouped based on the columns they access and the way in which they are used. The following work loads typically use different types of queries and are placed in different query clusters: drill downs, large aggregations, and large joins. For example, if a reporting tool and dashboard both access the same database, the reporting tool is likely to use a drill down to access a subset of data and the dashboard is likely to use a large aggregation to look across a large range of data. In this case, there would be at least two (2) query clusters.

Query optimizer

The component that evaluates different strategies for running a query and picks the best one.

Query-specific design

Using the Database Designer, a query-specific design creates a design with additional projections optimized for a single query.

Query-specific projection

Usually a custom pre-join projection designed to provide maximum performance for one or more specific queries. The Administrator's Guide describes how to write custom projections.

RDBMS

An abbreviation for relational database management system.

RLE (Run Length Encoding)

The abbreviation for Run Length Encoding, RLE replaces sequences of the same data values within a column by a single value and a count number.

RM

An abbreviation for Resource Manager. In a single-user environment, the system can devote all resources to a single query and get the most efficient execution for that one query. However, in a environment where several concurrent queries are expected to run at once, there is tension between providing each query the maximum amount of resources (thereby getting fastest run time for that query) and serving multiple queries simultaneously with a reasonable run time. The Resource Manager (RM) provides options and controls for resolving this tension, while ensuring that every query eventually gets serviced and that true system limits are respected at all times.

Range segmentation

Allows you to segment a projection based on a known range of values stored in a specific column chosen to provide even distribution of data across a set of nodes, resulting in optimal query execution.

Recovery

the process of restoring the database to a fully-functional state after one or more nodes in the system has experienced a software or hardware related failure. Vertica recovers nodes by querying replicas of the data stored on other nodes. For example, a hardware failure could cause a node to lose database objects or to miss changes made to the database (INSERTs, UPDATEs, and so on) while offline. When the node comes back online, it recovers lost objects and catches up with changes by querying the other nodes.

Referential integrity

Consists of a set of constraints (logical schema objects) that define primary key and foreign key columns.

- Each dimension table must have a PRIMARY KEY constraint.
- The fact table must contain columns that can be used to join the fact table to dimension tables.

- Fact table join columns must have FOREIGN KEY constraints in order to participate in pre-join projections.
- Outer join queries produce the same results as the corresponding inner join query if there is a FOREIGN KEY constraint on the outer table. Note that the inner table of the outer join query must always have a PRIMARY KEY constraint on its join columns

Refresh

Ensures that all projections on a node are up-to-date (can participate in query execution). This process could take a long time, depending on how much data is in the table(s).

Vertica internal operations (mergeout, refresh, and recovery) maintain partition separation except in certain cases:

- Recovery of a projection when the buddy projection from which the partition is recovering is identically sorted. If the projection is undergoing a full rebuild, it is recovered one ROS container at a time. The projection ends up with a storage layout identical to its buddy and is, therefore, properly segmented.

  Note: In the case of a partial rebuild, all recovered data goes into a single ROS container and must be partitioned manually.
- Manual tuple mover operations often output a single storage container, combining any existing partitions; for example, after executing any of the `PURGE()` operations.

Replication

The Vertica process of storing identical copies of data across all nodes in a cluster.

Resource pool

A resource pool comprises a pre-allocated subset of the system resources, with an associated queue. A resource pool is created using the CREATE RESOURCE POOL command as described in the SQL Reference Manual.

Resource Manager

In a single-user environment, the system can devote all resources to a single query and get the most efficient execution for that one query. However, in a environment where several concurrent queries are expected to run at once, there is tension between providing each query the maximum amount of resources (thereby getting fastest run time for that query) and serving multiple queries simultaneously with a reasonable run time. The Resource Manager (RM) provides options and controls for resolving this tension, while ensuring that every query eventually gets serviced and that true system limits are respected at all times.

Rollback

Reverts data in a database to an earlier state by discarding any changes to the database state that have been performed by a transaction's statements. In addition, it releases any locks that the transaction might have held. A rollback can be done automatically in response to an error or through an explicit `ROLLBACK` transaction.

Vertica supports transaction-level and statement-level rollbacks. A transaction-level rollback discards all modifications made by a transaction. A statement-level rollback reverses just the effects made by a particular statement. Most errors caused by a statement result in a statement-level rollback to undo the effects of the erroneous statement. Vertica uses `ERROR` messages to indicate this type of error. DDL errors, systemic failures, dead locks, and resource constraints result in transaction-level rollback. Vertica uses `ROLLBACK` messages to indicate this type of error.

ROS (Read Optimized Store)

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

ROS container

Subsets of the Read Optimized Store (ROS) and sometimes referred to as ROSs, ROS containers are created by changes to the data stored within a projection as a result of bulk loads and DML. The Tuple Mover periodically merges ROS containers to maximize performance.

rpm

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 metadata used to install and erase the archive files. The metadata 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 is safe if it has enough buddy projections for the current K-safety level.

SAN

An abbreviation for Storage Area Network, SAN 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.

Savepoint

A savepoint is a special mark inside a transaction that allows all commands run after the savepoint was established to be rolled back, restoring the transaction to its former state in which the savepoint was established.

Savepoints are useful when creating nested transactions. For example, a savepoint could be created at the beginning of a subroutine. That way, the result of the subroutine could be rolled back, if necessary.

Schema

Has several related meanings in Vertica:

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

Secure Shell

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. Secure Shell (SSH) utilizes strong encryption and authentication to ensure confidentiality, integrity, and authenticity of the transferred data.

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

Seed

The starting value used by a random number generation routine to create random numbers.

Segmentation

The horizontal partitioning of a projection so that it can be stored on multiple nodes in a database cluster. The goal is to distribute physical data storage evenly across a database so that all nodes can participate in query execution.

Note: Segmentation and partitioning are terms often used interchangeably for other databases, but they have completely separate functions in Vertica, where partitioning specifies how data is organized within individual nodes. Logically, the partition clause is applied after the segmented by clause.

Select list

The list of expressions that appears after the SELECT keyword and before the FROM clause is called the select list. The select list is where you specify the columns in the set of records you want Vertica to return from one or more tables or views.

Session

An occurrence of a user interacting with a database through the use of SQL statements. A session can be invoked using vsql or a JDBC application. In Vertica, the scope of a session is the same as that of a connection.

Session-scoped data is preserved beyond the lifetime of a single transaction. The table is truncated (all rows deleted) when you terminate a session.

Site

A deprecated term in Vertica that is used to mean node or host 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 a historical query. The syntax is:

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

The command queries all data in the database up to but not including the current epoch without holding a lock or blocking write operations, which could cause the query to miss rows loaded by other users up to (but no more than) a specific number of minutes before execution.

Snowflake schema

The same as a star schema except that a dimension table can be normalized (hierarchically decomposed) into additional dimension tables. Every dimension table participates in a 1::n join with the fact table or another dimension table.

Spread

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

Sort-merge join

If both inputs are pre-sorted, merge joins do not have to do any pre-processing. Vertica uses the term sort-merge join to refer to the case when one of the inputs must be sorted prior to the merge join. Vertica sorts the inner input side but only if the outer input side is already sorted on the join keys.

SQL

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

SSH

An abbreviation for Secure Shell, 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. Secure Shell (SSH) uses strong encryption and authentication to ensure confidentiality, integrity, and authenticity of the transferred data.

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

Stable functions

When run with a given set of arguments, stable functions produce the same result within a single query or scan operation. However, a stable function could produce different results when issued under a different environment, such as a change of locale and time zone. Expressions that could give different results in the future are also stable, for example `SYSDATE()` or `'today'`.

Star schema

Sometimes called a star join schema, a star schema is the simplest data warehouse schema. In a star schema design there is a central fact table with a large number of records, optionally surrounded by a collection of dimension tables, each with a lesser number of records. Every dimension table participates in a 1::n join with the fact table.

Standalone resource pool

A pool that is configured such that it cannot borrow memory from the GENERAL pool. A standalone pool has MEMORYSIZE = MAXMEMORYSIZE.

Storage location

A directory path used by Vertica to store catalog, actual data files or temporary data files.

Strict

Indicates that a function always returns null when any of its input arguments is null.

Superprojection

A projection that contains every column of a table in the logical schema. A table can have multiple superprojections with different sort orders.

Superuser

The automatically-created database user who has the same name as the Linux database administrator account and who can bypass all GRANT/REVOKE authorization. Superuser status cannot be granted to another user. Do not confuse the concept of a database superuser with Linux superuser (root) privilege. A database superuser cannot have Linux superuser privilege.

TM

An abbreviation for the Tuple Mover. Runs in the background, optimizing the data layout. Tasks include moving data from memory (WOS) to disk (ROS), combining small ROS containers into larger ones, and purging deleted data.

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.

Temp location

A storage location used as ***temp space*** (page 74) by Vertica.

Temp space

Disk space temporarily occupied by temporary files created by certain query execution operations, such as hash joins and sorts, in the case when they have to spill to disk. Such operations might also be encountered during queries, recovery, refreshing projections, and so on. If a ***temp location*** (on page 74) is provided to the database, Vertica uses it as temp space.

Transaction

A unit of work within a database management system that consists of an associated set of SQL statements. Vertica supports Atomicity, Consistency, Isolation, and Durability (ACID) for SQL transactions.

Transaction-scoped data is truncated (all rows deletes) after each `COMMIT` or rollback.

Tuple

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 could have entirely different storage representations.

Tuple Mover

Runs in the background, optimizing the data layout. Tasks include moving data from memory (WOS) to disk (ROS), combining small ROS containers into larger ones, and purging deleted data.

UDP

An abbreviation for User Datagram Protocol, UDP is an Open Systems Interconnection (OSI) transport layer protocol for client/server network applications based on Internet Protocol (IP). UDP is the primary alternative to the Transmission Control Protocol (TCP) and is used to send messages (datagrams), to other hosts on an IP network.

UTC

An abbreviation for Coordinated Universal Time (in English), UTC is the high-precision atomic time standard that replaced Greenwich Mean Time on 1 January 1972 as the basis for the main reference time scale or civil time in various regions. UTC is also referred to by the military and civil aviation as Zulu time (Z).

UTF-8

UTF-8 is an abbreviation for Unicode Transformation Format-8 (where 8 equals 8-bit) and is a variable-length character encoding for Unicode created by Ken Thompson and Rob Pike. UTF-8 can 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).

All input data received by the database server is expected to be in UTF-8, and all data output by Vertica is in UTF-8. The ODBC API operates on data in UCS-2, and JDBC and ADO.NET APIs operate on data in UTF-16. The client drivers automatically convert data to (from) UTF-8 when sending to (receiving from) the database server.

Unary operator

An operation with a single input.

Up to date

See ***Up-to-date*** (page 76).

Up-to-date

A projection is up-to-date (or up to date) if it is eligible to participate in query execution. Projections on empty tables are up-to-date upon creation. If the table has data loaded already, newly created projections are marked not up-to-date until refreshed. If a projection is refreshed while a node is down, that projection can be marked up-to-date even though it is missing data on one of the nodes. This is because the node will build the data during the recovery process before participating in queries.

User agent

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.

View

A named logical relation specified by an associated query that can be accessed similarly to a table in the FROM clause of a SQL statement. The results of the query are not stored but obtained on the fly when the SQL referencing the view is executed.

Volatile functions

Regardless of the arguments or environment, volatile functions can return different results on multiple invocations. RANDOM() is one example.

Volatility

Indicates whether a function returns the same output given the same input. There are 3 types:

- ***Immutable*** (page 61) (same output given same input)
- ***Stable*** (page 73) (same output within the same row)
- ***Volatile*** (page 76) (different output for each invocation, such as `RANDOM()`)

vsql

vsql is a character-based, interactive, front-end utility that lets you type SQL statements and see the results. It also provides a number of meta-commands and various shell-like features that facilitate writing scripts and automating a variety of tasks.

Window (analytic)

A *window* specifies partitioning, ordering, and framing for an analytic function—important elements that determine what data the analytic function takes as input with respect to the current row. The window `OVER()` clause specifies that the analytic function operates on a query result set (the rows that are returned after the `FROM`, `WHERE`, `GROUP BY`, and `HAVING` clauses have been evaluated). You use then use the `OVER()` clause to define a moving window of data for every row in a partition with certain analytic functions.

Workload management

In a single-user environment, the system can devote all resources to a single query and get the most efficient execution for that one query. However, in a environment where several concurrent queries are expected to run at once, there is tension between providing each query the maximum amount of resources (thereby getting fastest run time for that query) and serving multiple queries simultaneously with a reasonable run time. The Resource Manager (RM) provides options and controls for resolving this tension, while ensuring that every query eventually gets serviced and that true system limits are respected at all times.

WOS (Write Optimized Store)

An abbreviation for Write Optimized Store, WOS is a memory-resident data structure into which INSERT, UPDATE, DELETE, and COPY (without DIRECT hint) actions are recorded. Like the ROS, the WOS is arranged by projection but it stores records without sorting, compression, or indexing and thus supports very fast load speeds. The WOS organizes data by epoch and holds uncommitted transaction data.

Index

A

About the Documentation • 2
Access rank • 51
ACID • 43, 44, 51
Acrobat • 6
Administration host • 34, 51
Administration Tools • 31, 33, 51
Adobe Acrobat • 6
AHM • 52
Anatomy of a Projection • 24
Anchor table • 52
Anchor Table • 23, 27
Ancient History Mark • 52
Authentication • 52
Authorization • 52
Automatic Rollback • 44

B

Bit • 53
Bitstring • 53
Blade server • 53
Bold text • 7
Braces • 7
Brackets • 7
Buddy projection • 53
Bulk loading • 53
Byte • 53

C

Cardinality • 54
Case-folded • 54
Catalog • 54
Catalog path • 54
Character String Literals • 49
Checkpoint • 55
Cluster • 55
Cold backup • 55
Colored bold text • 7
Column store • 11
Column Store Architecture with FlexStore • 13
Comprehensive design • 55
Compression • 11, 15, 55
Connection • 55
Copyright Notice • 87
C-Store • 53
CSV • 54
Current epoch • 56

D

Data Encoding and Compression • 15
Data Loading and Modification • 39
Data path • 56
Data warehouse • 56
Database • 56
Database administrator • 34, 35, 56
Database Connections • 33
Database Designer • 16, 23, 31, 36, 57
Database Security • 38
Database Setup • 31
Database superuser • 57
DBA • 57
DDL • 57
DELTAVAL (delta encoding) • 57
Derived column • 57
Design • 58
Deterministic • 58
Dialog • 58
Dimension table • 58
Dimensional Modeling • 21
Dirty read • 43, 58
DML • 58
Documentation • 6
Dynamic SQL • 58

E

Ellipses • 7
Encoding • 59
EOF • 59
Epoch • 59
Epoch map • 59
ETL • 59
Executor node • 59
Expression • 59
External procedure • 60

F

Fact table • 60
FIFO • 60
Flattening (subqueries and views) • 60
Foreign key • 60
Full backup • 60

G

General pool • 61

Get Started • 50
Glossary • 51
Grid computing • 61
Grouped ROS • 61

H

Hash function • 61
Hash join • 61
Hash segmentation • 61
Hexadecimal • 61
High Availability and Recovery • 16, 23
High Availability Through Projections • 23, 28
Histogram • 62
Historical data • 41, 62
Historical query • 62
Host • 20, 62
Hot backup • 63
How Projections are Created • 23
HTML • 6
Hybrid Storage Model • 18

I

ICU • 63
Immutable (invariant) functions • 64, 81
Incremental backup • 63
Indentation • 7
Initiator node • 63
INSERT • 39
Instance • 63
International Languages and Character Sets • 48
Interpolation • 64
ISO Week-Year • 63
Italic text • 7

J

JDBC • 64

K

K-Safety • 16, 23, 31, 36, 64

L

Last epoch • 64
Last Good Epoch • 64, 65
LGE • 64
Locale • 65
Locales • 48
Logical schema • 24, 36, 65
Logical Schema • 21
LZO • 65

M

Man-in-the-middle attack • 65
Manual recovery • 65
Materialized view • 21, 23, 66
Mergeout • 66
Metadata • 66
Meta-functions • 66
Monospace text • 7
Moveout • 66

N

NaN • 66
Nibble • 67
Node • 67
Node definition • 67
Nondeterministic • 67
Non-repeatable read • 43, 45, 67
NUL • 67
NULL • 68

O

Octal • 68
Octet • 49, 68
ODBC • 68
OLAP • 68
OLTP • 13, 68
Out-of-date • 69

P

Partition key • 69
Partitioning • 69
PDF • 6
Phantom read • 43, 45, 69
Physical Architecture • 20
Physical schema • 24, 36, 69
Physical Schema • 13, 16, 23, 36
Physical schema design • 69
Pinned projection • 69
Predicate • 70
Preface • 9
Pre-join projection • 25, 70
Primary column • 70
Primary key • 70
Printing Full Books • 4
Projection • 13, 31, 70
Projection Concepts • 25

Projection Naming • 29
Projection Performance • 23, 25
Projection Segmentation • 26, 27, 28
Projection set • 70
Projections • 13, 23, 24, 25, 27, 28
Purge • 71

Q

Query cluster level • 71
Query Execution • 42
Query optimizer • 71
Query-specific design • 71
Query-specific projection • 25, 71

R

Range segmentation • 72
RDBMS • 13, 71
READ COMMITTED Isolation • 43, 44
Reading the Online Documentation • 2
Recovery • 72
Referential integrity • 21, 72
Refresh • 73
Replication • 73
Resource Manager • 73
Resource pool • 73
RLE (Run Length Encoding) • 71
RM • 72
Rollback • 74
ROS (Read Optimized Store) • 18, 39, 74
ROS container • 74
rpm • 74

S

Safe • 74
SAN • 75
Savepoint • 75
Savepoints • 44
Schema • 75
Secure Shell • 75
Security • 38
Seed • 75
Segmentation • 76
Select list • 76
SERIALIZABLE Isolation • 43, 46
Session • 41, 43, 76
Shell script • 7
Site • 20, 76
Snapshot isolation • 76
Snowflake schema • 21, 52, 77
Sort-merge join • 77
Spread • 77
SQL • 77
SQL extensions • 41
SQL Overview • 41
SSH • 77
Stable functions • 77, 81
Standalone resource pool • 78
Star schema • 21, 78
Storage location • 78
Strict • 78
String Functions • 48, 49
Suggested Reading Paths • 2, 4
Superprojection • 23, 25, 26, 78
Superuser • 78
Support • 1
Syntax conventions • 7

T

Table • 79
Technical Support • 1, 4, 24, 36
Temp location • 79
Temp space • 79
The Administration Tools • 34
The Database Designer • 36
The Vertica Approach • 11
TM • 52, 78
Transaction • 39, 41, 43, 79
Transactions • 41, 43
Tuple • 79
Tuple Mover • 18, 79
Typographical Conventions • 7

U

UDP • 80
Unary operator • 80
Unicode Character Encoding • 48
Up to date • 80
UPDATE • 39
Uppercase text • 7
Up-to-date • 80
User agent • 81
UTC • 80
UTF-8 • 80

V

Vertica Components • 12

Vertical line • 7
View • 81
Volatile functions • 81
Volatility • 81
vsql • 31, 81

W

Welcome to Vertica • 10
Where to Find Additional Information • 6
Where to Find the Vertica Documentation • 2
Window (analytic) • 81
Workload management • 82
Workload Management • 39
WOS (Write Optimized Store) • 18, 39, 82
WOS Overload • 39

Copyright Notice

Copyright© 2006-2011 Vertica Systems, Inc., and its licensors. All rights reserved.

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

The software described in this copyright notice is furnished under a license and may be used or copied only in accordance with the terms of such license. Vertica Systems, Inc. software contains proprietary information, as well as trade secrets of Vertica Systems, Inc., and is protected under international copyright law. Reproduction, adaptation, or translation, in whole or in part, by any means — graphic, electronic or mechanical, including photocopying, recording, taping, or storage in an information retrieval system — of any part of this work covered by copyright is prohibited without prior written permission of the copyright owner, except as allowed under the copyright laws.

This product or products depicted herein may be protected by one or more U.S. or international patents or pending patents.

Trademarks

Vertica™, the Vertica® Analytic Database™, and FlexStore™ 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.

DataDirect® and DataDirect Connect® are registered trademarks of Progress Software Corporation in the U.S. 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 Acknowledgments

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

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

ASMJIT

Copyright (c) 2008-2010, Petr Kobalicek <kobalicek.petr@gmail.com>

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

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

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

Boost

Boost Software License - Version 1.38 - February 8th, 2009

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.

bzip2

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

Copyright © 1996-2005 Julian R Seward. All rights reserved.

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

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

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

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

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

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

Julian Seward, Cambridge, UK.

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

bzip2/libbzip2 version 1.0 of 21 March 2000

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

Mike Burrows

David Wheeler

Peter Fenwick

Alistair Moffat

Radioed Neal

Ian H. Witten

Robert Sedgewick

Jon L. Bentley

Daemonize

Copyright © 2003-2007 Brian M. Clapper.

All rights reserved.

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

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

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

Ganglia Open Source License

Copyright © 2001 by Matt Massie and The Regents of the University of California.

All rights reserved.

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

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES 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 OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

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

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1995-2009 International Business Machines Corporation and others

All rights reserved.

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

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

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

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

Keepalived Vertica IPVS (IP Virtual Server) Load Balancer

Copyright © 2007 Free Software Foundation, Inc.
http://fsf.org/

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

jQuery

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

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

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

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

Lighttpd Open Source License

Copyright © 2004, Jan Kneschke, incremental

All rights reserved.

1. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
2. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
4. Neither the name of the 'incremental' nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

MersenneTwister.h

Copyright © 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,

Copyright © 2000 - 2009, Richard J. Wagner

All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

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

MIT Kerberos

Copyright © 1985-2007 by the Massachusetts Institute of Technology.

Export of software employing encryption from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Furthermore if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original MIT software. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

Individual source code files are copyright MIT, Cygnus Support, Novell, OpenVision Technologies, Oracle, Red Hat, Sun Microsystems, FundsXpress, and others.

Project Athena, Athena, Athena MUSE, Discuss, Hesiod, Kerberos, Moira, and Zephyr are trademarks of the Massachusetts Institute of Technology (MIT). No commercial use of these trademarks may be made without prior written permission of MIT.

“Commercial use” means use of a name in a product or other for-profit manner. It does NOT prevent a commercial firm from referring to the MIT trademarks in order to convey information (although in doing so, recognition of their trademark status should be given).

Portions of src/lib/crypto have the following copyright:

Copyright © 1998 by the FundsXpress, INC.

All rights reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

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

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

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

> Copyright © 2001, Dr Brian Gladman <brg@gladman.uk.net>, Worcester, UK.
> All rights reserved.

LICENSE TERMS

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

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

DISCLAIMER

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

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

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

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

are subject to the following license:

Copyright © 2004 Sun Microsystems, Inc.

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

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

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

Npgsql-.Net Data Provider for Postgresql

Copyright © 2002-2008, The Npgsql Development Team

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

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

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

Open LDAP

The OpenLDAP Public License

Version 2.8, 17 August 2003

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

1 Redistributions in source form must retain copyright statements and notices,

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

3 Redistributions must contain a verbatim copy of this document.

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

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

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

OpenLDAP is a registered trademark of the OpenLDAP Foundation.

Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.
Permission to copy and distribute verbatim copies of this document is granted.

Open SSL

OpenSSL License

Copyright © 1998-2008 The OpenSSL Project. All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)"

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

PCRE LICENCE

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

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

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

THE BASIC LIBRARY FUNCTIONS

Written by: Philip Hazel
Email local part: ph10
Email domain: cam.ac.uk
University of Cambridge Computing Service,
Cambridge, England.
Copyright (c) 1997-2010 University of Cambridge
All rights reserved.

THE C++ WRAPPER FUNCTIONS

Contributed by: Google Inc.
Copyright (c) 2007-2010, Google Inc.
All rights reserved.

THE "BSD" LICENCE

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

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

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

End

Perl Artistic License

Copyright © August 15, 1997

Preamble

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

Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The End

Pexpect

Copyright © 2010 Noah Spurrier

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

Free, open source, and all that good stuff.

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

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

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

PHP License

The PHP License, version 3.01

Copyright © 1999 - 2009 The PHP Group. All rights reserved.

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

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

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

3 The name "PHP" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact group@php.net.

4 Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "Foo for PHP" instead of calling it "PHP Foo" or "phpfoo"

5 The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number.

- Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.

6 Redistributions of any form whatsoever must retain the following acknowledgment:

"This product includes PHP software, freely available from <http://www.php.net/software/>".

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

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

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

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

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

PostgreSQL

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

Portions Copyright © 1996-2005, The PostgreSQL Global Development Group

Portions Copyright © 1994, The Regents of the University of California

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

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

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

Python Dialog

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

Upstream Author:

Peter Astrand <peter@cendio.se>

Robb Shecter <robb@acm.org>

Sultanbek Tezadov <http://sultan.da.ru>

Florent Rougon <flo@via.ecp.fr>

Copyright © 2000 Robb Shecter, Sultanbek Tezadov

Copyright © 2002, 2003, 2004 Florent Rougon

License:

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

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

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

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

RRDTool Open Source License

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

RRDTOOL - Round Robin Database Tool

A tool for fast logging of numerical data graphical display of this data.

Copyright © 1998-2008 Tobias Oetiker

All rights reserved.

GNU GPL License

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

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

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

FLOSS License Exception

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

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

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

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

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

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

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

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

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

FLOSS License List

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

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

Spread

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

Copyright © 1993-2006 Spread Concepts LLC.

All rights reserved.

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

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

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

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).

SNMP

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

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

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

> Copyright © 1989, 1991, 1992 by Carnegie Mellon University
>
> Derivative Work - 1996, 1998-2000
>
> Copyright © 1996, 1998-2000 The Regents of the University of California
>
> All Rights Reserved

Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU and The Regents of the University of California not be used in advertising or publicity pertaining to distribution of the software without specific written permission.

CMU AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL CMU OR THE REGENTS OF THE UNIVERSITY OF CALIFORNIA BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM THE LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

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

Copyright © 2001-2003, Networks Associates Technology, Inc

All rights reserved.

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

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

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

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

Portions of this code are copyright (c) 2001-2003, Cambridge Broadband Ltd.

All rights reserved.

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

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

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

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

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

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Use is subject to license terms below.

This distribution may include materials developed by third parties.

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

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

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

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

Part 5: Sparta, Inc copyright notice (BSD)

Copyright © 2003-2006, Sparta, Inc

All rights reserved.

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

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

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

Part 6: Cisco/BUPTNIC copyright notice (BSD)

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

All rights reserved.

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

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

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

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

Copyright © Fabasoft R&D Software GmbH & Co KG, 2003

oss@fabasoft.com

Author: Bernhard Penz

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

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of Fabasoft R&D Software GmbH & Co KG or any of its subsidiaries, brand or product names may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE

LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Tecla Command-line Editing

Copyright © 2000 by Martin C. Shepherd.

All rights reserved.

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

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

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

Webmin Open Source License

Copyright © Jamie Cameron

All rights reserved.

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

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

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

3 Neither the name of the developer nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

xerces

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

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

Portions of this software were originally based on the following:

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

zlib

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

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

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

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

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

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org

Mark Adler madler@alumni.caltech.edu