Myths About Historians

by Elliott Middleton, Product Manager
Schneider Electric Software

WONDER ON

Executive summary

Relational databases are ideal for many applications, but not the best solution for time-series data.

Historian applications have been around a long time, long enough so that certain preconceived notions, unfounded impressions, or 'myths' have developed through the years that seem to perpetuate without substance or evidence. If you want to better understand the merits and shortfalls of relational databases as well as uncover and challenge these myths in the context of process historian applications, then read on. It will also aid in understanding the basics of historian applications, discover what to consider during an evaluation, and properly set your expectations on the value a historian can bring.

Just because the New York Stock Exchange trading systems and other high-throughput applications use a relational database doesn't mean they're right for everything. Take time-series data, for example: can SQL Server or Oracle store time-series data? Certainly. Are there some problems you should know about before you try it? Absolutely.

Myth #1: Storage is so cheap that efficiency doesn't matter.

Be sure you really understand how much data a typical process actually generates. A modest 5,000 tag historian logging data every second generates 157 billion values per year. Stored efficiently in 8 bytes each, that's roughly a terabyte a year. In some tests comparing SQL Server storage requirements for time-series data with those of Wonderware Historian the difference was 50:1 when including the necessary indices. Even though storage prices are falling, 50 terabytes of data a year is still a lot. Also recognize that it isn't just a matter of having enough disk space to hold that much data—most historian applications also need that data protected, multiplying the amount of storage for backups or disk mirroring. Some industries even have regulatory requirements for several years' worth of data, further amplifying the storage need.

Myth #2: Relational databases are fast enough.

As hardware price-performance has improved relational databases have benefited. However, relational databases are designed to protect referential integrity around "transactions" that may update multiple table values in unison, which adds significant overhead. For example, on high-end hardware (running 64 Itanium processors) SQL Server 2008 established a world record 1126 transactions per second. Granted such transactions are not the same as those required for a historian, but even such high-end hardware would be taxed to store 5,000 values per second if each value was a transaction. This means a frontend buffering application must collect the data and stream numerous values into the database as a single transaction. Other databases without full transactional support, such as MySQL's freeware MyISAM storage engine, can support higher throughputs, but still require a frontend buffer to achieve adequate throughput for all but the tiniest historian applications.

Of course, the reason to store data in the first place is so that it is available to retrieve later. Naturally, that makes retrieval performance quite important, too. Particularly in general-purpose solutions like a relational database, it is possible to organize data so that it is either efficient to store (higher throughput) or efficient to retrieve (fast retrieval), but not both. Efficient retrieval of time-series data from a general purpose databases requires use of a "clustered index," something not available, for example, in the higher-throughput MyISAM storage engine.

In contrast, purpose-built storage engines designed specifically for time-series data leverage a knowledge of how data is collected and consumed to store it efficiently for both—this would not be possible if the data were more generalized.

Myth #3: Using a relational database as a historian is a new revelation.

There are frequently new ideas about how to apply existing technology—Post-It® notes were 3M's famous application of an existing not-so-sticky adhesive. However, companies have attempted using a simple relational database schema as a replacement for purpose-built time-series storage for over a decade. In spite of this "new idea" of using a relational database, the market for dedicated historian solutions has continued to grow significantly.

Myth #4: You can only use SQL to query data in relational databases.

Though relational databases have many advantages over alternative technologies, what catapulted them to prominence was the power of the Structured Query Language (SQL). SQL standardized and fundamentally changed how users can extract value from their data from being a complex programming exercise to being a relatively simple and flexible language to describe the data of interest.

Fortunately, it is very practical to adapt SQL to non-relational data stores and gain the tremendous benefits and power of SQL without some of the inherent limitations of a relational database.

Myth #5: There is nothing special about time-series data.

With all the power of Structured Query Language (SQL) to query data, some may claim that relational databases are just as good at retrieving time-series data as they are transactional data. It is certainly true SQL gives great flexibility, but it is based on some fundamental assumptions that don't apply to time-series data: a) there is no inherent order in the data records (in fact, time-series data is ordered by time), b) all the data is explicitly stored (in fact, most historian data only represents samples from a continuum of the real data), c) all data is of equal significance.

These differences are significant. For example, if an instrument reports a value timestamped at "7:59:58.603" and a user queries a relational database for the value at "8:00:00.000," no data will be returned since there is no records stored for precisely that time—the database does not recognize that time is a continuum. Similarly, if a temperature was "21.0 °C" and two-minutes later was "23.0 °C", it has no inherent ability to infer that halfway between these samples the temperature was approximately "22.0 °C".

In historian applications, it is rarely steady-state operations that are most significant. If the only way for a client application to find exceptions is to query all of the data for a measurement, it will place a heavy load on the overall system: server, network and client. In contrast, historians generally have means of filtering out insignificant data (based on comparing sequential records) to radically reduce the volume of data that must be delivered to client applications.

Myth #6: Managing time-series data in a relational database is trivial.

Relational databases are designed to accumulate massive amounts of data. However, as the amount of data grows, so do query execution times, the size of backups, and numerous other routine operations. To alleviate this performance problem of ever-growing tables, database administrators must routinely purge data from the database, rebuild indices and related operations. In any database that protects transactional integrity, this purge operation must suspend normal database updates—that's a problem for historian applications running 24 x 7 x 365. To even make the purge operation itself tolerable requires minimizing the amount of data maintained in the database.

In the event purged data is needed later (for example, in response to an audit or some regulatory demands), restoring the data is non-trivial. The generally recommended practice is to restore a full database backup that included the needed data, either to a separate system dedicated for this purpose, or to take your production system offline and use it. This is even more problematic if the required data isn't available within a single database backup—for example, if you only maintain the last 30 days of data in the online database and the audit requires 90 days of data., you must either manually merge all the data into a single database, have three systems, each with an isolated 30-day window, or examine each backup serially.

True historians, on the other hand, are designed to both handle the rapid growth in data and to provide simple means of taking subsets of the data offline and online.

Myth #7: The only options are fully relational or fully proprietary historian solutions.

While it's true that most historian solutions either use fully proprietary technology to address the inherent limitations of relational database or fully leverage relational database to reduce their own engineering costs, Wonderware Historian actually delivers the best of both worlds. It relies on a solid relational schema for managing all the relatively static configuration data, but extends the native transactional storage engine and query processor of Microsoft SQL Server with proprietary extensions to address their limitations for time-series data.

Building on Microsoft SQL Server delivers a solution that is easier to secure and manage than fully proprietary solutions, but without compromising on the fundamental capabilities required in a historian.

Myth #8: There is nothing special about industrial applications.

True historians provide facilities for dealing with the demanding, real-world realities of industrial applications that are outside the realm of pure relational databases. How do you intend to make use of the data? Do you need to convert rates into quantities for reporting? If so, that is quite complex with a SQL query. Is your instrumentation and data collection 100% reliable, or do you sometimes have to "make do" with data that includes instrument errors? While general-purpose databases can certainly store data, they aren't designed to incorporate notions of "data quality" into calculations and aren't able to simply perform routine time-series calculations such an integral calculation that are commonly needed.

Myth #9: Relational database applications aren't historians.

A "historian" addresses several related functions: continuously collecting real-time data, storing noteworthy subsets of that data, and providing a means of extracting meaningful information from that data. Whereas "historian" describes an application, "relational database" names a technology. Though there are certainly some significant challenges in using a relational database technology for time-series data, just because an application uses one doesn't mean it isn't a historian—it can still be a historian, just a fairly basic one. Rather than focusing on the underlying technology choices (relational databases or proprietary files), focus on the functionality needed—that requires an understanding of the overall application and involves much more than simply storing data.

Myth #10: All data is equal in importance and validity.

To a relational database, a stored value is precisely that, a value, and is always assumed to be valid—if it isn't, it is up to someone to correct it. In collecting millions of samples from thousands of data points from around a process, it is inevitable that some information is incorrect or missing. There may be issues with measurement equipment where values are out of range, communications was lost, or the data was simply erroneous.

In a plant historian, a stored data point not only has an associated value and time stamp, it also has an indication of the data quality. Storing a data point from an instrument, outside of the instrument's normal operating range, for example, will cause a specific series of quality indicators to be stored with the value. These indicators aren't simply separate columns in the database, but an inherent property of the sample. They can be retrieved, included in calculations and used to alert operations or engineering personnel to a potential anomaly.

When summarizing the values (for example, calculating an average temperature over the last hour), a historian must be able to reflect this data quality in calculation results, optionally filter out suspect data, and be able to extrapolate when data is missing or deemed invalid. If these real-world aberrations aren't handled correctly, resulting reports, business system integration, and decision making will be incorrectly skewed. Relational databases alone don't provide these capabilities.

Myth #11: Storing data in a relational database makes it integrated.

Putting two Excel spreadsheet files into the same folder doesn't make them "integrated" in any sense, even though they might both include production data. Similarly, taking information an ERP system and historian and storing both in a relational database doesn't make it "integrated". Certainly having all that data in a common technology is requisite first step, but it is only that and often it is by far the simplest.

Myth #12: Storing data in a relational database makes it easy to query.

Although support for SQL queries is one of the huge benefits of using a relational database, that doesn't necessarily mean a particular database design is easy to query via SQL—some designs are even so convoluted that they cannot effectively use manually-created queries and must, instead, rely on complex, programmatically generated queries. This can make sense from the perspective of managing the storage and from the standpoint of portability, but it largely neutralizes all the inherent advantage of using a relational database.

Myth #13: Using a relational database is cheaper than a purpose-built historian.

Before you assume you can't afford a serious historian solution, make sure you understand your real needs and explore the options—you might be very pleasantly surprised, even based only on the license costs. When you factor in the lower ownership costs and value of a solution adapted to your real needs, the purpose-built solution will likely be a lot cheaper.

Myth #14: Only large-scale continuous processes need a historian.

The original commercial historian systems began in the 1980s in oil refining, paper mills and other continuous processes since these were the only industries that could justify the cost of minicomputers required to run them. With the rapid adoption of Windows NT in the 1990s, the cost of the computers dropped significantly and opened the door for lower priced solutions developed specifically for the new platform, such as Wonderware's IndustrialSQL Server (now, "Wonderware Historian"). These new solutions quickly demonstrated their cost effectiveness outside of the traditionally DCS-oriented continuous process industries.

Conclusion

Although basing a historian solution on a relational database alone can sound like a great idea, there are many inherent challenges and limitations. However, this does not mean that commercial software has no place in an industrial environment. Today, process information is needed both outside of the plant environment and inside the corporate network. And, there is no better way to provide this interface between the plant data and the enterprise systems than a commercially accepted, standards-based technology, such as SQL.

Wonderware Historian extends a commercially available product (Microsoft SQL Server) with an open, standard query interface (SQL) adapted to plant historical data. This interface can easily be used by the IT department for reporting or integrating into the ERP systems.

Wonderware Historian offers all the capabilities discussed within this paper and more. Trusted and in use in over 75,000 installations worldwide, Wonderware Historian empowers plant operations and enterprise business users alike, delivering the right information to the right person, and leaves database management where it belongs, in the enterprise IT department and not on the plant floor.

About the author

Elliott Middleton has over 30 years of experience with industrial software, primarily with process historians, business system integration and operations intelligence applications. As a product manager, he is responsible for setting the direction for the Schneider Electric software information solutions, including the Wonderware Historian, Historian Client (aka ActiveFactory), and Wonderware Online. Working with customers, business partners, and the Schneider Electric development and support teams, Elliott works to identify market problems, define product requirements and prioritize enhancements. Prior to joining Wonderware in 2001, he held similar positions at AspenTech and IndX. He holds a Bachelor of Science degree in Computer Science from Baylor University.