



The Event Analysis and Retention Dilemma

*The critical challenge of managing and making use of
ever-increasing amounts of event log data for
real-world compliance audit and investigation*

The Event Analysis and Retention Dilemma

WHY MANAGE MORE EVENT DATA?	4	EVOLUTION OF STORAGE	10
INCREASED SOPHISTICATION OF EXTERNAL SECURITY THREATS.....	4	FIRST GENERATION: DIRECT ATTACHED STORAGE	10
INCREASED SOPHISTICATION OF INTERNAL SECURITY THREATS.....	4	SECOND GENERATION: STORAGE ARCHITECTURES.....	10
COMPLIANCE WITH GOVERNMENT REGULATIONS.....	4	THIRD GENERATION: DATA TYPE SPECIFIC STORAGE ARCHITECTURES	10
CORPORATE GOVERNANCE.....	4	THE SENSAGE SOLUTION	11
BUSINESS GROWTH	4	HIGH PERFORMANCE SEARCH	11
EXPANSION OF AUTOMATION.....	5	HIGH VOLUME LOADING	11
HETEROGENEOUS GROWTH.....	5	HIGH VOLUME AND LOW COST STORAGE	11
INCREASED SYSTEM COMPLEXITY.....	5	LOW COST OF OWNERSHIP.....	11
INCREASED AWARENESS OF AN EXPANDED EVENT DATA MANAGEMENT SOLUTION SET	5	INCREMENTAL SCALABILITY	11
USING RELATIONAL DATABASE MANAGEMENT SYSTEMS TO RETAIN AND ANALYZE EVENT DATA 5		HIGH AVAILABILITY	11
TRANSACTIONAL DATA	6	DATA PROTECTION	11
ISOLATED CONCURRENT ACCESS	6	SENSAGE ARCHITECTURE.....	11
INFREQUENT SCHEMA CHANGES	6	CLUSTERED PARALLEL DISTRIBUTION.....	11
PRECISION QUERIES	6	<i>Incremental Scalability</i>	11
UNIQUE CHARACTERISTICS OF EVENT DATA	6	<i>Distributed Loading</i>	12
NON-TRANSACTIONAL.....	6	<i>Distributed Search</i>	12
TIME-BASED	6	<i>Distributed Aggregation</i>	12
FIELD REPETITIVENESS	6	<i>Data Redundancy</i>	12
TIME-BASED ARCHIVAL OR REMOVAL	6	KEY SENSAGE ARCHITECTURAL ELEMENTS:	12
VARIABLE SEARCH REQUIREMENTS	6	EVENT DATA SPECIFIC STORAGE ORGANIZATION	13
EVOLVING SEARCH REQUIREMENTS.....	6	<i>Time-based organization</i>	13
STREAM-BASED LOADING.....	7	<i>Column-based Compression</i>	13
DATA PROTECTION	7	<i>No Indices</i>	13
SYSTEM WIDE HIGH AVAILABILITY	7	NON-TRANSACTIONAL MODEL	13
CONTRASTING RELATIONAL DATA WITH EVENT DATA	7	<i>No Concurrency and Locking Overhead</i>	13
STORAGE BARRIERS OF RDBMS MANAGED EVENT DATA	8	<i>No Transaction Log</i>	13
INCREASED STORAGE REQUIREMENTS	8	CONCLUSION	14
INCREASED CPU REQUIREMENTS	8	ABOUT THE AUTHOR	14
INCREASED I/O REQUIREMENTS	8	APPENDIX A – STORAGE REQUIREMENTS COMPARISON.....	15
GEOMETRICALLY DECREASING LOAD PERFORMANCE	8		
SEARCH SPECIFIC OPTIMIZATION.....	8		
SEARCH OPTIMIZATION AND LOAD RATE TRADE-OFF	9		
MITIGATING RDBMS EVENT DATA MANAGEMENT BARRIERS	9		
DATA FILTERING	9		
LIMITED TIME RANGE SEARCHES.....	9		
LIMITED COMPONENT MONITORING.....	9		
EVENT STORAGE LIMITED BY CAPACITY	10		
TWO-TIER STORAGE ARCHITECTURE.....	10		
TIME-BASED DATABASE SEGREGATION	10		

Introduction

Corporations deploy information technology (IT) applications to achieve greater efficiencies in internal operations, and to provide better service through online access to their customers and partners. As corporate IT systems continue to expand in size and complexity, so does the need to effectively monitor and manage these systems. The objective is to enhance response time, achieve maximum availability, and lower costs – all while reducing security risks and complying with government regulations.

The “event” is the fundamental data unit used for system monitoring. Events are also commonly referred to as “logs.” Each system component generates events that indicate something of significance has happened for that component. System components include perimeter network components, internal network infrastructure components, security devices, application middleware, business applications, and databases.

Until now, enterprise event data has been selectively collected and sampled, or collected but never used or maintained. However, for continuous-process enhancement, corporate governance, and compliance mandates, this situation is no longer tenable.

As a result, several strategies have emerged for using event data to better manage IT systems. Initially, event data was stored in log files and made available for visual inspection. System administrators analyzed it using time-consuming ad-hoc methodologies, such as home-grown tools and scripts. These methodologies grew more difficult, tedious, error prone and, in some cases, impossible to use. As event-data volume grew beyond the ability for manual methods to derive value from it, a variety of commercial log analysis tools were created. Initially, these log-analysis tools focused on exposing web-site access trends to improve the effectiveness of web sites for marketing and customer-acquisition purposes.

As security incidents became significant IT issues, events were used to detect, analyze and prevent security breaches. Now, event data is used in two fundamental ways. First, it is used to monitor the flow of events, correlate events in real time, and detect security-intrusion patterns – **security response**. Second, event data are stored for longer time periods providing historical trend analysis, investigation, compliance reporting and audit support – **security analytics**.

By storing and managing event data for longer periods, the way is paved for security analytics, forensic investigation, and root-cause analysis. By combining events from all system components into a central location, security staff can examine one homogeneous log instead of several heterogeneous ones. Consequently, an analyst is able to use his/her time more efficiently.

In addition to its use for security purposes, event data is also being used for system management to help monitor and improve the health of a system.

This white paper examines the demands, scalability challenge, and approaches concerned with managing, analyzing and long-term storing of events for the purposes of compliance, security and system management. This paper discusses trends and forces that are shaping event-data management and storage requirements. It explains why Relational Database Management Systems (RDBMSs) were initially adopted for managing event data. Then, the paper goes on to illustrate the inherent limitations of RDBMSs for enabling security analysis and retention. Specifically, the paper addresses aspects of event data that distinguish it from generic business data. In conclusion, it introduces SenSage's solution and describes its advantages for storing, managing and analyzing event data. The paper shows how the solution meets security compliance and investigation requirements within gigabit-class network environments.



Why Manage More Event Data?

Some companies are already experiencing a need to store and manage greater volumes of event data. Other companies do not have, or are not yet aware of, this need. For those who believe they do not have a need to manage large volumes of event data, this section will either a) help confirm that conclusion, or b) change that perception. The following are the main drivers for increasing volumes of event data.

Increased Sophistication of External Security Threats

As companies increase their ability to prevent and detect external threats, those posing the threats also get more sophisticated. External threats are increasing in complexity and length of duration. The time for conducting a successful attack has extended from hours, to days, to weeks, and in some cases to months. The ability to detect such threats is directly limited by the time

Here are the relevant conclusions of that paper:

- There must be no time gaps in event data.
- Event data for all related assets must be available.
- All original event data must be available; which means there should be no filtering, interpretation or aggregation.
- A chain-of-custody for event data must be shown.

Event data can be considered forensic evidence for some future criminal action. Any missing or filtered event data will be considered contaminated evidence. It would render the entire set of available event data both suspect and inadmissible. Compliance legislation such as Sarbanes-Oxley and the Gramm-Leach-Bliley acts affect all corporations. Other compliance laws such as HIPAA, FFIEC, FISMA, NISPOM, DCID and VISA CISP, affect specific vertical industries. The scope of compliance varies from business to business, but all companies are



External threats are increasing in complexity and length of duration. The time for conducting a successful attack has extended from hours, to days, to weeks, and in some cases to months.

range represented by the event data available for analysis. To keep pace with the ever increasing time range of attacks, security managers must have access to greater volumes of event data.

Increased Sophistication of Internal Security Threats

Internal security threats are often more serious and costly versions of external threats. The person conducting an internal threat has more information, more authorized access points, more awareness of the value of corporate assets, more knowledge of IT infrastructure and most importantly, more time. Historically, security management has been mostly focused on external threats - even though the greatest financial and legal risks come from inside! To detect internal threats, one has to analyze event data over a longer period, and that analysis must include both authorized and unauthorized access events. Access to event data is crucial for any company hoping to strengthen its ability to manage internal security threats.

Compliance with Government Regulations

To comply with government regulations, companies face legal mandates about the quantity and quality of event data that must be captured, stored and made accessible. This relationship between compliance and event data results in increased needs for event data accessibility and storage. For a more complete treatment of this topic, see SenSage's white paper *Using Log Files in Digital Forensics and Compliance*.

required to comply with some level of regulation. Many government compliance mandates are beginning to require enterprises to store and analyze greater volumes of event data over long periods of time.

Corporate Governance

Recent high-profile court cases show that high-level executives can be held accountable for corporate governance failures. In many cases, pleading ignorance is no longer an acceptable defense. With executives being held directly responsible for corporate wrong-doing, they are motivated to demand greater corporate control and visibility into a company's inner workings. Corporate complexity requires that IT be part of the corporate governance process. When event data management is limited, one can achieve only component-level corporate visibility. With centralized management of high-volume event data, one can achieve cross-functional control and audit.

Business Growth

Corporations experience business growth for a variety of reasons such as success in the market place, expansion into new markets, and mergers and acquisitions. In all cases the IT infrastructure must grow to match the expansion of business. The growth results in the addition of more system components, more types of components and ultimately, more system events. So even if a business wishes only to maintain existing system and security management capabilities, there is an inevitable increase in the volume of events it generates.

Expansion of Automation

To reap the benefits provided by IT efficiencies, enterprises must seek to automate more of their operations. However, increased IT infrastructure automation also increases the volume of events produced.

Heterogeneous Growth

With acquisitions and mergers, businesses must manage the unanticipated combination and integration of disparate systems. As businesses grow, they must integrate new generations of technology with legacy systems. These heterogeneous growth requirements spur the need for centralized management to provide a comprehensive view. Heterogeneous growth geometrically increases the number of actual combinations of interdependent system components. All of this results in an increased volume of events produced, and an increased need to correlate varieties of events.

created. Until the Internet was fully adopted, business models such as amazon.com, eBay and Google would not have worked. So too, the availability of high-volume event-data management technology will perpetuate the creation of more effective solutions capable of even more efficiently handling of the growing volumes of enterprise event data.

***Example:** A Fortune 500 financial services company initially purchased a high-volume event management solution to detect external security threats. As the company became aware of the reliability and scalability of their event-management system, they extended its use to implement Sarbanes-Oxley compliance, and later to perform system performance analysis and tuning.*

Using Relational Database Management Systems to Retain and Analyze Event Data

Through more than twenty-five years of Relational Database Management Systems (RDBMS) technology

Initially, using RDBMSs to manage event data met enterprise requirements. But as the demand to manage greater volumes of event data emerged, the limitations of RDBMSs became apparent.

Increased System Complexity

Effective system management yields optimal performance, minimum response time, maximum utilization of IT assets, rapid response to failures, and fulfillment of service level agreements. As system complexity increases, these system-management goals cannot be achieved only by component-level monitoring and restricted time-span analysis. Additionally, event-data filtering assumes that one knows in advance where system problems lie, and does not allow for post-mortem analysis of unanticipated scenarios. As system complexity grows, the need increases to monitor the interdependency of system components and conduct analysis within longer time spans. Therefore, the ability to collect, store and analyze event data spanning many components for longer periods of time increases the efficacy of system management. Regarding event data, system and security management have redundant needs, and larger available volumes of event data have a multi-dimensional positive impact on corporate IT governance and effectiveness.

development, RDBMSs have become the preferred and “safe” choice for almost all data-management problems. Through IT’s broad-based adoption of RDBMS technology, data-management problem solving has become primarily relational data model driven. In other words, no matter what form the data originally appears, the first step in creating a data management solution is to evaluate the data using the relational model. This process has been successful in the vast majority of cases, so the relational model’s acceptance has become self-perpetuating.

As Security Information Management (SIM) vendors discovered the need to manage event data beyond real-time requirements, they followed the well-established practice of incorporating event data into the relational model and using RDBMS technology to store and analyze it. As a result, SIM vendors could focus on other areas of concern, such as real-time event correlation, mitigation and user-friendly presentation.

Increased Awareness of an Expanded Event Data Management Solution Set

Popular solutions are inherently limited for managing event data. Many accept these limitations and lower their expectations with regard to the value of event-data management. However, more robust solutions with quantum advances in handling volume and scale are increasing those expectations. There are many examples of this in IT history. Until affordable PCs were available, personal productivity tools such as word processors and spreadsheets would have never been

Initially, using RDBMSs to manage event data met enterprise requirements. But as the demand for managing greater volumes of event data emerged, the limitations of RDBMSs became apparent.

To understand why RDBMS technology now presents event-data management obstacles, one needs to understand the foundational requirements that drive RDBMS technology.

Transactional Data

RDBMSs are designed to support the commit/rollback protocol which dictates only complete transactions (data changes) will be permanently stored and visible. Well-designed applications have very small transactions that take microseconds to complete. Any data stored in an RDBMS database can be changed within a transaction. To support this, RDBMSs have elaborate logging sub-systems that log every change in order to be prepared in the event a transaction rollback occurs.

Isolated Concurrent Access

RDBMSs present a virtual view of the data, so a given user only sees committed data, or data that the user has changed. Although other isolation levels are supported,

Non-transactional

Event data is non-transactional. It is meant to be stored, searched, removed and archived. Once event data is stored, it will never be updated. In fact, for compliance purposes, altering and deleting event data should be strictly prohibited. Although there are some transactional semantics for event data, they are not the same, nor as stringent, as those for relational data.

Time-based

Event data is a collection of data about a particular event, at a specific point in time. Thus, every event will have a time stamp associated with it. Any search on event data is likely to have some time boundary.



To understand the mismatch between event data and RDBMS technology, it is helpful to understand unique event data characteristics.

the concurrent isolation paradigm requires synchronicity and locking sub-systems at the row level.

Infrequent Schema Changes

Before data can be loaded into an RDBMS, a schema must be in place that defines the semantics and existing data relationships. This requires that a data model must be complete before an application can be created. Relational schemas are generally static, or evolve slowly. Infrequent changes in application requirements drive schema changes.

Precision Queries

RDBMS are designed to optimize precision queries on structured data. This means precise information is known about the data before a query is formulated, and the data itself has been structured to fit into a predetermined model or schema. Often, queries are statically stored and optimized, with the query data being variable. RDBMSs are best optimized for unique key queries, such as customer number or invoice number. RDBMSs are not optimized for range, or pattern-matching style, queries. Examples of this are "list all invoices over \$100,000" or "list all companies with '.com' in their names." These query types usually involve a complete table scan. RDBMS databases must be tuned to support a specific set of applications. The tuning is accomplished through indirect references to data, such as indices; or data organization, such as data clustering. Therefore, once a database is tuned for a specific set of applications, it can easily become less optimal for other applications.

Unique Characteristics of Event Data

By understanding the objectives of RDBMS technology and the characteristics of event data, one can show how well event data maps onto RDBMS technology.

Field Repetitiveness

Event data is generally highly repetitive. For example, a company will have a relatively small set of authorized users. Event data for successful connection events will repeat this small set of authorized users over and over again. Other examples of highly repetitive data are URLs, IP addresses, and IDS signatures.

Time-based Archival or Removal

All event data is eventually removed, or archived, based on aging, as determined by system or security management requirements.

Variable Search Requirements

Searches on event data can be precise or pattern-oriented. For example, one search may require a precise matching of a user name while another may be based on URL patterns such as "hotmail.com." Although both kinds of searches must be supported, most log data is unstructured and must be searched with some form of pattern matching. The storage of event data cannot be organized to optimize precision searches. The unstructured nature of event data resists query optimization using indices.

Evolving Search Requirements

Event-data search requirements evolve with the security and systems management landscape. New searches must be created to detect newly discovered security threats, or to monitor new system components. This can happen daily or weekly. During a forensic process, the results on one search will determine the nature of the next search. Because of the unpredictable nature of rapidly evolving search requirements, it is insufficient to have an event storage system optimized for current requirements to the detriment of future search requirements. In contrast, RDBMS databases have query requirements

which are static, or evolve slowly with the gradual introduction of new application requirements.

Stream-based Loading

Event data is created in real time and must be loaded as fast as it is created. Although all event data does not need to be available in real time, the load rate must keep pace with the creation rate in the long run. Unlike relational data, the load rate cannot be slowed down by reducing user response time. System components create event data at a rate based on their usage, independent of the event data load rate.

Data Protection

Event data must be protected from being changed or destroyed by any source, including applications, users, and component failures. Destruction or modification of

event data can result in non-compliance with government regulations, increased security risk, or failures in system management. Although relational data has similar data protection requirements, in many cases these requirements are not as absolute, and protection failures don't have the same pervasive impact.

System Wide High Availability

Security and system management, designed to use event data, become reliant on event data. Should that event data become unavailable, operations may have to be shut down to prevent security breaches or non-compliance. In contrast, loss of availability of relational data may stop the operation of a group of applications or segment of the system, yet allow other independent operations on the same system to continue.



Contrasting Relational Data with Event Data

In the previous sections this paper described the nature of event data, and the dominant design objectives of RDBMS technology. Clearly event data and RDBMS technology are, at best, poorly matched. Below is a side-by-side comparison of standard relational data and event data. This comparison illustrates the stark contrast between these two kinds of data.

Area of Concern	Event Data	Relational Data
Transactions	Limited transactional requirements	Fundamental requirement
Isolation Concurrency	Stored data is never updated so all stored data is available to all users (subject to authorization filtering)	Must present an isolated virtual database view to prevent visibility of non-committed data
Schema	Must be general and flexible to accommodate future event types. Semantics of data are often determined at search time.	Generally static and must be determined before data can be stored and accessible by applications
Search	Search criteria can be precise or pattern based and search requirements evolve rapidly	Search criteria are precise and databases are optimized to support a known set of queries Search requirements are static or evolve slowly
Time Attribute	Time attribute is part of every event and is usually a key criterion for search	Time is one of many possible attributes and may not be present
Life Cycle	All event data is eventually removed or archived based on aging and retention rules.	Has no consistent life cycle requirements Most data has an indefinite life span.
Data Protection	Data protection to prevent destruction and modification is absolute and must be supported regardless of user access or component failure Data protection failures may cause compliance failures.	Data protection is customized per application and is based on database security authorizations and application business logic.
High Availability	Lack of availability is likely to impact the entire system and may have legal ramifications.	Lack of availability may only impact a segment of the system.
Load Rate	Load rate must keep pace with event data creation rate System components generating event data do not wait for event data loading	Load rate is based on user response time Increased response time reduces the load rate requirements

Storage Barriers of RDBMS Managed Event Data

The mismatches between event-data management requirements and RDBMS core technology strengths can be roughly depicted as either a) RDBMS over-head not required to manage event data or b) lack of technology to support the unique requirements of event-data management.

Industry experience and empirical evidence show that these differences are manifested in real failures to meet compliance, security and system management requirements. These failures can result in: failing to meet audits, increased security risks, and increased IT infrastructure costs.

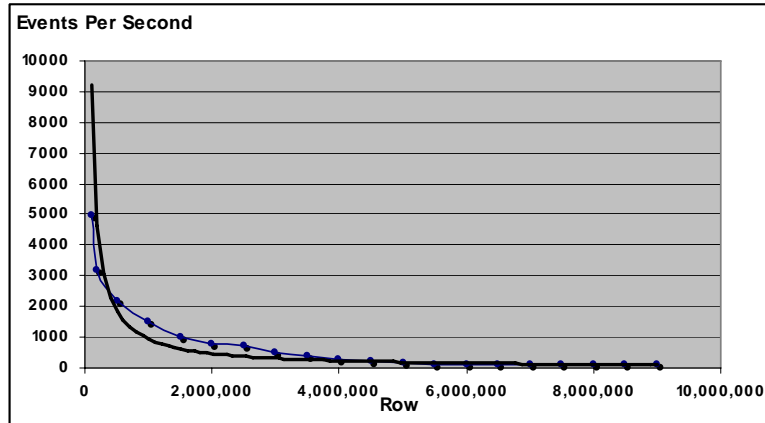
There are multifaceted problems in using an RDBMS to manage event data. And, companies using RDBMS technology in this manner are likely to experience most, if not all, of these problems.

Increased Storage Requirements

To support the commit/rollback protocol, RDBMSs maintain a transaction log that will allow for the potential rollback of non-committed updates. To optimize precision searches for specific applications, RDBMSs support the creation of special access data structures, such as b-tree

Geometrically Decreasing Load Performance

As the volume of data increases, the cost and time to load the next row of data increases geometrically. This is due to maintaining indices. RDBMS are optimized for precision searching of data, which is loaded one transaction at a time; and not for bulk data loading. The graph below shows the dramatic decrease in load rate as the volume of events increases.



The rate decreases to fewer than 300 events per second when the event volume reaches 2 million. Because of



Industry experience and empirical evidence show that these differences are manifested in real failures to meet compliance, security and system management requirements.

indices and hash tables. Some RDBMSs support isolation concurrency through the storage of lock information along with the data. To support data versioning, multiple versions of the data are stored.

This is just a partial list of RDBMS features the result in increased storage requirements. The net result is the volume of RDBMS storage overhead may be up to three times larger than that of the original data, resulting in a 4 to 1 expansion of storage requirements.

Increased CPU Requirements

Row-level concurrency control requires that an RDBMS check a lock/transaction table for every row access. Most RDBMSs use a paging system to allow multiple transactions to share data and reduce I/O requirements. These and other internal structures use additional CPU cycles.

Increased I/O Requirements

Increased storage requirements result in increased I/O requirements. Indices must be maintained in tandem with every row update.

RDBMS indices, this loading-rate degradation is permanent. Although the absolute numbers may vary based on hardware configuration, the shape of the curve should be the same.

Search Specific Optimization

Search is optimized in RDBMSs by creating indices which anticipate a specific set of search criteria. For example, if it is anticipated that many searches will involve a user name, then an index will be created on the user name column. Searches which fall within the bounds of the anticipated search criteria will execute quickly. Searches that do not fall within the bounds of the anticipated criteria will cause a complete table scan.

Some common event-data searches cannot be optimized by an RDBMS at all. These include substring and pattern-matching searches. These also result in a complete table scan. For large databases with billions of rows, these searches can become multi-day queries rendering a result of no practical value.

Example: A customer attempted to perform a search against 1 billion web proxy events that would return timestamp, IP address, User ID and URL for all records where the URL contained a specific string.

Because this is a pattern search, the RDBMS was required to conduct a full table scan and could not take advantage of any indices. This resulted in a search that took days to complete using high-cost, massive multi-processor, high-performance servers.

Search Optimization and Load Rate Trade-off

To optimize a greater number of search scenarios, more indices must be created. The more indices that are created, the slower the event-data load rate becomes. This forces the database administrator to have to constantly maintain a balance between search performance and load performance. To do so, indices have to be created and dropped. Consequently, search scenarios must be anticipated and prioritized. If the event-data load rate becomes unacceptable, then indices must be dropped. Dropping an index can reduce search performance in ways that are hard to predict. For large volumes of event data, creating and dropping indices are major undertakings that can temporarily shut down database operations and take hours or days to complete.

Mitigating RDBMS Event Data Management Barriers

Faced with the limitations imposed by the RDBMS solutions for event-data management, SIM companies and their customers have adopted a number of strategies to mitigate RDBMS shortcomings. These strategies represent a valiant but costly effort to deal with RDBMS'

Example: Denied access is monitored, but successful connections are not. If a buffer overflow attack is being accomplished by a server making excessive outbound FTP connections, examining the event data collected will fail to identify the attacking server.

Additionally, government compliance requires that all original event data be available to establish full context and to ensure that there was no data tampering. In general, data filtering produces an incomplete record resulting in limited value of the event data collected.

Limited Time Range Searches

Limiting the time range requires less storage capacity. However, this approach ignores business imperatives that require longer-term search capability, and it may miss an attack that occurs over a long time period.

Example: For a particular company, only one week of event data is kept. A sophisticated attacker spreads pre-attack reconnaissance over a few weeks. Use of the available event data is unable to detect this low-and-slow attack.

Limited Component Monitoring

If one monitors fewer components, one needs less event-data storage. However, determining which components do not need to be monitored, as in data filtering, presumes knowing ahead of time the nature of future-event analysis.



Faced with the limitations imposed by RDBMS solutions for event-data management, SIM companies and their customers have adopted a number of strategies to mitigate RDBMS shortcomings.

event-data management disadvantages. If the sum of all of these strategies were successful, then this paper could end at this point. However, each strategy is either insufficient, risk-producing, or both.

Data Filtering

To reduce the amount of event data that needs to be stored, one can filter to reduce the number of events stored, and the amount of data that is stored for each event. Though allowing event data storage to span longer ranges of time, filtering creates harmful side effects. Filtering pre-supposes that the natures of searches needed in the future are known in advance. However, unanticipated security or system management scenarios may require data not stored, rendering limited value from any event data that is stored.

Discovery of new security and system management scenarios may point out the need to monitor non-monitored components. This creates a "missing link" that impedes a forensic investigation from uncovering root cause of a security breach. Compliance mandates typically require the monitoring of most components within IT infrastructures.

Example: A hospital uncovered a risk scenario related to leakage of its VIP patient information via web surfing from shared workstations also used for patient data access. To determine the root cause of the leakage, daily event data from windows logins, web proxies, DHCP and a patient management application needed to be correlated for a trailing week. However, collection of web-proxy monitoring event data was previously eliminated to reduce RDBMS storage requirements. This made it impossible to discover the source of this critical scenario.

Event Storage Limited by Capacity

The amount of event data stored is often limited by the available RDBMS storage capacity and load rates. When event storage reaches a pre-defined threshold, the oldest events are purged until event storage falls below the threshold. This strategy guarantees control over the amount of data stored and load rate achieved. But it sacrifices a predictable time range for available event data. A spike in activity would effectively reduce the time range of event data available for analysis.

***Example:** To catch low-and-slow attacks, the event storage policy is changed from a one-week to a three-month retention period. The RDBMS capacity is increased by 1,200% by purchasing additional disk capacity. But the event-data load rate declines to the point where load rate cannot keep pace with event data creation rate. Moreover, the increased capacity does not hold three months of event data because of the unanticipated non-linear increase in space*

into multiple searches, and the results manually aggregated.

Evolution of Storage

To reiterate, RDBMS technology was developed primarily to address business data which is transaction and record oriented. Event data is different than relational data. RDBMS technology is unable to adequately address the unique characteristics of event data, and attempts to force an RDBMS solution on event-data management often create significantly diminished performance and increased storage requirements. Strategies aimed at smoothing over the wrinkles simply produce even more issues.

There is a clear need for an event-storage technology that overcomes RDBMS shortfalls and leverages the unique characteristics of event data. This technology is not intended to compete across the board with RDBMS technology, but instead follows recent trends of purpose-



There is a clear need for an event storage technology that overcomes RDBMS shortfalls and leverages the unique characteristics of event data.

required for indices. As a result, the expected time span of collected data is not achieved, and the low-and-slow attacks can still remain undetected.

Two-tier Storage Architecture

To alleviate the high cost of RDBMS storage, aged events can be removed from the database and archived into lower cost compressed storage. Should events from the archive be needed, they must be uncompressed and restored to the database. Removal and restoration of event data from an RDBMS database is time consuming. It creates resource contention with other operational data loading, and may be manual operations requiring the administrator and system-administration resources. Also, compression algorithms are not sensitive to the repetitive nature of event field data and, therefore, only achieve standard compression ratios. While a two-tier strategy is a good approach for Information Life-cycle Management (ILM), it is no substitute for having adequate on-line event-data storage.

Time-based Database Segregation

To mitigate geometric degradation of event-data loading-performance, one can segregate event data into separate time-ranged databases. This effectively creates an event-time-based meta-index maintained by the user. It does create a sustainable minimum event-data loading rate. However, part of the search optimization burden is now shifted to the user. Searching is now more complex and has to consider which databases to search. What was once a single search must now be manually broken up

built storage solutions. As such, it reflects the advantages of the third-generation of storage-system technology.

First Generation: Direct Attached Storage

First generation architectures – direct attached storage (DAS) - consisted of a hard disk attached directly to a computer, with storage controlled by the computer's CPU. Today, greater than 95% of all computer storage devices (disk drives, disk arrays, and RAID systems) are directly attached to a computer through various adapters - via standardized software protocols such as SCSI, Fibre Channel and others. This type of storage is alternatively called "captive storage" or "server attached storage."

Second Generation: Storage Architectures

As architectures evolved, new ones began utilizing a combination of computers, network, and DAS that essentially provide virtual DAS for client computers.

Third Generation: Data-Type-Specific Storage Architectures

Third generation storage architectures are similar to generic data storage architectures, but are specifically designed to manage the unique characteristics of the type of data stored, such as event data.

This table outlines the evolution and classification of storage:

Storage Evolution				
First Generation	Standard spindle storage	DAS	Direct attached storage	IBM, Seagate
Second Generation	Storage architectures	NAS	Network attached storage	Network Appliance, IBM, HP, Maxtor, Quantum, EMC...
		SAN	Storage area network	HP, IBM, Hitachi, EMC...
Third Generation	Data type specific storage architectures		Content addressable storage	EMC...
			Email archive storage	EMC, Veritas/KVS...
			Security event analytics	SenSage

The SenSage Solution

SenSage delivers a high-performance, scalable means for organizations to centrally aggregate, cost-effectively store, and efficiently analyze massive volumes of event log data over long periods of time while retaining the original source data.

SenSage eliminates the standard RDBMS overhead, which is not needed to manage event data, and materially increases the performance and capacity to manage massively high volumes of event data.

This solution provides significant benefits to customers who need advanced event-data management.

High Performance Search

The SenSage solution executes searches in minutes or hours, whereas RDBMS searches often take hours or days.

High Volume Loading

SenSage enables gigabit-class network event-data loading to keep pace with enterprise-wide event collection, with no degradation based on the volume of data stored.

High Volume and Low Cost Storage

SenSage uses low-cost Linux servers to store highly compressed data. No expensive RDBMS licenses are required. Servers are more efficiently utilized due to the elimination of RDBMS overhead.

Low Cost of Ownership

This solution requires no administration. Data organization is simple and self-tuning.

Incremental Scalability

This solution is proportionately scalable. Adding servers enables capacity and throughput to match business growth.

High Availability

Built in redundancy allows continued operation even with a server failure.

Data Protection

Event data is protected against modification by all outside sources. Data redundancy protects against loss of data in the event of component failure.

SenSage Architecture

SenSage core technology is a combination of:

- Server clustering
- Event-data-specific storage organization, and
- A non-transactional model.

Clustered Parallel Distribution

SenSage leverages a clustered server architecture to distribute work-load and achieve parallel computing on a massive scale.

Incremental Scalability

SenSage distributes work-load equally among its clustered servers enabling concurrent processing. This allows SenSage to maximize the throughput and usage of each server. Storage and throughput capacity can be increased simply by adding more SenSage servers.

The resulting increase in throughput and storage capacity is directly proportional to the number of servers added. For example, adding a server to a two server cluster would increase storage and throughput capacity by 50%. Similarly, by adding two servers, one achieves 100% capacity improvement. Event-data management capacity scales directly with business growth. Cost and capacity planning are easily accomplished.

Distributed Loading

As data is loaded into SenSage, data is evenly and concurrently distributed across all the servers in the cluster. This maximizes load efficiency.

Distributed Search

Search requests received by SenSage are also evenly distributed on the SenSage servers. Each server conducts its portion of a search in parallel with the other servers. The final results from each server are aggregated and returned to the user.

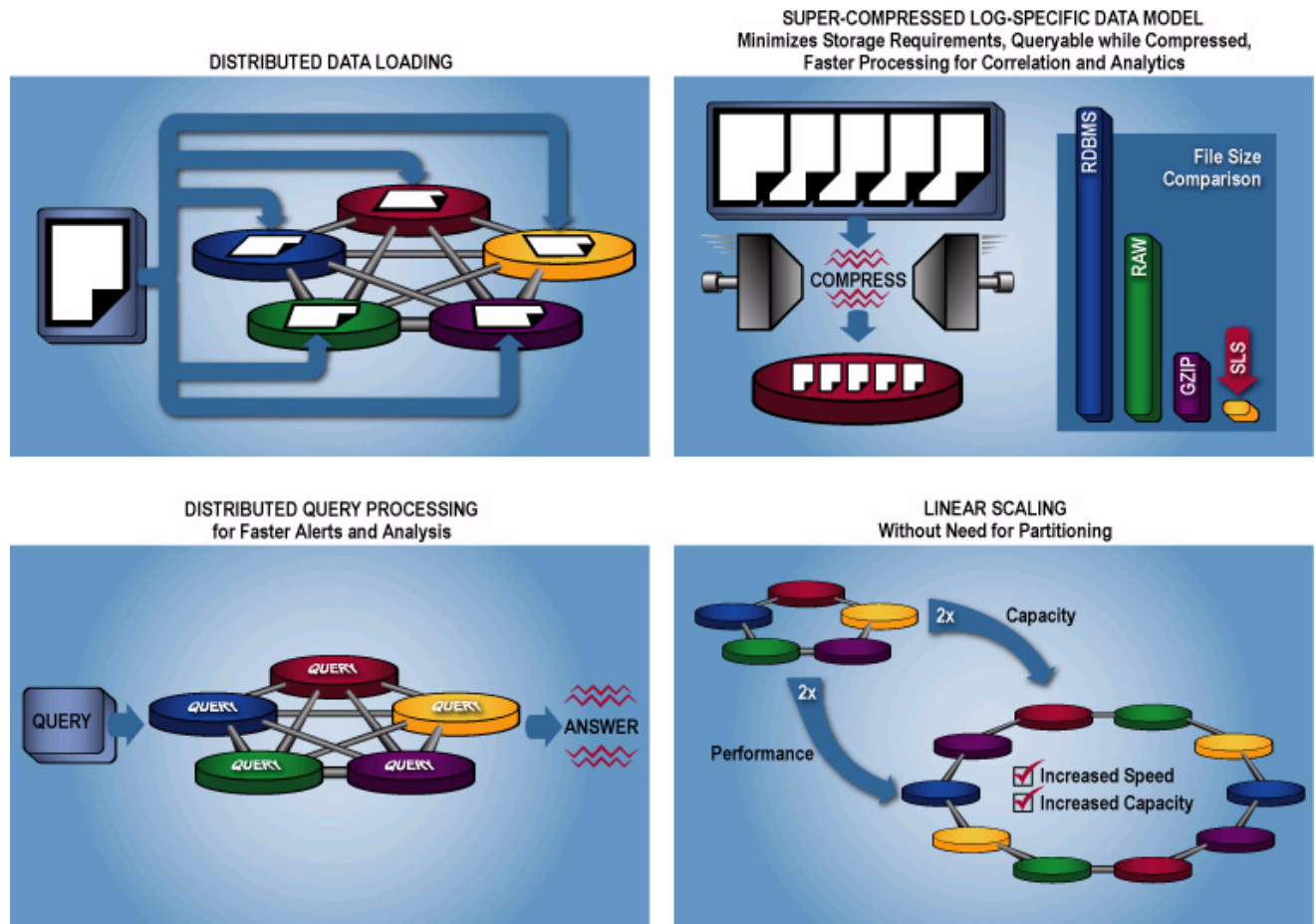
Distributed Aggregation

Aggregation searches that use 'GROUP BY' are distributed across the SenSage servers for complete parallel processing of the aggregation operation. The results of each server are then aggregated into a final result.

Data Redundancy

Every event is recorded twice in the SenSage server cluster. Each copy is stored on a separate server. Should a server fail, the server that holds the copy of the failed server's event data is designed to automatically take over all search operations for the failed server. Searches and loading continues with the throughput degraded by a factor proportional to the number of servers in the cluster. For example, in a five server cluster throughput is reduced by 20% should a single server fail. This design enables the SenSage architecture to provide high availability with marginal loss of performance.

Key SenSage architectural elements:



Event Data Specific Storage Organization

SenSage's storage organization is specific to the unique nature of event data, and produces significant advantages when managing that data.

Time-based organization

The data on each server in a SenSage cluster is partitioned in time ranges. This creates advantages at load time and search time. For load time, since event data generally has increasing time stamps, the likelihood is small of combining new load data with data already loaded. This dramatically reduces data reorganization needs. For search with time constraints, SenSage's search engine quickly eliminates the need for scanning data that does not meet the time constraints.

Column-based Compression

SenSage uses time-based organization, puts event data into columnar storage, and then compresses it when it is written to disk. High compression ratios are achieved because of the repetitive nature of event data within a column. Compared to the volume of data stored in an RDBMS database, SenSage security-event-storage can achieve up to a 40:1 compression ratio. (For a further detailed comparison see Appendix A.) Only the columns referenced in a search are decompressed and searched. As with time-based organization, this eliminates the need to decompress and scan large volumes of unnecessary data. The native storage format of our SenSage solution is compressed, with decompression only required after the event data has been selected based on time or column references. Because the basic unit of storage is a flat file, data removal and archival operations are simplified and extremely fast.

No Indices

Because of the unstructured nature of event data, indices render little value. SenSage provides dramatic search response time through distributed parallel searching, and event-data-specific data organization. This architecture requires no indices. Unlike an RDBMS, our solution does not need an administrator to create and drop indices to balance between search and load performance. There is also no overhead of index maintenance during loading for SenSage. This means the event-data load rate will remain constant, no matter how much data has already been loaded. Additionally, because no indices are needed, there is also no need for storage of index information. This substantially reduces storage requirements as compared to those by RDBMS-based SIM products.

Non-transactional Model

SenSage delivers unparalleled performance versus RDBMS-based SIM products, largely because of its non-transactional model. This is accomplished by minimizing overhead and optimizing use of computing resources.

No Concurrency and Locking Overhead

Because event data is never updated, our solution has no RDBMS overhead of row and table locking. Searches never need to wait for updates.

No Transaction Log

Since the commit/rollback model is not meaningful for event data, the SenSage solution avoids CPU, I/O and storage capacity overhead required to maintain a transaction log.

Conclusion

This paper shows how business drivers for security risk management, system management, and government compliance have all created a need to store and manage system event data - and the volumes of that event data are steadily increasing. Initially, SIM vendors adopted RDBMS technology as a preferred event-management solution. As the volumes of event data grew, RDBMS technology could not meet event-data management requirements.

SenSage recognized the tremendous volume, on-going mass, and unpredictable use of event data in future analytics. From the start, SenSage designed an event-centric, high-performance architecture that is able to collect an enormous amount of complete log data, at high velocity. This robust data management solution maps log sources to others, virtually at query, and provides flexibility to support a broad number of sources. It enables unparalleled precision and long-term search and trending, while significantly saving on storage capacity. Furthermore, clustering technologies provide our customers incremental scalability on load, query throughput, as well as data redundancy and capacity.

SenSage delivers a high-performance, scalable solution for organizations to centrally aggregate, cost-effectively store, dynamically monitor and efficiently analyze massive volumes of events over long periods of time, while retaining the complete original source data. This empowers organizations to respond to business threats, conduct thorough investigations, and fortify audit compliance processes.

About the Author



Bruce Scott Vice President of Engineering, SenSage, Inc.:

Mr. Scott currently directs product strategy, development, and customer support for SenSage. He brings over 25 years of experience at the forefront of building highly scalable data management technologies and founding several companies. Prior to SenSage Bruce was Founder and CEO of PointBase where he led the development of Java-based embedded database technology that enabled applications to manage, synchronize and extend data across networks of servers, desktops, laptop, and mobile/wireless devices. Previously, he founded and was Vice President of Database and Connectivity Research and Development for Gupta Corporation. There he invented SQLBase, the first commercially available PC-oriented client/server database and he patented

software development design paradigm for SQLWindows. As one of the four founders of Oracle, Mr. Scott was the Principle Engineer and Architect of the first three versions of Oracle. Bruce holds a BS in Computer Science from California Polytechnic State University.

About SenSage

SenSage, the leading provider of enterprise security analytics, offers unparalleled performance and a scalable means for organizations to centrally aggregate, dynamically monitor, efficiently analyze, and cost-effectively store massive volumes of event log data. Our solutions empower companies to readily respond to business critical threats, conduct thorough and precise investigations, and maintain compliant operations. Based in San Francisco, CA, SenSage currently protects Global 2000 customers in financial services, government, healthcare, manufacturing, and technology. The company markets its product directly and through partners including Cerner, EMC, Hewlett-Packard and Lockheed Martin. For more information, please visit www.sensage.com.

© Copyright 2005 SenSage, Inc. All rights reserved. SenSage and Scalable Log Server are trademarks of SenSage, Inc. in the United States. All other trademarks used herein are the property of their respective owners.

Appendix A – Storage Requirements Comparison

In this appendix, we compare the storage requirements of SenSage and an RDBMS database. Comparisons are based on the following assumptions:

Average original event data size	150 bytes
Working days per year	260 days
SenSage Compression Ratio	10:1
RDBMS Expansion Ratio	4:1

The expansion ratios listed above for RDBMS storage are conservative when compared to those where a greater number of indices are involved. The following table compares storage requirements in gigabytes based on these assumptions for three different sizes of companies:

Company Size	Millions of Events Per Day	RDBMS Daily Storage Requirements	SenSage Daily Storage Requirements	RDBMS Annual Storage Requirements	SenSage Annual Storage Requirements
Medium	15	9.0	0.2	2,340.0	58.5
Large	150	90.0	2.3	23,400.0	585.0
Global 500	1,000	600.0	15.0	156,000.0	3,900.0

This table illustrates the dramatic difference in storage requirements for SenSage. For a company that has a billion events per day it would require over 150 terabytes of RDBMS storage. This is well beyond the capability of current technology and it would be prohibitively expensive. On the other hand it would take about 4 terabytes of SenSage storage to store a years worth of events for a Global 500 company which is well within technological and cost parameters of such a company.