The Vertica® Analytic Database Technical Overview White Paper

A DBMS Architecture Optimized for Next-Generation Data Warehousing

Table of Contents

The Vertica® Analytic Database Advantage

The Vertica Analytic Database is the only database built from scratch to handle today's heavy business intelligence workloads. In customer benchmarks, Vertica has been shown to manage terabytes of data running on extraordinarily low-cost hardware and answers queries 50 to 200 times faster than competing row-oriented databases and specialized analytic hardware

This document summarizes the key aspects of Vertica's technology that enable such dramatic performance benefits, and compares the design of Vertica to other popular relational systems.

The key to Vertica's performance is three fold:

1. Vertica organizes data on disk as columns of values from the same attribute, as opposed to storing it as rows of tabular records. This organization means that when a query needs to access only a few columns of a particular table, only those columns need to be read from disk. Conversely, in a row-oriented database, all values in a table are typically read from disk, which wastes I/O bandwidth.

2. Vertica employs **aggressive compression** of data on disk, as well as a query execution engine that is able to keep data compressed while it is operated on. Compression in Vertica is particularly effective, as values within a column tend to be quite similar to each other and compress very well—often by up to 90%. In a traditional row-oriented database, values within a row of a table are not likely to be very similar, and hence are unlikely to compress well. Columnar compression and direct operation on compressed data shift the bottleneck in query processing from disks (which are not getting faster) to CPUs (which are getting faster all the time).

3. Because data is compressed so aggressively, Vertica has sufficient space to store multiple copies of the data to ensure fault tolerance and to improve concurrent and ad-hoc query performance. Logical tables are decomposed and physically stored as overlapping groups of columns, called *"projections,"* and each projection is sorted on a different attribute (or set of attributes), which optimizes them for answering queries with predicates on its sort attributes

   A Vertica database is composed exclusively out of these query-optimized structures on disk, without the overhead of base tables. It's similar in concept to a database made entirely of materialized views (with no base tables).

Of course, compression, column-orientation, and table decomposition are transparent to the end-user. In fact, Vertica provides a standard SQL interface to users, as well as compatibility with existing ETL, reporting, and business intelligence (BI) tools. This makes it easy to migrate existing databases to Vertica and use other BI technologies with Vertica databases. Vertica is designed to run on inexpensive clusters or "grids" of off-the-shelf Linux servers that use local disk for storage—no expensive SANs or high-end servers are required to run large data warehouses on Vertica (although it performs well using shared SAN storage if that's a preferred deployment route). Vertica both reduces hardware costs (often by up to 90% relative to other data warehouse databases) and improves the ability to answer more queries for more people against more data.

Besides remarkable performance on a variety of database workloads, Vertica includes several other features designed to offer performance, scalability, reliability, and ease of use. These include:

1. A **shared nothing, grid-based database architecture** that allows Vertica to scale effectively on clusters of commodity CPUs.
2. A **hybrid data store**, where newly inserted records are added to a write-optimized portion of the database to allow continuous, high-performance insert operations concurrently with querying to enable real-time analytics. In fact, in December 2008, Syncsort, HP and Vertica teamed up to break the world record[1] for data warehouse data loading. It took 57 minutes 21 seconds to load 5.4TB of TPC-H data (generated retail sales data) into Vertica by the Syncsort ETL software running on an HP BladeSystem C7000, which beat the previous record set by Microsoft (2.36TB in 1 hour).
3. **Automated physical database design tools** that recommend how data should be organized both locally on each node in a cluster, as well as horizontally partitioned across a cluster. In addition to choosing projections and sort orders, these tools ensure **k-safety**, meaning that all data is replicated on multiple nodes so that k node failures can be tolerated by the system without interrupting functionality. These tools reduce administrative costs by simplifying physical database design decisions. They also allow Vertica to automatically adapt to on-the-fly the addition or removal of database nodes.
4. **High-performance, ACID-compliant database** system with a light-weight transaction and concurrency control scheme that is optimized towards loading and querying data. Vertica's failure recovery model is based on replication (k-safety) rather than traditional log-based methods.
5. **Flexible deployment options –**
   a. Downloaded and installed on Linux servers of your choice
   b. Pre-configured and shipped on HP BladeSystem hardware
   c. Licensed and used on an on-demand basis, hosted in the Amazon Elastic Compute Cloud (EC2).
6. **Monitoring and administration tools and APIs** for controlling performance, backup and disaster recovery, etc.

In the remainder of this white paper, we describe the basic architecture of the Vertica Analytic Database, and describe these key features of Vertica in more detail. With these features in mind, we then describe how it is that Vertica can provide such remarkable performance compared to other relational systems.

[1] Visit http://www.ETLWorldRecord.com for more information about the benchmark

Vertica Analytic Database Architecture and System Overview

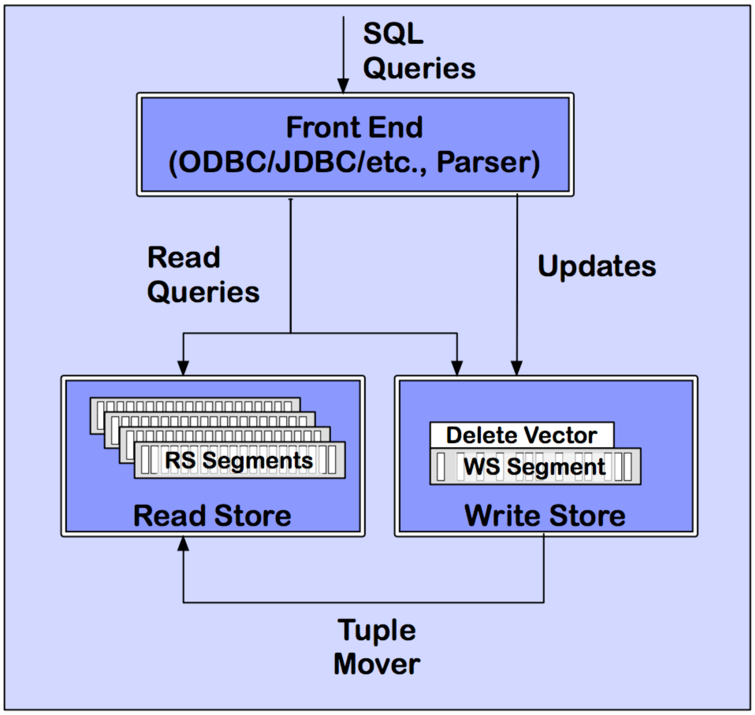
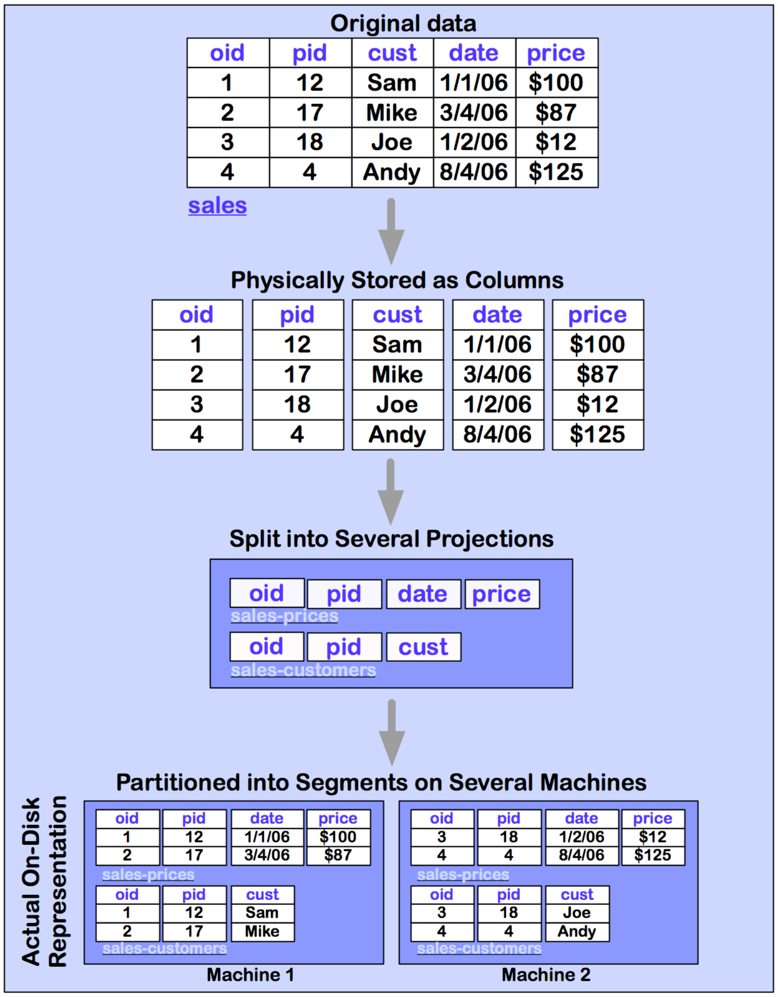


***Figure 1**: The Vertica Analytic Database Architecture*

Figure 1 illustrates the basic system architecture of Vertica on a single node. As in a traditional relational database, queries are issued in SQL to a front end that parses and optimizes queries. Unlike a traditional relational database, however, Vertica is internally organized into a **hybrid store** consisting of two distinct storage structures. The *Write-Optimized Store* (WOS) is a data structure that generally fits into main memory and is designed to efficiently support insert and update operations; the data within the WOS is unsorted and uncompressed. Conversely, the *Read-Optimized Store* (ROS) contains the bulk of the data in the database, and is both sorted and compressed, making it efficient to read and query. A background process called the **Tuple Mover** , moves data out of the WOS and into the ROS. Because it operates on the entire WOS, the tuple mover can be very efficient, sorting many records at a time and writing them to disk as a batch.

Internally, both the WOS and ROS are organized into columns, with each column representing one attribute of a table. For example, for a table of *sales* data with the schema (order-id, product-id, sales-date, customer-id, sales-price), Vertica would store this as at least five distinct columns (though they would logically appear as one table to the user). By storing these columns separately, Vertica is able to access only those columns that are needed to answer a particular query. Figure 2 illustrates how logical data in an example *sales* table is physically stored as columns. It also shows data being distributed across several projections and servers, which further improves performance, as we describe below.
If certain columns are always read together, e.g. address lines, region then these can be grouped together so they are retrieved in a single I/O.

In Vertica, each column may be stored in one or more ***Projections*** that represent partially redundant copies of the data in the database. For example, as shown in Figure 2, our sales table might be stored as two projections, one called *sales-prices with* the schema (order-id, product-id, sales-date, sale-price), and one called *sales-customers* with the schema (order-id, product-id, customer-id). Each of these projections also has a sort order that specifies how the data in the projection is arranged on disk. For example, the *sales-customers* projection might be stored sorted on customer id; this makes it particularly efficient for totaling all of the products that a customer bought. By storing several overlapping projections of a table in different sort orders, Vertica can be efficient at answering many different types of queries. Vertica's **Database Designer** automatically selects a good set of overlapping projections for a particular table based on the set of queries issued to that table over time.


Original data
sales
Physically Stored as Columns
Split into Several Projections
oid pid date price
sales-prices
oid pid cust
sales-customers
Partitioned into Segments on Several Machines
Actual On-Disk Representation
Machine 1
Machine 2

oid	pid	cust	date	price
1	12	Sam	1/1/06	$100
2	17	Mike	3/4/06	$87
3	18	Joe	1/2/06	$12
4	4	Andy	8/4/06	$125

Figure 2: *Logical tables are physically stored as columns, and partitioned into segments on several machines and in several different projections.*

It may seem that redundantly storing data in multiple projections is very wasteful of disk space. However, Vertica includes a suite of aggressive **column-oriented compression schemes** that allow it to reduce the amount of space a particular projection takes up in the ROS by as much as 90%. This also improves query performance by reducing the amount of data that must be read off disk. These compression schemes, described in more detail below, have low CPU overhead and allow direct operation by the database on the compressed data. This means that Vertica can often process queries over compressed data substantially faster than over uncompressed data.

Projection storage is physically distributed and lends itself well to parallelization in several ways. First, on a single machine, the ROS is horizontally partitioned into several *Storage Containers*. This allows the Tuple Mover to insert new data into the ROS without having to merge together and rewrite all old containers. To keep the total number of containers small, from time to time, the tuple mover merges these ROS containers together in the background. This process is illustrated in Figure 3.

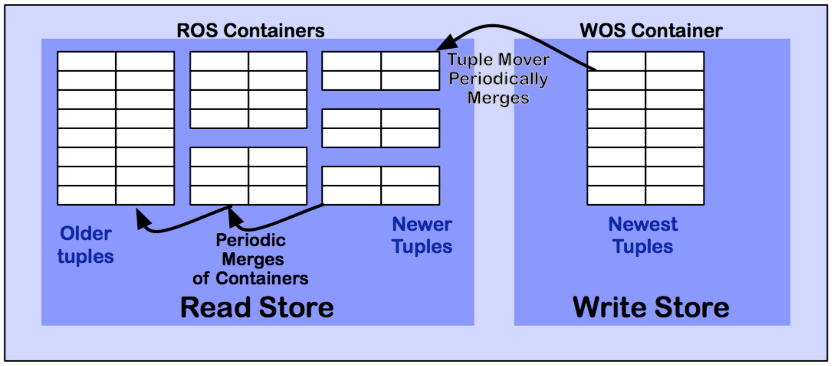


***Figure 3**: A hybrid storage architecture features a Tuple mover process that asynchronously moves batches of new data from a memory-based WOS to disk-based ROS containers*

Second, each projection is horizontally partitioned into *segments,* each of which is stored on a different machine in the cluster based on the value of one attribute in the table. This makes it possible to parallelize queries, allowing many applications to scale linearly with the number of nodes in the database. One possible partitioning across machines is shown for our example *sales* database at the bottom of Figure 2.

Now that we have covered the basic architecture of the Vertica database system, we look at the key features of the system that allow it provide **outstanding performance** as well as **simplify administration and reduce total cost of ownership**.

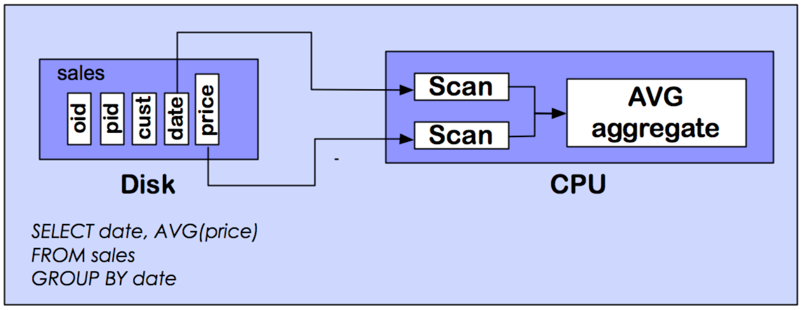
Performance Features

Vertica's performance on read-intensive workloads is due to a number of factors. To help understand its performance properties, let's look at the execution of a simple query over the *sales* table described above. Suppose we wish to compute the average sales price grouped by date in our *sales* database. The steps involved in processing this query are shown in Figure 4. There are two primary costs in processing a typical query: disk access and CPU cycles (in a distributed database, network I/O may also be significant). On most modern computers, it is possible to perform disk I/O in parallel with CPU operations. Thus, if answering a query takes D seconds of disk time and C seconds of CPU time, the total query time will be the maximum of C and D.

Column-orientation

Since many database queries are "disk bound," meaning D > C, the most obvious performance advantage of Vertica is that it only needs to read the two columns involved in the query from disk. In a row-store database, the disk scans all five columns. This extra three columns of scanning can represent as much as a 250% slowdown in performance. Since most databases have many more than five columns (hundreds, even), the performance of a row-store is often even worse.

Other database vendors include support for materialized views and data cubes, which are data structures designed to optimize the performance of a few common queries. They work by reducing the need to scan over unused columns by partially pre-computing query answers into data structures stored on disk. These approaches are optimized for answering a few queries well and are unable to provide comparable performance to Vertica across many different query workloads. Besides Vertica's column-orientation, one of the major ways in which Vertica achieves this performance advantage is via aggressive columnar compression, which we describe next.



```
SELECT date, AVG(price)
FROM sales
GROUP BY date
```

***Figure 4**: A query plan for a simple query in Vertica*

Aggressive Compression

In our simple model of database performance, compression helps to dramatically reduce both C and D. As an example, suppose the *sales* table accessed by the plan in Figure 4 is sorted by date, and then within each date by price. If a database contains 1 year of data, and each order has one of about 1,000 different prices on a given day, and there are 10 million total records in the database, there will on average be about 27 different orders from each of the 1,000 prices during each day. This means that, rather than representing the price column as a list of the form (price1, price2, price3, …), we can represent it as a *run-length encoded (RLE)* list of the form (<count, price1>, <count, price2>, ….), which will be about 27 times smaller than the initial representation. We can also compress the date column in a similar way, storing just 365 different <count, date> pairs to store the entire column of 10 million dates! Compressing data in this way greatly reduces the disk I/O (D). However, we also reduce the amount of CPU work, C, because database operators (like *scan* and *aggregate* in Figure 2) in Vertica are specially optimized to work directly with compressed data. For example, the *AVG* aggregate does not need to convert the RLE compressed list of prices back into uncompressed form, but can instead compute averages directly from the compressed records, reducing C by about a factor of 27.

Vertica includes delta encoding of successive values and an efficient Lempel Ziv implementation, that are well-suited to use with sorted columns with more distinct values as well as with unsorted columns. Also included are special compressions for float and time series data, as well as many other compression methods. Most of these methods allow direct operation on the data within the query processor. Other database systems – even those that have a column-oriented architecture – cannot operate on compressed data in this way.

Read-Optimized Storage

Recall that the bulk of the data in Vertica is stored in the ROS, which is optimized for efficient read operations. Keeping the ROS sorted and compressed is one way in which its structure is optimized. However, Vertica includes a number of other optimizations. For example, data in the ROS is *dense packed*, meaning that no empty spaces are left at the ends of disk pages; traditional databases leave a substantial portion of each page empty to allow insert operations to be performed without reorganizing all data on the disk. Vertica does not need to do this because inserts to the ROS are only done in bulk via the tuple mover. As another optimization, Vertica pre-fetches large blocks during most reads to avoid unnecessary seeks when large portions of several columns are being scanned.

Ability to exploit multiple sort orders

As described earlier, Vertica's approach to high availability and recovery involves the use of replicas which store the same data in different sort orders. Not only does Vertica's recovery scheme avoid expensive logging operations, but also, these additional sort orders can be used to further improve query speed. Furthermore, Vertica's aggressive compression schemes mean that different projections can be created on a single node that store the overlapping sets of columns in different sort orders. Vertica can then select the best sort order for processing a given query.

Parallel shared-nothing design on off-the-shelf hardware

Vertica's performance scales linearly to a large number of CPUs. Because Vertica does not depend on custom hardware, it will benefit over time from the dramatic performance improvements that commodity hardware has traditionally enjoyed. It is much harder for custom hardware solutions to reap these same benefits. Furthermore, because it is designed from the ground up to run on off-the-shelf, Linux-based servers, it will maintain compatibility and scalability over time.

Bottom Line

In summary, many vendors offer non-column store systems. For read-mostly workloads, these designs typically offer about 1/100th the performance of Vertica when using the same amount of disk space, or 1/10th the performance of Vertica in 10 times the disk space (for example, when they have many materialized views). A few column-oriented systems do exist, but these first-generation designs lack aggressive compression features, a hybrid WOS/ROS design, or Vertica's sophisticated recovery and replication strategies. Typically, Vertica offers 10 times the performance of these column-based systems in about ¾ the disk space.

The table below illustrates the performance wins in a typical data warehouse application. The study comes from a Vertica customer in the financial services industry, and it compares Vertica to a popular row-oriented data warehouse DBMS. Here, queries against 1.5 terabytes of stock market trade and quote history are 270X faster when Vertica queries the data. Vertica allows them to answer 33x more queries per day, and data is available in real-time rather than being a day late. All of this can be achieved with hardware cost savings of over $1M (USD) less. Similar benefits can be reviewed for other industries and applications by visiting www.vertica.com/benchmarks.

	Row-Oriented Data Warehouse DBMS	Vertica® Analytic Database	Vertica Advantage
Avg Query Response Time	37 minutes	9 seconds	270x faster answers
Reports per Day	30	1000	33x more reports
New Market Data Availability	Next day	1 minute	Real-time views
Hardware Cost	$1.4M (2x 6 servers + SAN)	$50,000 (6 HP ProLiant servers)	1/28th of the hardware, built-in redundancy

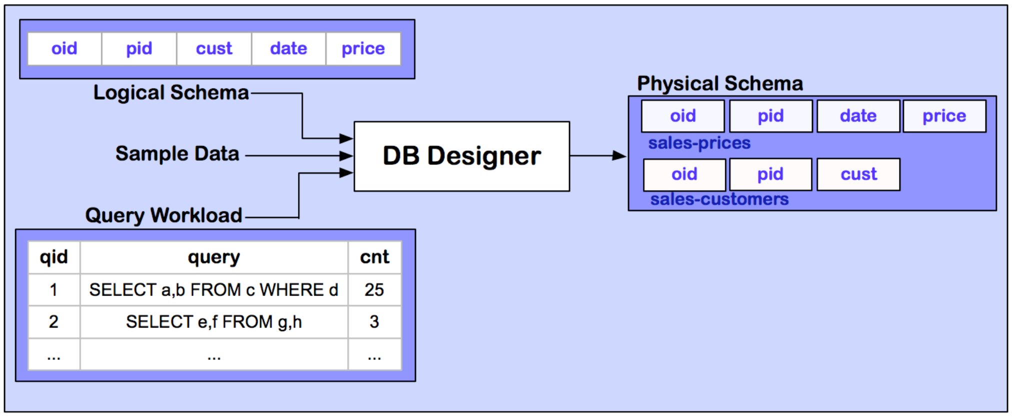
Administrative and Management Features

In addition to outstanding performance on a range of applications, Vertica includes a number of features designed to simplify database administration and reduce total cost of database ownership.

Vertica Database Designer

The primary role of the Vertica Database Designer (DB Designer) is to recommend a physical database design (including projections, compression and partitioning) that will provide good performance for the queries that the user most commonly issues over his or her database and cover any ad-hoc queries that users may choose to run. Of course, the database administrator is able to override any of the decisions made by the DB Designer, but in many cases, the designer can do a remarkably good job of optimizing performance, thus reducing the time administrators must spend on physical database tuning.

The DB Designer takes in a logical schema, a set of sample data from those tables and a "training set" of typical queries. Based on those inputs, it recommends a physical schema consisting of a set of projections and a partitioning of those projections across multiple machines that provides good performance on the training set. The recommendation ensures that any SQL query can be answered (which means all of the columns of the database must be stored in at least one projection) even in the event that *k* server nodes fail (i.e., recommends a k-safe design). Figure 6 illustrates the basic process.



qid	query	cnt
1	SELECT a,b FROM c WHERE d	25
2	SELECT e,f FROM g,h	3
...	...	...

Figure 6: *The DB Designer produces a physical schema consisting of a set of projections and a partitioning of those projections from a logical schema, a query workload, and a collection of sample data.*

The number of projections produced can be controlled by an administrator based on their query and data loading performance requirements. In general, it may take more time to load more projections. Typically, excellent performance can be achieved by a 2-projection design.

Furthermore, the DB Designer can be run after a Vertica database has been deployed into production. The DB Designer is able to re-design the database incrementally to iteratively optimize it for the current workload as it changes over time.

Recovery and High Availability through k-Safety

The traditional method to ensure that a database system can recover from a crash is to use logging and (in the case of a distributed databases), a protocol called *two-phase commit*. The main idea is to write in a sequential log a *log record* for each update operation before the operation is actually applied to the tables on the disk. These log records are a redundant copy of the data in the database, and when a crash occurs, they can be replayed to ensure that transactions are *atomic* – that is, all of the updates of a transaction appear to have occurred, or none of them do. The two-phase commit protocol is then used to ensure that all of the nodes in a distributed database agree that a transaction has successfully committed; it requires several additional log records to be written. Log-based recovery is widely used in other commercial systems, as it provides strong recoverability guarantees at the expense of significant performance and disk space overhead.

Vertica has a unique approach to distributed recoverability that avoids these costs. The basic idea is to exploit the distributed nature of a Vertica database. The Vertica DB Designer ensures that every column in every table in the database is stored on at least k+1 machines in the Vertica cluster. We call such a database *k-safe*, because if k machines crash or otherwise fail, a complete copy of the database is still available. As long as k or fewer machines fail simultaneously, a crashed machine can recover its state by copying data about transactions that committed or aborted while it was crashed from other machines in the system. This approach does not require logging because nodes replicating the data ensure that a recovering machine always has another (correct) copy of the data to compare against, replacing the role of a log in a traditional database. As long as k-safety holds, there is always one machine that knows the correct outcome (commit or abort) of every transaction. In the unlikely event that the system loses k-safety, Vertica brings the database back to a consistent point in time across all nodes.

K-safety also means that Vertica is *highly available*: it can tolerate the simultaneous crash of up to any k machines in a grid without interrupting query processing. The value of k can be configured to provide the desired tradeoff between hardware costs and availability guarantees. Note, however, that adding additional machines to increase the k-value of a Vertica database not only improves availability but also improves performance. This is because all that is required for k-safety is that each column is replicated k times, but each copy of a column may be stored with a different sort order. As noted earlier, different sort orders are good for answering different queries.

It is instructive to contrast Vertica's high-availability schemes with traditional database systems where high availability is achieved through the use of active standbys – essentially completely unused hardware that has an exact copy of the database and is ready to take over in the event of a primary database failure. Unlike Vertica's k-safe design employing different sort orders, active standbys simply add to the cost of the database system without improving performance.

Because Vertica is k-safe, it supports *hot-swapping* of nodes. A node can be removed, and the database will continue to process queries (at a lower rate). Conversely, a node can be added, and the database will automatically allocate a collection of objects to that node so that it can begin processing queries, increasing database performance automatically.

Continuous Load: Snapshot Isolation and the WOS

In a traditional database system, adding new data is done in one of two ways. It can be inserted a record-at-a-time, which is convenient for users, since they can update the database whenever they need to, but substantially slows the database. This slowness comes from two factors: first, inserts usually require locking operations that potentially block other transactions running in the system and second, because *every* insert requires expensive updates to indices, materialized views, and cubes. Bulk loading is an alternative to record-at-a-time inserts, where the database system is taken offline for several hours so that new records can be added without acquiring locks and so that indices can be updated in bulk.

Vertica, in contrast, offers a **continuous load** feature that does not require the system to ever be taken offline but still provides excellent insert performance. This is achieved through two techniques. First, inserts do not affect the performance of most queries in the system. This is because read-only operations in Vertica usually run in a **snapshot isolation** mode, where they read a recent, consistent snapshot of the database. This snapshot is essentially “read-only” and cannot be modified by concurrent updates and deletes – therefore the read-only query does not need to hold any locks to block out concurrent insert or update operations. As a result, queries and updates can proceed concurrently without interfering with one another.

The second way in which Vertica achieves good continuous load performance is through the WOS. Since the WOS is unsorted and un-indexed, individual insert operations are fast. The tuple mover process which runs in the background, updates the (sorted and indexed) ROS very efficiently because it can apply all of the WOS operations in bulk (much as the bulk-load process does in traditional databases, but without taking the system offline).

Monitoring and Administration Tools and APIs

Finally, Vertica includes a set of administrative tools and APIs for monitoring and controlling the Vertica databases. These tools display graphically, both database and system performance and can also be used used to conduct database archival and restore for use in disaster recovery. These tools support parallel backup of multiple nodes to a single image stored on a remote disk (e.g., a SAN). They also support an incremental backup mode, where new data added to the database is automatically written to an archival image.

Who Should Use Vertica?

Vertica is a relational SQL database system that is best suited to read-intensive analytic database applications such as data warehouses and data marts. It is optimized for databases with ad hoc query and OLAP-style workloads that include a some update operations. However, because of the WOS and tuple movers described above, a large portion of the SQL operations can be INSERT operations. For database-backed applications that meet these requirements, Vertica offers a substantial performance increase over row-oriented OLTP databases, other column databases and even proprietary data warehouse appliance hardware, while using significantly less disk space and without requiring a large investment in hardware or annual database administration overhead.

Additional Resources

The Vertica Analytic Database supports SQL and integrates with 3rd-party ETL, analytic and BI reporting tools and applications via JDBC, ODBC, ADO.NET and specific language bindings. Therefore, using all your existing SQL knowledge and technology, a Vertica database can be very quickly created and loaded with data.

If you would like to learn more about the Vertica Database or if you would like to evaluate it yourself, then visit the following links:

Gartner on Vertica and EDWs	www.vertica.com/gartner	Watch a recording of Don Feinberg of Gartner explain why supplementing an EDW with column-DB based data marts improves the ROI on EDWs
Resource Library	www.vertica.com/resourcelibrary	White papers, demos, webcasts, system requirements
Vertica Benchmarks	www.vertica.com/benchmarks	See customer-submitted cost and performance comparisons between Vertica and other databases
	www.etlworldrecord.com	Vertica was used to set the data warehouse data loading world record – 5.4TB in 57 minutes
Vertica for the Cloud	www.vertica.com/cloud	Get a Vertica database instance provisioned instantly on the Amazon Cloud and use it on a month-to-month basis
Vertica Customers	www.vertica.com/customers	See who's using Vertica
Evaluate Vertica	www.vertica.com/download	Request a free evaluation copy of the Vertica Analytic Database to download and install

About Vertica Systems

Vertica Systems is the market innovator for high-performance analytic database management systems that run on industry-standard hardware. Co-founded by database pioneer Dr. Michael Stonebraker, Vertica has developed grid-based, column-oriented analytic database technology that lets companies of any size store and query very large databases orders of magnitude faster and more affordably than other solutions. The Vertica Analytic Database's unmatched speed, scalability, flexibility and ease of use helps customers like JP Morgan Chase, Verizon, Mozilla, Comcast, Level 3 Communications and Vonage capitalize on business opportunities in real time. For more information, visit the company's Web site at http://www.vertica.com.