The Database Decision:
Key Considerations to Keep in Mind
# Table of Contents:

<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The Database Decision: A Guide</td>
</tr>
<tr>
<td>7</td>
<td>Leverage Time-Series Data to Speed up and Simplify Analytics</td>
</tr>
<tr>
<td>10</td>
<td>Open-Source Technology Finally Living Up to its Promise</td>
</tr>
<tr>
<td>12</td>
<td>Data Lakes Receive Mixed Reception at Hadoop Summit</td>
</tr>
<tr>
<td>14</td>
<td>How I Learned to Stop Worrying and Love the Cloud</td>
</tr>
<tr>
<td>17</td>
<td>Make Your Data Perform in the Cloud</td>
</tr>
<tr>
<td>20</td>
<td>From SQL to API Frameworks on Hadoop</td>
</tr>
<tr>
<td>22</td>
<td>What NoSQL Needs Most Is SQL</td>
</tr>
<tr>
<td>25</td>
<td>Don’t Overlook Data Transfer Requirements in Hadoop</td>
</tr>
<tr>
<td>27</td>
<td>Time to Renovate: Updating the Foundation of Data Analytics</td>
</tr>
<tr>
<td>30</td>
<td>The Data Lake: Half Empty or Half Full?</td>
</tr>
<tr>
<td>32</td>
<td>Fulfiling the Promise of Big Data with HTAP</td>
</tr>
<tr>
<td>35</td>
<td>Big ETL: The Next ‘Big’ Thing</td>
</tr>
<tr>
<td>38</td>
<td>Why Open Source Storage May Not Be for You</td>
</tr>
<tr>
<td>41</td>
<td>Rethinking DBMS: The Advantages of an Agile Infrastructure</td>
</tr>
<tr>
<td>44</td>
<td>Are Shared-Nothing SQL Database Clusters Right for Your Business?</td>
</tr>
<tr>
<td>47</td>
<td>How to Innovate Using Multi-Source, Multi-Structured Data</td>
</tr>
<tr>
<td>51</td>
<td>Modernizing M2M Analytics Strategies for the Internet of Things</td>
</tr>
<tr>
<td>54</td>
<td>Is SQL-on-Hadoop Right for Your Real-Time, Data-Driven Business?</td>
</tr>
<tr>
<td>57</td>
<td>How Database Environments Lose Performance and What You Can Do About It</td>
</tr>
<tr>
<td>60</td>
<td>Fast Database MapD Emerges from MIT Student Invention</td>
</tr>
<tr>
<td>67</td>
<td>Innovative Relational Databases Create New Analytics Opportunities for SQL Programmers</td>
</tr>
<tr>
<td>70</td>
<td>Advice for Evaluating Alternatives to Your Relational Database</td>
</tr>
<tr>
<td>73</td>
<td>How an In-Memory Database Management System Leads to Business Value</td>
</tr>
<tr>
<td>75</td>
<td>How the Financial Services Industry Uses Time-Series Data</td>
</tr>
</tbody>
</table>
The Database Decision: A Guide

by Frank Hayes

As your organization collects more data — and wants to make better use of the data it’s already acquiring — you now have options that stretch far beyond traditional databases. Using business intelligence tools by pulling extracts from your data warehouse may no longer be enough; what you really want is to mine all the business data you can get. And at the point where that data is too big to analyze fast enough by conventional means, it’s time for a hard look at database alternatives.

Your choice of database technology will be dictated by your project’s parameters: how much data, how fast it arrives, the analysis you’ll do and whether you’ll use that analysis operationally or for making strategic decisions. You may want to mine business insight from retail point-of-sale transactions or a CRM (customer relationship management) program. You might be trying to speed up logistics, identify where inventory is in real time in your supply chain or track the wanderings of large numbers of customers on a website.

Which databases are appropriate for what sorts of projects? The following is a guide to help you evaluate your options.

Relational Database

What it is: Traditional relational database system, with structured data organized in tables

Strengths: Uses standard SQL query language; well supported with tools and staff skills; guarantees consistency for results of queries

Use cases/examples: Conventional online transaction processing (OLTP)

Weaknesses: Requires complete table to be stored in file or memory; requires structured data and relational schema; canonical table must be replicated in order to perform analysis on parallel servers

If you’re dealing with structured, relatively simple transaction data — say, customer orders or point-of-sale data — and you want maximum flexibility in the questions you’ll be able to answer, the place to start is the familiar relational database. The advantages: You’re probably already using an RDBMS, so your IT staff includes database analysts who can create schema and SQL queries. You just need to choose an enterprise-class database that will scale up to match your data analysis needs, such as Oracle, IBM InfoSphere Warehouse, Sybase and Teradata.
The downsides: The price for familiarity and flexibility in queries is that data is rigorously structured and is stored in a single canonical set of tables that has to be replicated if you use parallel processing to speed up performance. Every update has to be replicated to other copies, so if data changes a lot, there’s a lot of overhead, which negatively affects the response times for queries.

You can keep those tables in memory instead of on disk in order to speed up reads and writes — but at a higher cost. And as tables grow to millions of rows, response time for your analytical queries can quickly slide from near-real-time down to sometime-this-week.

**Columnar Database**

**What it is:** Relational database that stores data by column rather than by row

**Strengths:** All the strengths of a relational database, and optimized for better performance when analyzing data from a few columns at a time

**Use cases/examples:** Analyzing retail transactions by store, by state, by product, and other criteria

**Weaknesses:** Requires complete table to be stored in file or memory; requires structured data and relational schema

What if that data is highly structured but you know you’ll be doing a lot of analysis on data stored in just a few particular columns — say, analyzing retail transactions by store, by state or by product? Then a columnar database makes more sense, like kdb+, Sybase IQ, Microsoft SQL Server 2012, Teradata Columnar or Actian VectorWise.

Despite the name, a columnar database is also a true relational database. The difference is under the covers. Instead of storing each data table as one row after another in a single file, a columnar database stores columns in separate files, so they can be indexed and searched separately, even on different servers in a cluster to improve performance.

As a result, columnar databases can be much faster than row-based relational databases for analysis that concentrates on only one or a few columns, and many staff skills can transfer directly to columnar. They also inherit many relational database drawbacks, including the staff effort and system processing time required to maintain canonical tables.

**Object-Oriented Database**

**What it is:** Database that directly stores software objects (from object-oriented programming languages such as C++ and Java)

**Strengths:** Designed for querying complex data; much faster at storing and retrieving data in software objects
Use cases/examples: Storing and analyzing complex data from a customer loyalty program

Weaknesses: Some products only support a few programming languages; no standard query language; fewer tools and staff skills available

Suppose instead that your project requires analyzing data from a customer loyalty program that uses a highly complex customer model. Your developers might already know that their biggest bottleneck will involve packing and unpacking data from software objects so the data can be transferred to and from a database. That’s a good candidate for an object-oriented database such as Objectivity/DB, Versant Object Database and db4o, which store entire objects automatically.

Letting the database handle object storage by itself has the advantage of simplifying programming and thus reducing the chance for bugs, and it may improve performance. But object-oriented databases aren’t relational, so there’s none of the flexibility for ad-hoc queries that’s available with SQL. And dealing with data complexity becomes a programming problem, not a database management issue.

NoSQL Databases

What it is: Non-relational database that stores data as transaction logs rather than as tables

Strengths: Faster and more compact when dealing with very large but sparse databases; supports unstructured or irregularly structured data; supports easily distributing analysis to parallel servers

Use cases/examples: Tracking all user actions on a website (not just purchase transactions)

Weaknesses: Not mature technology; no standard query language or tools; does not guarantee consistency in query results

Suppose your project will deal with orders of data that’s not nearly so structured — say, tracking inventory logistics in real time or analyzing how customers use your website. With conventional databases, that could require huge, multidimensional tables in which most of the cells would likely be empty, while others might have to hold multiple values at once.

That’s the kind of messy situation that NoSQL databases were created for. It’s a catch-all name for a class of non-relational databases that include Hadoop HBase, Hypertable, MongoDB and MarkLogic. With NoSQL, there’s no actual data table. Instead, the database consists of the list of all data writes; a query searches that list to find out what data was most recently written to a particular cell. (It’s as if you created an enormous table, logged every write, then threw away the table and just searched the log.)

There’s no need for a giant canonical table stored on disk or in memory. Individual cells are never written to a table, so “writes” are effectively instantaneous. And copies of the log-like database can be written to many servers within a cluster, so analysis can be split up in
parallel. Compared to conventional data tables, NoSQL has a smaller footprint, performs much faster and is easier to parallelize.

But as the name suggests, NoSQL offers no relational integrity or consistency guarantees, and currently no standard query language. It’s most suitable for specific, well-defined analysis on very large volumes of data, especially when results need to be delivered in real time.

There’s also no widespread skills support for NoSQL databases, and that could be a problem. However, there are software tools that support NoSQL work such as Hadoop, which splits up the work to be performed on multiple servers in a cluster. And cloud-computing vendors are increasingly bundling those tools with their cloud server and storage offerings, so it’s not necessary to buy hardware in order to do NoSQL-based analysis.

Management Considerations to Keep in Mind

Whatever database you choose for a Big Data project, you’ll probably need to acquire new staff skills and technology. Make sure those costs fit into the context of your business project. Unfamiliar technology requires that the new technology get the proper vetting and skills investment. Your staff knows how to deal with conventional databases, but much of that knowledge won’t transfer directly. That means it’s a good idea to build up from small-scale experiments and prototypes, as well as tapping the experience of other enterprises.

And keep in mind that vendors are always working to address the limitations and add features to meet new big-data needs — but they also may change techniques that your developers will need to use.

This kind of data analysis is still in its early days, and you’re likely to make mistakes. The less costly those are, the better. But the sooner you begin identifying the best database technology for your own analysis projects, the better off you’ll be.

Frank Hayes is a veteran technology journalist and freelance writer.
Leverage Time-Series Data to Speed up and Simplify Analytics

By Mark Sykes

Time-series databases increasingly are being recognized as a powerful way to manage big data. Over the past decade, they have been widely adopted by financial services companies for their speed and performance. Today, industries with IoT applications are beginning to implement time-series databases more widely. Here at Kx Systems, an early leader in high-performance time-series database analytics, we still see a lot of misconceptions about exactly what time-series means.

Most people understand that there is a temporal nature to data. But many still look at their big data applications purely through the lens of the architecture, which means that they are missing a vital point: the unifying factor across all data is its dependency on time. The ability to capture and factor in time when ordering events can be the key to unlocking real cost efficiencies for an organization’s big data applications.

Within financial institutions, and many other types of enterprises, data falls into one of three categories — historical, recent, or real-time. Firms need to access each of these data types for different reasons. For example, customer reporting requires firms to get a handle on recent data from earlier in the day, while various surveillance activities demand a combination of recent and real-time insight. Market pricing-related decisions rely on a combination of real time and historical data analytics.

Creating a Temporal View

Whether it’s market data, corporate actions, client orders, chat logs, emails, SMS, or the P&L, each piece of data exists and changes in real time, earlier today, or further in the past. However, unless they are all linked together in a way that firms can analyze, there is no way of providing a meaningful overview of the business at any specific point in time.

One approach organizations use to achieve this overview is a flexible temporal approach, which allows firms to knit together information at precise points on a metaphorical ball of string. That means they can rewind events and see what the state of the business was at a particular moment in history, including the impact of any actions leading up to that moment.
Some firms have struggled to introduce this concept due to the limitations of their technology. Some architectures simply don't allow for time-based analysis. It's a common issue that can result from opting for a seemingly safe technology choice.

Even when firms have succeeded in layering a temporal view on their data, it is only part of a longer journey. Just storing the time at which each business event occurred is simply not enough — firms need a fundamental understanding of time. That means they have to interconnect all the information within the time dimension in a meaningful way. To look at the world in this way calls for speed so users can cut through the complexity and maintain performance.

**Square Pegs and Round Holes**

Legacy technology, again, is a hurdle. In many cases, the three different collections of data will live on three different platforms — such as a traditional database for historical data, in-memory for recent data, and event processing software for real-time data.

This type of technology stack can offer a wide variety of options. However, it is rare for constituent applications to span more than one of the three domains effectively — and even rarer for them to do so with sufficient performance.

As a result, users can make connections within their historical or in-memory databases, but not across their whole system. The unfortunate reality is that programmers might have to write very similar code two or three times in different systems and then aggregate results back together before those results can be used. This is time-consuming, prone to errors, and makes the system very difficult to maintain over the long term.

Ultimately, if the setup is not designed for time-series analysis at its heart, it becomes an expensive, complex, and slow solution — and users will find themselves trying to fit square pegs into round holes.

**A Unified Approach**

The question, then, is how to write the code only once, so users can deploy a single query and see results across all of their data sets. The answer is one unifying piece of architecture that spans all time boundaries. This is the key to putting theory into practice.

Many firms will recognize that all their data is, in some way, time-dependent. By rethinking and unifying the underlying technology, they can move beyond simply storing data, or even analyzing only part of it, to truly understanding all of their data and making correlations across the business.
The cost benefit is huge — and that’s crucial. Often, this ends up as a balance sheet issue. Time really does mean money, and firms that invest in embedding time into their technology in the right way stand to realize the greatest savings.

Mark Sykes is Global Market Strategist at Kx Systems. Mark is engaged in expanding Kx’s reach in financial services and other industries around the world. He has over 20 years’ experience in capital markets and data analytics. As Global Head of Algorithmic Solutions at Bank of America Merrill Lynch from 2010 to 2013, he managed the quant, trading, and technology teams, and introduced the FX execution platform built on kdb+. Previous roles include Director, Foreign Exchange at Citigroup, designing FX algorithmic execution strategies; and Director, Head of Algorithmic Trading at Deutsche Bank, implementing the bank’s first tick history database using kdb+. Mark is based in London.
Open-Source Technology Finally Living Up to its Promise

by Ron Bodkin

Open-source platforms have long been magnets for innovation and experimentation. From the early days of Unix through Mozilla, Linux, and the most recent Apache Foundation Hadoop distributions, open source has been the place to be for eclectic and sometimes groundbreaking development.

Despite this rich environment, even the most die-hard open source enthusiast has to admit that enterprise IT has been guided and largely shaped by innovations coming from the vendor community. Starting with the mainframe and the personal computer and extending to modern virtual and software-defined architectures — the major gains in IT development have come from the labs of companies such as IBM, Apple, Microsoft, VMware, and more.

The previously mentioned vendors’ platforms have helped companies streamline footprints, control costs, and improve data productivity. But in case you have missed it, there is a sea change underway, in which data platforms and services are now created, deployed, and consumed via open source. And the many open-source communities in existence today already are making major contributions to the worldwide data management ecosystem.

This is a “community-driven” technology revolution because for the first time, users of data platforms are setting the terms regarding the future of software development. And it just happens to coincide with the age of big data, the Internet of Things, collaborative workflows, and other initiatives that are forcing enterprises to think out of the box when it comes to redesigning and re-imagining their data stack.

Examples of this dynamic at work include Yahoo’s Hadoop cluster, currently the largest in the world at an estimated 450 petabytes. In addition, there is the Facebook Hive project that translated SQL-like queries into MapReduce so that data analysts can utilize familiar commands to manage and scale out Hadoop workloads and its successor, Facebook’s Presto system, for interactive SQL on Hadoop. And then there is LinkedIn's Kafka, which enabled a distributed publish-subscribe messaging system, supporting functions such as event-flagging and data partitioning for real-time data feeds.
Open-Source Technology Finally Living Up to its Promise

Once projects such as Hadoop, Hive, Presto, and Kafka were open sourced, it was the strength of their respective communities that both led to and continues to fuel innovation. For example, when the team at Yahoo opened up Hadoop as an independent, open-source project, it kicked off an era of big data management and allowed the broader open-source community to begin producing myriad solutions that were vital to enterprise productivity. Big data is now a gold rush as organizations of all kinds seek to mine the valuable nuggets that lie within their data stores.

And there is every reason to think this trend will only accelerate as the data ecosystem evolves along increasingly virtual, scale-out lines. Unlike proprietary vendors, hyperscale entities like Facebook and Google are not in the business of creating and selling software. So the solutions they develop and then release freely provide the community with a heightened level of knowledge, which in turn benefits the developers in the form of still further development and increased patronage of their cloud or social/collaborative products and services. It’s a symbiotic relationship that ultimately benefits the entire IT industry.

Vendor-driven innovation provides a steady and somewhat straightforward development path, while open-source initiatives tend to be a bit more chaotic, with new products coming at a rapid pace.

This is why knowledgeable systems consultants are crucial for the development of enterprise-class open-source architecture. No single platform or development community has all the pieces to implement end-to-end data architecture, and it is very unlikely that all the right components will be implemented by chance, so a solid integration partner is a critical asset as open platforms evolve.

This is an exciting time for open source. With data architectures rising and falling at a rapid pace on newly abstracted, scale-out infrastructure, the need for broad interoperability and cross-vendor compatibility is paramount. Community-driven initiatives have shown they can step up to the plate for big data and other critical needs, but ultimately it is the enterprise that needs to reach out and take advantage of the wealth of new tools emerging from these vibrant and energetic development efforts.

Ron Bodkin is the founder and CEO of Think Big Analytics. Ron founded Think Big to help companies realize measurable value from big data. Think Big is the leading provider of independent consulting and integration services specifically focused on big data solutions. Our expertise spans all facets of data science and data engineering and helps our customers to drive maximum value from their big data initiatives.
SAN JOSE, Calif.—The advantages of using data lakes as a way to corral big data by putting it all in one place, or “lake,” are well documented.

Ah, if only it were that easy. While it can be handy for data scientists, analysts, and others to pull disparate data sources from a data lake, finding what you need and reconciling file compatibility issues can make it tricky.

At the recent Hadoop Summit here, data lakes were a controversial topic. Walter Maguire, Chief Field Technologist at HP’s Big Data Business Unit, said the data lake is a relatively young concept and noted that it has had its share of criticism, with some saying that, in practical terms, it would be more accurate to call it a “data barn” or “data swamp.” As part of a broader presentation on the topic, he talked up HP’s own Haven for Hadoop solution as a way to make murky data lakes “clear” for data scientists and others to get at the data they need.

Ron Bodkin, Founder and President of Teradata company Think Big, focused his keynote on data lakes, noting the pros and cons as well as customer examples of successful implementations.

“To us, a data lake is a place where you can put raw data or you can process it and refine it and provision it for use downstream,” said Bodkin. “That ability to work with a variety of data is really critical. We see people doing it fundamentally so they can manage all their data and can take advantage of innovation in the Hadoop community and beyond.”

He noted that new tools such as Presto (SQL Query Engine), Spark (for running in-memory databases at greatly higher speeds), and Storm (a distributed real-time computation system for processing large volumes of data) make working with Hadoop faster and more effective. He said that Teradata has a team of 16 people working on enterprise Presto development.

Bodkin offered the example of a high-tech manufacturer that keeps data about its manufacturing processes around the world in a data lake. Used effectively, the data lake lets the company trace its parts and improve the yield, leading to faster time to market and reduced waste. “Putting a Hadoop data lake into their manufacturing system has been a major accomplishment,” he said.

But not all data lake implementations are so successful. Acknowledging the “data swamp” issue, Bodkin said the first wave data lake deployments typically have been for one use case, some specific thing that a company wanted to accomplish.
“What happened though is these systems have grown as companies are putting in thousands of tables, dozens of sources, and many more users and applications, and it isn’t scaling,” he said. “We have seen this dozens of times with companies that started working with Hadoop two years ago and now it’s a nightmare of a hundred jobs they have to monitor, run, and worry about.”

**Building a Mature Data Lake**

From Teradata’s enterprise customer engagements, Bodkin said, he’s come to some conclusions about what constitutes a “mature data lake” that can be a trusted part of the enterprise infrastructure.

“Underlying it, of course, is the ability to have security, to be able to secure data,” he said. “You need to have regulatory compliance and the ability to store data in an efficient way that’s not used as often but when it is, in an active way.”

He also pointed to the need for a metadata repository so you can index and find what you need easily, and noted that efficient file transfers into the cluster are important because a lot of them can get cut.

“Whatever the means of ingestion, you need to govern it,” he said. “You need to be able to trace as the data’s going into the data lake and what versions.”

The governed data is that trusted version that you can use downstream to, for example, create a data lab where data scientists can play with new data that comes in and combines it with other data in the repository using some of the new tools that have been recently released.

**Getting Started**

Wrapping up, Bodkin noted that many companies are still trying to get their footing on how a data lake can help them. Teradata’s approach is to offer a road map designed to help companies plan out what they want to do, including showing how a data lake can be built in a scalable way that can be governed.

“There are best practices and patterns you have access to so you don’t incrementally work your way into a data swamp,” said Bodkin. “You can start off on a clean footing.”

---

Veteran technology reporter David Needle is based in the Silicon Valley, where he covers mobile, enterprise, and consumer topics.
How I Learned to Stop Worrying and Love the Cloud

by Gerardo Dada

With the allure of cost savings, greater flexibility, and more agility, many organizations are eyeing the cloud as an alternative to deploying new applications, including those with high database performance requirements. Just a few years ago this may have struck fear in the heart of the average database administrator. But today it is a much more viable option. Consider that technology research firm TechNavio predicts a 62 percent annual aggregate growth rate for the global cloud-based database market through 2018.

However, this doesn't mean there aren't still new complexities and challenges associated with cloud-based databases that must be accounted for. Thus, when it comes time to actually migrate their applications and the databases that support them, many organizations are still somewhat unsure of where to start. Perhaps you fall into this category. To help, here are some key considerations you should remember when thinking about migrating databases to the cloud.

Let Go of Your Fear

I still hear too many people say the cloud is too slow for their database needs. A couple of years back, shared storage systems in the cloud delivered very unpredictable performance, which sometimes slowed to a crawl.

However, the architecture of today’s cloud storage systems, often based on solid state drives (SSD), storage-optimized instances and guaranteed performance options, offer up to 48,000 Input/Output Operations Per Second (IOPS), more than enough performance to meet the requirements of most organizations. In fact, in some cases it’s even easier to get higher performance from the cloud, as pre-configured options are sometimes faster than many on-premises systems.

Know What You Need

Perhaps the best way to arrive at the point that you can let go of these performance fears is to truly understand what your cloud database performance requirements are. That knowledge, combined with insight into how the available cloud options might be able not
only to meet those needs but exceed them will go a long way. But how do you really know what you need? The best way to figure this out is to look at what the database resource requirements of your applications currently are, including the following:

- CPU, storage, memory, latency, and storage throughput (IOPS can be deceiving)
- Detailed performance information from wait-time analysis
- Planned storage growth and backup requirements
- Expected resource fluctuation based on peak application usage or batch processes
- Security requirements, including encryption, access, etc.
- Availability and resiliency requirements, such as backups, geographic dispersion, and mirroring
- Data connection dependencies, especially from systems not in the cloud

One of the advantages of the cloud is the ability to dynamically scale resources up and down. So, rather than being the source of performance uncertainty concerns, it actually can give you peace of mind that the right amount of resources can be allocated to your applications to ensure adequate performance. The risk here, however, is that without clear knowledge of where the bottlenecks are from proper wait-time analysis, it is easy to overprovision and spend too much money on cloud resources, sometimes even without the expected performance benefit.

Failing to Plan is Planning to Fail

The old proverb still rings true, especially when it comes to picking your cloud deployment model. For example, Database as a Service (DBaaS) provides simplicity in deployment, automation, and a managed service. Leveraging Infrastructure as a Service (IaaS) is an alternative for running database instances on cloud servers that provides more control and that looks and feels like a traditional on-premises deployment. There are also various storage options, including block storage, SSD drives, guaranteed IOPS, dedicated connections, and database-optimized instances. As the cloud is mostly a shared environment, it is also important to understand and test for performance consistency and variability, not just peak theoretical performance.

Try Before You Buy

With new options and capabilities available for you in minutes and at low cost, take advantage of the cloud and experiment! Just like you would take a car for a test drive before buying it, do the same with the cloud. In just an hour, you can set up a proof-of-concept database for very minimal cost. If you want, it only takes a little more time and money to create a sandboxed copy of an actual database from your organization to test
out specific functions and deployment options to see how your specific database will operate in the cloud.

**Don’t Be Afraid to Ask Questions**

As much as you might like one, there is no plan that can account for every possible cloud migration use case or issue that might come up. Most cloud service providers offer migration and architecture guidance. Don’t be afraid to ask for help. It’s also a good idea to run a mirror of your on-premises system in the cloud for some time before fully transitioning.

These planning and migration best practices might not make you an expert (just yet), but taking them into account should help you get started. Not taking advantage of the cloud could be a missed opportunity. Experience in the cloud is good for every professional, and soon will be necessary for your career.

Gerardo Dada is Vice President of Product Marketing and Strategy for SolarWinds' database, applications cloud businesses globally, including SolarWinds Database Performance Analyzer. Gerardo is a technologist who has been at the center of the Web, mobile, social, and cloud revolutions at companies like Rackspace, Microsoft, Motorola, Vignette, and Bazaarvoice. He has been involved with database technologies from dBase and Btrieve to SQL Server, NoSQL and DBaaS in the cloud.
In a previous column, I discussed a few key considerations you should remember when thinking about migrating databases to the cloud. Now, it’s time to talk about managing them once they are in the cloud.

What’s that you say? “But I thought not having to manage my database is the whole reason for moving it to the cloud!”

True, in addition to a number of other benefits previously discussed, hosting your databases in the cloud does mean much of the demand and pressure associated with the day-to-day database infrastructure management upon you is relieved. However, that doesn’t mean you are completely off the hook.

With that in mind, here are a few things to consider:

- **You are ultimately responsible for performance.** While giving the cloud provider control of your database infrastructure may take some of the burden off of you for administrative and configuration tasks, you are still responsible for overall database performance. You need to pay attention to resource contention, bottlenecks, query tuning, indexes, execution plans, etc. You may need new performance analysis tools, as many do not work properly on the cloud, but some work great.

- **You are ultimately responsible for security.** There are both real and perceived security risks associated with databases in the cloud, and you can’t expect the cloud service provider to take care of security for you. You need to think of security as a shared model in which you are responsible for access and governance, encryption, security monitoring, backups, and disaster recovery.

So, what exactly is your role in managing your cloud-based database? Here are some ideas:

**1. Understand and manage data transfers and latency.** You will need to determine where your data actually is — region and data center — as well as plans for using multiple availability zones, active and passive disaster recovery, or high-availability sites. It will be important to take into account data transfer and latency for backups and to have all your databases in sync, especially if your application needs to integrate with another one that is not in the same cloud deployment. Some cloud providers allow you to ship them hard drives, some have dedicated high-speed connections, and some will provide architectural guidance to help you through the decision process.
2. **Know your service provider’s cloud and stay on top of changes.** It’s imperative that you take the extra time to understand your service provider as cloud service providers tend to be evolving quickly. You should stay on top of new services and capabilities, understand your provider’s SLAs, review its recommended architecture, and be very aware of scheduled maintenance that may impact you. The cloud is a partnership, and you and the service provider need to be in sync.

3. **Be aware of the cost structure.** It’s easy to get started in the cloud, but that simplicity can quickly lead to an expensive habit. You should seek to understand all the elements that make up the cost of running your database in the cloud — such as instance class, running time, primary and backup storage, I/O requests per month, and data transfer — and their growth expectations over time. Doing so can help you avoid overprovisioning and utilize cloud resources more efficiently.

4. **Avoid putting all your eggs in one basket.** Think through, plan, and manage the requirements you need for backup and recovery to ensure you don’t lose important data in the event of a vendor failure or outage. Do this by keeping a copy of your data with a different vendor that is in a different location, so it’s safe and can easily be recovered in case of a catastrophe.

5. **Stay on top of security.** What are your organization’s security requirements and what regulations do you need to follow for compliance? Encryption is only the tip of the iceberg. There are considerations like which keys will you use, who has access to them, what algorithm will be used to do the encryption, how will data be protected at rest, in transit, and backups. Also, who will monitor database access for malicious or unauthorized access? Remember, most security threats come from inside. Last, plan for the worst and have a documented course of action ready in case of a security breach or data loss.

6. **Monitor and optimize your cloud environment.** If it is important to monitor and optimize on-premises deployments, it’s even more important in the cloud, given its dynamic nature. Database performance optimization tools can do complex wait-time analysis and resource correlation to speed database operations significantly and lower costs. These tools also can issue alerts based on baseline performance to identify issues before they become big problems. Database administrators, developers, and operations teams can benefit from a shared view of the performance of production systems that allows them to see the impact of their code and pinpoint the root cause of whatever could be slowing down the database, whether it’s queries, storage systems, blockers, etc.

Running your database in the cloud can lead to greater flexibility and agility, and it also can mean less of your time being spent in the administration of the database. However, don’t fall into the trap of thinking that it means you can check out of your database.
management responsibilities. Really, you should look at it as an opportunity to spend more time managing the next level — honing and optimizing database and application performance.

Gerardo Dada is Vice President of Product Marketing and Strategy for SolarWinds’ database, applications and cloud businesses globally, including SolarWinds Database Performance Analyzer and SolarWinds Cloud. Gerardo is a technologist who has been at the center of the Web, mobile, social, and cloud revolutions at companies like Rackspace, Microsoft, Motorola, Vignette, and Bazaarvoice. He has been involved with database technologies from dBase and Btrieve to SQL Server, NoSQL, and DBaaS in the cloud.
From SQL to API Frameworks on Hadoop

by Supreet Oberoi

As part of the Hadoop community, we regularly gather and compare product announcements to see how they score on the cliché scale. Over the years, none have scored as commendably for the prize as SQL-on-Hadoop technologies, which offer to “democratize” Hadoop for the thousands of analysts who have never developed software applications. SQL-on-Hadoop technologies could be either proprietary implementations or open-source initiatives, which include Hive.

These bets, taken by technology vendors and enterprises, relied on the implied belief that most of the computation directives required to build data applications on the Hadoop stack could be represented through SQL — that a recommendation, a fraud detection, and a gene-sequencing algorithm could be mostly written with restrictive primitives such “SELECT” and “WHERE” — to name a few.

Instead, such enterprises using SQL-like languages as their core development platform found that while SQL is a great language to represent queries on a relational data set, it is not ideal for developing real-world, sophisticated data products — just as it was not the correct language for writing web applications.

After experiencing issues with SQL-on-Hadoop technologies, product leaders are finally realizing that Hadoop is not intended as a database, but as a data application framework — and that’s why SQL is not at its most efficient on Hadoop. So this begs the question: What is a suitable alternative? The answer: API frameworks.

API-based approaches may have spent time in the wilderness while SQL-only approaches caught all the attention, but now they are back. Although they may not always have been the shining stars of the big data industry, they consistently have met expectations as a development framework.

Here are four reasons why API frameworks have been used successfully for developing data applications, while SQL-based technologies are proving too lackluster:

**Degrees of freedom.** Unlike programming languages, SQL-based languages lack the degrees of freedom to develop sophisticated data products. This makes sense because SQL was designed to analyze correlated and tabular data sets. However, modern data products on the Hadoop stack rely on manipulating non-relational data with sophisticated
operations. Imagine doing matrix or vector math or traversing a B-tree without having the rich data structures that a language such as Java or Scala provides.

**Test-driven development.** When an engineer is asked to write software application code, he or she has to validate the functional correctness by writing unit and regression tests. However, this important software engineering principle is skipped in SQL-based application platforms because they are missing in the SQL-based ecosystem. For example, identifying exceptions with stack trace, writing unit test code before implementing the functionality, and having checks for invariants and preconditions are all best practices that have proven to significantly improve product quality and reduce development time. Although these qualities are missing in Hive and other frameworks, they can be employed easily when programming in Java or Scala.

**Code re-use.** Yes, SQL-based approaches support the concept of code re-use with UDFs. Within a Hive application, people can neatly organize all their data-cleansing logic in reusable UDFs — but only in Java. In other words, to support code re-use, SQL-based frameworks rely on extensions with Java, and this becomes brittle and unmanageable. There is no convenient method to step through the application in a debugger when one part of the code is in Hive and another part is implemented in Java through a UDF. Many enterprises are now discovering that it is better to stay within a programmatic framework, such as Java or Scala, that supports encapsulation and object-oriented manner for promoting re-use of code to support these software engineering best practices.

**Operational readiness.** Operational readiness means the ability to monitor applications in production — on which businesses depend — and SLA metrics that provide insights into how the application performance can be improved when it isn’t reaching performance goals.

To improve the performance of an application, the first task is to instrument the code to identify areas within an application that need improvement. Instrumentation can mean measuring total execution time on the cluster, the CPU, and the I/O usage. Java or Scala developers can easily provide such visibility and insights into their application by reusing existing tools for application instrumentation, but it is not equally easy for Hive developers. The operational visibility provided by the Hadoop Job Tracker dashboard is at the level of the compute fabric (mappers and reducers) and not as much at the application level. It is easier for Java developers to identify bottlenecks in their applications than for Hive developers.

While it may not be the most efficient tool on Hadoop, all is not lost with SQL. It still plays a large part in creating sophisticated data products because a major portion of their development requires data preparation and normalization.

Supreet Oberoi is the vice president of field engineering at Concurrent, Inc. Prior to that, he was director of big data application infrastructure for American Express, where he led the development of use cases for fraud, operational risk, marketing and privacy on big data platforms. He holds multiple patents in data engineering and has held leadership positions at Real-Time Innovations, Oracle, and Microsoft.
What NoSQL Needs Most Is SQL

by Timothy Stephan

I don’t think I am going out on a limb by saying that NoSQL is very powerful and flexible — and very useful — technology. By powerful, I mean that it can easily handle terabytes of data, can scale to billions of users, and can perform a million ops per second. By flexible, I mean that, unlike relational databases, NoSQL seamlessly handles semi-structured data like JSON, which is quickly becoming the standard data format for web, mobile, and IoT applications. And, NoSQL performs a real mission-critical service: It is an operational database that directly supports some of the largest applications in existence.

So given that, my question is, “While NoSQL use is growing very rapidly, why isn’t it everywhere yet?” Why isn’t NoSQL already the standard database for web, mobile, and IoT applications? Why are people still force-fitting JSON into relational databases?

The simple answer is because NoSQL lacks a comprehensive and powerful query language. And I get it because, really, how useful is big data if you can’t effectively query it? And how powerful and dynamic can your applications on NoSQL be if they can’t easily access and utilize the data?

NoSQL Needs an Effective Query Capability

The lack of comprehensive query for NoSQL databases is a main reason why organizations are still force-fitting JSON data into relational models and watching their applications creak and groan under the strain of skyrocketing amounts of data and numbers of users. Developers are wedging JSON data into tables at great expense and complexity so that they can retain the query capabilities of SQL/RDBMS.

If we could start from scratch, how would we design a query solution for NoSQL? What would it need to do?

I will admit that this is a long list, but that’s because we need to make sure we get this right. There already have been attempts at creating a query language for NoSQL, and all have fallen short. Either they miss core functionality that renders them ineffective or they are simply an API that can require hundreds of lines of Python to perform a simple lookup.
A query language for NoSQL must enable you to do the following:

**Query data where it lies.** There should be no requirement to manipulate data in order to query it. Your query language should work with the data model you create, not the other way around.

**Understand the relationships between data.** Without commands such as JOIN, you essentially would be forced to combine all your data into a single JSON document, losing valuable insight into the relationships between data.

**Both reference and embed data.** You should be able to choose the structure that is more applicable to your project, and not be forced into making and then maintaining copies of data in order to query documents.

**Create new queries without modifying the underlying data model.** No one wants to have to anticipate every query required prior to designing the data model. And no one wants to go back and alter the data model when query requirements evolve.

**Support secondary indexing.** Again, the query language must be flexible. You should be able to query and manipulate data where it lies, without the requirement to create and maintain multiple copies in multiple formats.

**Avoid the impedance mismatch** between the data format and the database structure that occurs when storing JSON in a relational table. Removing the complex translation layer would streamline application development. By extension, you need to process query results as JSON so that your application can consume the information directly.

And, perhaps most importantly, the query language for NoSQL must be easy for developers to use.

Developers want to focus on their application and would prefer everything else to just go away. They don’t want to learn a whole new language. They won’t adopt a whole new tool. Developers want a query language that is easy and familiar. And while the query language for NoSQL must provide value, it also must be simple, recognizable, and accessible via the tools and frameworks that developers already use. You know, kind of like SQL.

**Why Not Just Use SQL for NoSQL?**

Let’s explore that idea. Why might SQL work for NoSQL?

- SQL is powerful and yet simple — expressive and easy to read.
- SQL is flexible — able to fully express the relationships between data.
- SQL is proven — 40 years of history supporting the largest implementations of relational databases.
- SQL is known — millions of developers already use it either directly or through their development framework of choice.
What NoSQL Needs Most Is SQL

- SQL has a whole ecosystem of tools and frameworks built around it — data is accessible via standard ODBC/JDBC drivers to enable seamless transfer and plug in to standard reporting and BI solutions.

The Ideal Solution for NoSQL Query Is SQL Itself

NoSQL is both powerful and flexible, but in order for NoSQL to become ubiquitous and used as the standard operational database for web, mobile, and IoT applications, it needs a powerful, flexible query language. And that query language must be SQL. It can’t be a token subset of SQL, it must enable the full power of SQL. And while there will be some required modifications to support the JSON data format, they must be minimal to enable adoption and reduce the learning curve.

Adding SQL to NoSQL is not the only requirement for NoSQL to become ubiquitous, but if we are able to marry the flexibility of JSON and the scalability and performance of NoSQL with the full power and familiarity of SQL, it will be a big step forward. As funny as it may sound, what NoSQL needs most right now is actually SQL itself.

Timothy Stephan leads the product marketing team at Couchbase. He previously held senior product positions at VMware, Oracle, Siebel Systems, and several successful startups.
As organizations with new big data analytics initiatives look to utilize Hadoop, one critical step is often forgotten. Unless your data is being captured in the same place where it will be analyzed, companies need to move large volumes of unstructured data to readily accessible shared storage, like HDFS (Hadoop distributed file system) or Amazon S3, before analysis. Traditional methods for moving data often fail under the weight of today’s large data volumes and distributed networks. And you can’t do anything with Hadoop unless the data is accessible locally.

The majority of unstructured data is not generated where it can be analyzed. Typically, the data are generated at distributed points and must be transferred to a central location for analysis. For example, to perform efficient image analysis of surveillance video using a Hadoop cluster, video captured at remote camera locations first must be transferred to shared storage accessible to the cluster. Given that each minute of HD video recorded at 50Mbps represents almost half a gigabyte of data, transferring video or any other type of large unstructured data in a timely manner is not a trivial task. Even with high bandwidth, traditional methods are very inefficient for transferring large volumes of data over long distances.

Traditional methods for moving files over the Internet, like FTP and HTTP, are still the default means of transferring data to shared storage, but are very inefficient when it comes to using high bandwidth over long distances. Designed in the ’70s, FTP was created to solve problems during the early days of the Internet, when files and file collections were relatively small and bandwidth was relatively limited. These protocols cannot capitalize on the higher bandwidths of today’s networks and remain slow and unreliable for large files and unstructured data sets, no matter how big the pipe is.

So why haven’t tools that can quickly handle large file transfers been developed? They have. But initially, the most advanced transfer acceleration technologies were being used for far different purposes. Originally created by companies such as Signiant to transfer the huge movie and television files of organizations like Disney and the BBC, next-generation file-acceleration technologies mostly have been confined to the vast IT infrastructures of media enterprises.

It wasn’t until the cloud revolution and the development of SaaS/PaaS (software-as-a-service/platform-as-a-service) solutions that this technology became relevant and
accessible to everyone else. The public cloud offers a virtually unlimited, elastic supply of compute power, networking and storage, giving companies ready access to big data analysis capabilities including quickly and easily creating Hadoop clusters on the fly. Combined with the SaaS/PaaS model’s pay-for-use pricing, software becomes a utility service, where you pay for what you use without having to own and manage its underlying infrastructure.

There are several reasons why businesses might need to move collected data very quickly. The faster that businesses can move data for analysis, the faster they can free up the storage required to cache it at the collection points, cutting down on storage costs and management. And, if a business is after real-time analytics in order to gain a competitive advantage or if the business offers analytics services, the faster it gets results, the greater the return on the business’ Hadoop investment will be.

Moving data to the cloud, whether it’s for analysis or storage, is sure to be a part of almost every company’s future, but especially those companies that already are data driven. When analyzing big data collected in remote locations with Hadoop, don’t overlook the benefits of fast file transfer.

Ian Hamilton is Chief Technology Officer at Signiant. He has been an innovator and entrepreneur in Internetworking infrastructure and applications for more than 20 years. As a founding member of Signiant, he has led the development of innovative software solutions to address the challenges of fast, secure content distribution over the public Internet and private intranets for many of the media and entertainment industries’ largest companies. Prior to Signiant, Ian was Chairman and Vice President of Product Development at ISOTRO Network Management. He was responsible for launching ISOTRO’s software business unit and created the NetID product suite before the company was successfully acquired by Bay Networks. Ian held senior management positions at Bay Networks and subsequently Nortel Networks, from which Signiant emerged. Previously, Ian was a Member of Scientific Staff at Bell Northern Research performing applied research and development in the areas of Internetworking and security.
Time to Renovate: Updating the Foundation of Data Analytics

by Jay Lillie

Data is getting a fresh look in the Federal space. There have been several overhauls specifically designed to make data something that drives better insights and improved decision making. One recent initiative was the creation of the data.gov site. The goal was to make economic, healthcare, environmental, and other government information available on a single website, allowing the public to access raw data and use it in innovative ways. In just one year, Data.gov published datasets have grown from 47 to more than 385,000 — and we are only seeing the tip of the iceberg. With an enormous national treasure trove of data to make sense of, it’s no wonder that President Obama appointed Dr. DJ Patil as the first U.S. Chief Data Scientist in February to maximize the nation’s return on its investment in data.

This development would have been hard to predict even a decade ago, and reflects a need to make all this data not just available, but usable. The way data is consumed and synthesized within the government space needs to change.

The traditional method for getting clean, curated, ready-to-analyze data has always resided with IT. Technical experts with deep skillsets in SQL and scripting would collect raw data, put it into an extract, transform, and load (ETL) process, dump it into a data warehouse and extract it into files for the analysts to use. However, to make it usable, the analysts first had to understand and commit to how they planned to use the data, and convey what questions they hoped to ask to make it usable. The data then would be delivered to the analysts, who would look at it and immediately realize that they needed more or different data, and the tedious, time-intensive cycle would begin again.

Recently, many government agencies have embraced Hadoop data lakes as a means for collecting and storing data. This meant that IT professionals could provide unlimited data storage without the rigid structure of previous databases or data warehouses. More importantly, it meant that users no longer would be constrained in terms of the number of questions they could ask based on the star-schema/snowflake-schema of their dimension and fact tables.
Analysts, too, have seen some significant results, but only in part of their work-related responsibilities. In recent years, analytic freedom has emerged for non-technical users in the form of a new visualization and business intelligence tools from Tableau and Qlik. But despite all the business knowledge that has been achieved by using these platforms, an archaic, unstable foundation has remained.

While non-technical users relied on IT to use ETL processes, there was a well understood but rarely addressed reliance by analysts to use spreadsheets like Microsoft Excel to prepare data. After all, analysts understood the data best and could move faster with this approach. However, the cost to government agencies was countless hours of effort and the introduction of risk involving data leakage on lost laptops, unmanaged access to sprawling “spreadmarts,” and indecision as debates arose as to the accuracy of the spreadsheet-transformed data.

With the adoption of Hadoop ecosystems, the responsibility has shifted dramatically toward the analyst. These non-technical users were expected to gather the right data, prepare it, and maintain traceability throughout the process. As the demand for analytical answers grew due to Tableau and Qlik’s growing popularity, the less-than-ideal data preparation platforms of the past started to crack and crumble.

Excel started to choke on the size and sophistication of the new data scale. IT staff were often unavailable for ad-hoc queries, and the available data science tools often required specialized skills that were not easy to come by — not to mention difficult for non-technical users to adopt. There was a good chance that, no matter the tool, analysts were spending more time doing the tedious data preparation work rather than the actual analytics or decision making for which they were hired.

So while analytics, storage, and retention have come of age with new technologies, the real question is why haven’t the foundations? What about the actual work that goes into taking a collection of things and ideas and turning them into a structure that “fits” and means something? Perhaps even something actionable?

A recent article in The New York Times captured the struggle of FBI analysts as part of a 9/11 Commission report. The report concluded that the FBI “needed to improve its ability to collect information from people and to efficiently analyze it,” along with the formation of a dedicated branch meant to expand the use of intelligence across all investigations.

In this example, analysts may know what they need to do by correlating social media data with a suspicious-persons dataset, but how do they accomplish it? Where are the overlap points? How can duplicate entries be removed? What happens if geographic data are needed?

This is where we are starting to see a seismic shift. It seems obvious that there needs to be a change in how data is consumed and synthesized within the government space. The creation of data.gov and the formalization of a Chief Data Scientist are just the beginning.
After all, much of what they must confront deals with addressing the foundation of data and its usability, veracity, and relevance.

Robust analytic technology and big data Hadoop lakes are driving the need for a modern data preparation solution. This solution must provide the dynamic governance and controls necessary to maintain data integrity within a collaborative platform. It also needs to give a range of analysts, scientists, researchers, and agents the ability to work together in preparing data. In doing so, it will form a foundation that creates confidence in the insights and ensure that the true value of the government’s data initiatives is unlocked.

Jay Lillie is Director of Federal Solutions at Paxata, providers of the first purpose-built Adaptive Data Preparation solution. For more information, visit [http://www.paxata.com/] (http://www.paxata.com/).
Conference season is in full swing in the world of data management and business intelligence, and it’s clear that when it comes to the infrastructure needed to support modern analytics, we are in a major transition. To put things in paleontology terms, with the emergence of Hadoop and its impact on traditional data warehousing, it’s as if we’ve gone from the Mesozoic to the Cenozoic Era and people who have worked in the industry for some time are struggling with the aftermath of the tectonic shift.

A much-debated topic is the so-called data lake. The concept of an easily accessible raw data repository running on Hadoop is also called a data hub or data refinery, although critics call it nothing more than a data fallacy or, even worse, a data swamp. But where you stand (or swim) depends upon where you sit (or dive in). Here’s what I’ve seen and heard in the past few months.

**Hadoop love is in the air.** In February, I attended Strata + Hadoop World in Silicon Valley. A sold-out event, the exhibit hall was buzzing with developers, data scientists, and IT professionals. Newly minted companies along with legacy vendors trying to get their mojo back were giving out swag like it was 1999. The event featured more than 50 sponsors, over 250 sessions on topics ranging from the Hadoop basics to machine learning and real-time streaming with Spark, and even a message from President Obama in the keynote.

One of the recurring themes at the conference was the potential of the data lake as the new, more flexible strategy to deliver on the analytical and economic benefits of big data. As organizations move from departmental to production Hadoop deployments, there was genuine excitement about the data lake as a way to extend the traditional data warehouse and, in some cases, replace it altogether.

**The traditionalists resist change.** The following week, I attended The Data Warehouse Institute (TDWI) conference in Las Vegas, and the contrast was stark. Admittedly, TDWI is a pragmatic, hands-on type of event, but the vibe was a bit of a big data buzz kill.

What struck me was the general antipathy toward the developer-centric Hadoop crowd. The concept of a data lake was the object of great skepticism and even frustration with many people I spoke with — it was being cast as an alternative to traditional data warehousing methodologies. IT pros from mid-sized insurance companies were quietly discussing
vintage data warehouse deployments. An analyst I met groused, “Hadoopies think they own the world. They’ll find out soon enough how hard this stuff is.”

And that about sums it up: New School big data Kool-Aid drinkers think Hadoop is the ultimate data management technology, while the Old Guard points to the market dominance of legacy solutions and the data governance, stewardship, and security lessons learned from past decades. But Hadoop isn’t just about replacing data warehouse technologies. Hadoop brings value by extending and working alongside those traditional systems, bringing flexibility and cost savings, along with greater business visibility and insight.

**Making the Case for Hadoop**

It’s wise to heed the warnings of the pragmatists and not throw the baby out with the bath (lake) water. As one industry analyst said to me recently, “People who did data warehousing badly will do things badly again.” Fair enough. Keep in mind that, just like a data warehouse, a data lake strategy is a lot more than just the technology choices. But is the data lake half full or half empty? Will Hadoop realize its potential, or is it more hype than reality?

I believe that, as the market moves from the early adopter techies and visionaries to the pragmatists and skeptics, Hadoopies will learn from the mistakes of their predecessors and something better — that is, more flexible, accessible, and economical — will emerge. Traditional data warehousing and data management methodologies are being re-imagined today.

Every enterprise IT organization should consider the strengths, weaknesses, opportunities, and threats of the data lake. Hadoop will expand analytic and storage capabilities at lower costs, bringing big data to Main Street. There are still issues around security and governance, no doubt. But in the short term, Hadoop is making a nice play for data collection and staging. Hadoop is not a panacea, but the promise of forward-looking, real-time analytics and the potential to ask — and answer — bigger questions is too enticing to ignore.

Darren Cunningham is vice president of marketing at SnapLogic.
Fulfilling the Promise of Big Data with HTAP

by Eric Frenkiel

It’s clear that the promise of big data still outweighs the reality. Until now, many businesses have found an effective way to store their data, but have limited means to act on it. Much of the problem lies with businesses’ using separate systems for data storage and data processing. To close the gap, database technology focused on realizing instant value from growing and changing data sources has manifested itself in hybrid transactional and analytical processing (HTAP) — that is, performing analysis on data directly in an operational data store.

To enable HTAP, the following features must be in place:

**In-Memory Computing.** Storing data in-memory is especially valuable for running concurrent workloads, whether transactional or analytical, because it eliminates the inevitable bottleneck caused by disk contention. In-memory computing is necessary for HTAP because no purely disk-based system is able to provide the required input/output (I/O) with any reasonable amount of hardware.

**Code Generation and Compiled Query Execution Plans.** With no disk I/O, queries execute so quickly that dynamic SQL interpretation actually becomes a performance bottleneck. This problem can be addressed by taking SQL statements and generating a compiled query execution plan. This approach is much faster than dynamically interpreting SQL thanks to the inherent performance advantage of executing compiled versus interpreted code, as well as enabling more sophisticated optimizations.

Some organizations turn to using a caching layer on top of their relational database management systems, but this approach runs into problems when data is frequently updated. By storing compiled query plans rather than caching one-time query results, every time a query executes, the user has the most up-to-date version of data. The compiled query plan strategy provides an advantage by executing a query on the in-memory database directly, rather than fetching cached results, which is important as data is frequently updated.

**Fault Tolerance and ACID Compliance.** Operational data stores cannot lose data, making fault tolerance and Atomicity, Consistency, Isolation, Durability (ACID) compliance prerequisites for any HTAP system. Some important enterprise readiness features to consider include storage of a redundant copy of data with automatic failover for high
availability, persistence to disk for in-memory databases, and cross-data center replication. While these features are not tied to HTAP performance per se, they are absolutely necessary for HTAP to be implemented in an enterprise environment.

**Benefits of HTAP**

HTAP performs transactional and analytical operations in a single database of record, often performing time-sensitive analysis of streaming data. For any data-centric company, an HTAP-capable database can become the core of its data processing infrastructure, handling day-to-day operational workloads with ease. It serves as a database of record, but is also capable of analytics.

The business benefit that this convergence of operations and analytics can bring is dramatic. Three key reasons why enterprises are turning to HTAP are as follows:

**Enabling new sources of revenue.** An example of this can be illustrated in financial services, where investors must be able to respond to market volatility in an instant. Any delay is money out of their pockets. HTAP is making it possible for these organizations to respond to fluctuating market conditions as they happen, providing more value to investors.

**Reducing administrative and development overhead.** With HTAP, data no longer needs to move from operational databases to separated data warehouses or data marts to support analytics. Rather, data is processed in a single system of record, effectively eliminating the need to extract, transform, and load (ETL) data. This benefit provides much welcomed relief to data analysts and administrators, as ETL often takes hours, and sometimes days, to complete.

**Real-time analytics.** Many databases promise to speed up applications and analytics. However, there is a fundamental difference between simply speeding up existing business infrastructure and actually opening up new channels of revenue. True “real-time analytics” does not simply mean faster response times, but analytics that capture the value of data before it reaches a specified time threshold, usually some fraction of a second.

Combined, these three key features allow companies to see real business results from big data. Until recently, technological limitations have necessitated maintaining separate workload-specific data stores, which introduces latency and complexity, and prevents businesses from capturing the full value of real-time data.

HTAP-enabled databases are filling the void where big data promises have failed to deliver. A hybrid data processing model gives enterprises real-time access to streaming data, providing faster and more targeted insights. The ability to analyze data as it is generated allows businesses to spot operational trends as they develop rather than reacting after the fact. For applications like data center monitoring, this helps reduce or eliminate downtime.
For applications that require monitoring a complex dynamic system, such as a shipping network, for instance, it allows analysts to “direct traffic” and make real-time decisions to boost revenue.

As such, organizations focused on realizing quick business value on new and growing data sources should look for an HTAP system with in-memory storage, compiled query execution, enterprise-ready fault tolerance, and ACID compliance. ■

Eric Frenkel is co-founder and CEO of MemSQL. Before MemSQL, Eric worked at Facebook on partnership development. He has worked in various engineering and sales engineering capacities at both consumer and enterprise startups. Eric is a graduate of Stanford University’s School of Engineering. In 2011 and 2012, Eric was named to Forbes’ 30 under 30 list of technology innovators.
Big ETL: The Next ‘Big’ Thing

by Joe Caserta and Elliott Cordo

There is no question that data is being generated in greater volumes than ever before. In addition to vast amounts of legacy data, new data sources such as sensors, application logs, IOT devices, and social networks further complicate data-processing challenges. What’s more, to drive business revenue and efficiency, IT is pressed to acquire new data and keep up with ever-expanding storage and processing requirements.

As businesses look beyond the relational database for solutions to their big data challenges, Extract, Transform, Load (ETL) has become the next component of analytic architecture poised for major evolution. Much new data is semi-structured or even non-structured, and constantly evolving data models are making the accepted tools for structured data processing almost useless.

Because the majority of available tools were born in a world of “single server” processing, they cannot scale to the enormous and unpredictable volumes of incoming data we are experiencing today. We need to adopt frameworks that can natively scale across a large number of machines and elastically scale up and down based on processing requirements. Like the conceptual tipping point that brought to life the term “big data,” we have reached the same scale of evolution with ETL. We therefore nominate the term “Big ETL” to describe the new era of ETL processing.

We’ll define Big ETL as having a majority of the following properties (much like the familiar 4 Vs):

- The need to process “really big data” — your data volume is measured in multiple Terabytes or greater.
- The data includes semi-structured or unstructured types — JSON, Avro, etc.
- You are interacting with non-traditional data storage platforms — NoSQL, Hadoop, and other distributed file systems (S3, Gluster, etc).

Powered by Open Source

Unlike traditional ETL platforms that are largely proprietary commercial products, the majority of Big ETL platforms are powered by open source. These include Hadoop (MapReduce), Spark, and Storm. The fact that Big ETL is largely powered by open source is
interesting for several reasons: First, open-source projects are driven by developers from a large number of diverse organizations. This leads to new added features that reflect a varied set of challenges across solutions and industries. This creates a community of the developers and users working together to improve the product.

Second, one of the most important features of ETL platforms is the ability to connect to a range of data platforms. Instead of waiting for a vendor to develop a new component, new integrations are developed by the community. If you need to connect an MR pipeline to Redis NoSQL, or build a Spark SQL job on top of Cassandra, no problem. Chances are someone has already done this and open sourced their work. And if you don’t find what you need, you can build and open source it yourself.

Third, and perhaps most important, the fact that these engines are open source (free) removes barriers to innovation. Organizations that have a great use case for processing big data are no longer constrained by expensive proprietary enterprise solutions. By leveraging open-source technology and cloud-based architecture, cutting-edge systems can be built at a very low cost.

**Tooling: DSLs, not GUIs**

Unlike traditional ETL tools, most of the tooling for Big ETL does not necessitate traditional GUI-based development. Those familiar with the traditional landscape will recognize that almost all of these ETL tools leverage a palette-based, drag-and-drop development environment. In the new world of Big ETL, development is largely accomplished by coding against the platform’s APIs or through high-level Domain Specific Languages (DSLs). The DSLs include Hive, a SQL like framework for developing big data processing jobs, and Pig, a multipurpose procedural programming language.

To some extent, this return to coding is a reflection of landscape maturity. As time goes on, we likely will see more GUI-based development environments popping up. But, more and more, this movement is questioning whether the value proposition of traditional graphical ETL tools is really true. Is graphical coding always more efficient than code? Does this concept create obstacles when we need to represent very customized and complex data flows? Has the promise of Metadata and reusability been fully delivered by GUI based tools? And, finally, are these concepts not just as possible, and just as central, in code?

With the advent of YARN, Hadoop 2’s new resource manager, we will see an increasing number of legacy tools adapt big data processing capabilities, but the movement to code will definitely continue.

**Big Data Needs More ETL**

Because of the inability of NoSQL and Hadoop to perform ad-hoc joins and data aggregation, more ETL is required to pre-compute data in the form of new data sets or
materialized views needed to support end-user query patterns. In the NoSQL world, it is common to see the same event appear in several rows and/or collections, each aggregated by different dimensions and levels of granularity.

Access to most dimensional data must be “denormalized” into relative events or facts. This means Big ETL now has an increased payload in materializing these additional views. Additionally, process orchestration, error recovery, and data quality become more critical than ever to ensure there are no anomalies caused within the required data redundancy.

The bottom line is not only do we have to process enormous data volumes, but also we need to process it to a greater extent, as well as take greater care in data quality and data monitoring.

These are exciting times in data management. Innovation is the only sustainable competitive advantage a company can have, and we have seen unprecedented technology breakthroughs over the past few years. IT departments are busy enabling businesses with opportunities that were unimaginable just a few years ago, the open-source community is driving innovation at a mind-bending pace, and Big ETL will continue to tackle new and exciting data management challenges, displacing brittle legacy architectures throughout the enterprise.

Joe Caserta, President and CEO of Caserta Concepts, is a veteran solution provider and co-author of the industry best seller *The Data Warehouse ETL Toolkit*. Joe has built big data and data warehouse solutions for businesses and organizations in eCommerce, Insurance, Finance, Healthcare, Retail and Education.

Joe is dedicated to providing big data, data warehouse and business intelligence consulting and training services to help businesses realize the value in their data and to gain new business insights, the key to competitive strength.

Elliott Cordo is a big data, data warehouse, and information management expert with a passion for helping transform data into powerful information. He has more than a decade of experience in implementing big data and data warehouse solutions with hands-on experience in every component of the data warehouse software development lifecycle. As chief architect at Caserta Concepts, Elliott oversees large-scale major technology projects, including those involving business intelligence, data analytics, big data, and data warehousing.

Elliott is recognized for his many successful big data projects ranging from big data warehousing and machine learning to his personal favorite, recommendation engines. His passion is helping people understand the true potential in their data, working hand-in-hand with clients and partners to learn and develop cutting edge platforms to truly enable their organizations.
Why Open Source Storage May Not Be for You

by Russ Kennedy

Open source storage (OSS) software has been in the news a lot in 2014. This reflects the growing interest in open-source technology in general, which relies on software with a source code generally given to the public free of charge by a public collaboration project (often, some commercial vendors also offer a supported distribution of the software for a service fee).

Consider that OpenStack, a four-year-old project managed by the nonprofit OpenStack Foundation, drew 6,000 developers and others to the OpenStack Summit in Paris in early November, 2014. That's one-third more attendees than attended a summit in Atlanta in May, 2014, where attendance had climbed 50 percent from a Hong Kong Summit six months before that.

Indeed, the OpenStack technology market is expected to increase to $3.3 billion in revenue in 2018 from $883 million in 2014, according to a global market study by 451 Research.

As for do-it-yourself (DIY) storage based on open-source technology, it offers several advantages for data storage needs. But it also comes with significant risks, infrastructure shortcomings, and major cost commitments. In short, despite all the buzz, OSS software isn’t for everyone. In fact, it may not be right for the majority of companies storing and managing hyperscale levels of data. Let’s explore its pros and cons.

Advantages of Open-Source Storage

OSS software can be used to build a functioning storage system, enabling an enterprise to bring new servers and applications online within minutes. This can solve the problem of storing data at high speed. That’s good news because the global volume of digital data doubles roughly every two years.

Many IT managers also like the price and flexibility they believe OSS provides. OpenStack Swift, Ceph and Gluster, for instance, are appealing because they push the outer limits of distributed object-storage systems and can essentially be deployed and used for free. Hadoop, another major OSS software framework — indeed, ecosystem — for distributed storage and processing of big data, delivers computing capabilities previously delivered only by supercomputers.
What users say they get from OSS is reliable, high-performing software with the flexibility to shape its configuration to the needs of a situation. And they don’t have to worry about, say, the viability of a given supplier.

In short, more enterprises are inching their way into the technology, using it for test and development requirements, backup and archive workloads, or new applications. Some organizations will see an expansion of OSS deployments in their infrastructure in the coming years.

**Disadvantages of Open-Source Storage**

The shortcomings of OSS are formidable for most organizations. First, while the software may be free, the hardware isn’t. Most OSS solutions use multi-part replication as the primary form of data protection. Some are starting to implement more advanced forms of data protection, like erasure coding, but the maturity level of this technology is low. With replication, organizations may need to purchase three to five times the hardware to protect critical assets from drive, node, or site failure.

Hardware costs are a factor, as are other costs associated with development, downtime, and data loss, which can be more prevalent with open-source technology. For the infrastructure, enterprises may need to recruit specialists familiar with open-source projects to help deploy, support, and develop open-source-specific systems for each enterprise.

Then there are higher support and maintenance costs, which can represent 70 percent of an IT budget with the use of OSS software. Why? Because IT organizations assume the risk and problems associated with support and maintenance of OSS.

Two other major issues involve security and downtime. The recent Heartbleed computer bug exposed vulnerabilities within OSS systems and raised questions about how quickly a development team can react to attacks, and about the potential downtime required to fix an issue such as Heartbleed.

The rise in support and maintenance costs associated with OSS software significantly increases operating expenses, even though it might reduce capital spending marginally. Budget-minded CFOs and financially conservative businesses in general may decide that the extra costs tip the scale against OSS. It reflects the issues raised earlier.

Related to costs, an OSS solution will require a dedicated set of resources to monitor service levels and debut, test, and develop the underlying open-source platform for an enterprise’s needs. Special expertise may be involved, which will increase operational costs.

Then issues such as liability, responsibility and governance must be considered. These pose a major risk to enterprises with OSS software. Take liability: Who is accountable for failure? Users don’t pay a licensing fee, so they incur no real damages or reimbursement.
Developers and maintainers of an open-source project aren’t liable either — the no-fee license generally includes a blanket disclaimer and release. In other words, if your data is lost, or your system has major downtime, there is no one to make things right.

Next comes responsibility: Who ensures that no data is lost? This involves determining who installs a new feature or fixes a defect, as well as handling leaks, denials-of-service, and updates. Even more unfortunate: these risks can lead to data corruption, which may be difficult to detect or determine the source.

Another large responsibility associated with DIY systems is staying up to date with the latest updates and patches that come from the OSS community. Many times, IT teams building on OSS systems may end up maintaining their own custom stream of code that is not leveraging the latest OSS features.

As for governance, who will make sure the data is stored properly and is uncompromised? There is no way for an auditor to guarantee compliance to corporate policies if there is no way to control or change how certain applications store data.

All in all, OSS software is gaining traction. It may prove the best solution for enterprises with a major need for specific features that traditional vendors and products cannot deliver. Yet, for enterprises with petabyte-scale storage needs, OSS can become too complicated to manage and, for many, it has too many unforeseen costs with data loss, downtime, additional hardware, and more.

OSS also carries a level of risk that’s probably too high for many enterprises, especially financially conservative companies that don’t go overboard with tech investments. These organizations should consider a more mature and supportable infrastructure that uses software that’s already been integrated and tested. They also must consider how their platform can deliver the necessary stability, uptime, and scalability.

It may turn out that the risks run too high, and the “free” storage software wasn’t as free as they thought.

Russ Kennedy is senior vice president of product strategy and customer solutions at Cleversafe, the industry-leading object-based storage provider.
Rethinking DBMS:
The Advantages of an Agile Infrastructure

by Cory Isaacson

Big data is growing fast, and it isn’t going to slow down. Companies across a broad spectrum of industries are using data to govern decision making at every turn, from personnel decisions to marketing and information management. As the caches of data continue to grow, the need for agile management systems grows alongside.

We are collecting data from a gamut of sources, including social networks, mobile devices, system data, and, increasingly, the burgeoning Internet of Things. Current database management systems are struggling to keep up with the demands of the growing volume of data and are sorely in need of a re-evaluation. To achieve the next level of database management, we need to ask ourselves, “What if we’ve been looking at databases wrong this entire time?” What if there is an alternative perspective — an agile perspective — that would open up a veritable well of new ideas about how we develop and use databases?

Traditionally, we’ve looked at databases as static repositories: collect data, input data, retrieve data, by any means necessary. Developers and architects start with a modeling process to meet the early requirements, and queries are straightforward as the schema addresses the first features of the application. In the early phase, these features are easy to implement.

Inevitably, however, the data structure becomes more complex over time, requiring new features, which adds to the big data challenge. As the application is relied on to do more and increasingly complex tasks, the database grows in complexity as well. Invariably, this makes the database far more difficult to work with and often results in severe performance degradation.

To avoid a decline in performance, developers often resort to using multiple database engines for a single application, using one or more new engines that support performance and data requirements more closely. This adds even more complexity to the environment and an additional burden on the application developer — now multiple DBMS engines must be kept in sync from the application code. While this can be an effective workaround, it remains just that: a workaround. As more and more engines are added, things become
more complex, forcing developers to interact with multiple data structures via multiple languages.

Sharding, which segments data across an engine by using a key, is a common technique to improve database performance. It can be effective, but when data is partitioned by key, it means that many operations cannot be performed on a single shard. This requires multiple shards and, again, increases both complexity and latency.

These issues are commonly understood, and while portioning can indeed help solve the problems temporarily, it has some effects on the database structure that aren’t as obvious. For instance, a key can support a certain subset of application requirements, but it makes other requirements cumbersome and reduces performance due to increased latency.

With this in mind, let’s take a step back. If we consider the possibility that we have been looking at databases incorrectly, we can begin to consider alternatives that can result in meaningful improvements. One such alternative is the concept of an agile big data approach. An agile big data approach is a completely new perspective of database infrastructures. It takes the traditional view — databases as static repositories — and instead looks at databases in real-time views and dynamic streams. Agile big data allows developers and managers to view data as it comes in rather than waiting for complete compilation before beginning the modeling process. Views are built and maintained by the agile infrastructure itself, on an incremental basis in real time. In other words, views are a picture of the data as it happens. Views can be structured as indexes on other data, data in specific formats for easy application access, aggregate data, and external data engines such as data warehouses. The key is that all types of views are built and maintained from the dynamic, real-time stream of data. You can then build dynamic views with that data — views that exactly match given application requirements.

Anyone using data to guide decision making, analytics, or organization understands the growing need for real-time streaming and specific views to mirror requirements. And because these mirrors can be manipulated in a step-wise fashion, queries become simple, and answers and meaning can be obtained now, in real time. Turning a static repository into a dynamic stream allows us to take virtually any data source — applications, files, existing databases — and quickly solve many problems, such as the aforementioned high latency or exponential complexity birthed by too much portioning, increasing the capabilities of the infrastructure.

Rather than relying solely on separate engines or shards, the agile approach relies on real-time streams and views — views that exactly match application requirements. These views can be structured as indexes on other data, data in specific formats for easy application access, aggregate data, and can even exist in external data engines such as data warehouses. Further, an agile big data infrastructure can be implemented in existing infrastructures without changing existing database systems. For example, it is possible to connect to existing database technologies (e.g., MySQL, Oracle, MongoDB), and “tap
into” their transaction logs or supported Change Data Capture capabilities. These streams can then be used just like any other source (files, direct API access, network streams, etc.), pushing data through the infrastructure as a real-time flow. The streams can then be used to manipulate data into views that can be queried, exposing the exact elements needed by given features of an application. An agile view of big data provides an adaptable, versatile infrastructure that has the ability to evolve rapidly along with application requirements, delivering on the promise of true business agility in the enterprise.

Cory Isaacson is the CEO/CTO of CodeFutures Corporation. Cory has more than 20 years’ experience with advanced software architectures and has worked with many of the world’s brightest innovators in the field of high-performance computing. Cory has spoken at hundreds of public events and seminars and helped numerous organizations address the real-world challenges of application performance and scalability. In his prior position as president of Rogue Wave Software, he actively led the company back to a position of profitable growth, culminating in a successful acquisition by a leading private equity firm. Cory can be reached at cory.isaacson@codefutures.com.
Are Shared-Nothing SQL Database Clusters Right for Your Business?

by Robert Hodges

Shared-nothing database clusters ensure high availability and performance for enterprise applications by spreading data across multiple, automatically maintained replicas. The shared-nothing approach is especially well suited for open source database systems like MySQL and PostgreSQL, which handle a wide variety of business-critical data on commodity hardware. Both database management system (DBMS) types include a multiplicity of powerful and cheap replication mechanisms and, consequently, a multiplicity of ways to build clusters.

Yet that very multiplicity can hide important pitfalls. Approaches that work well with small data sets on local replicas may perform very poorly as transaction load scales or as the replicas are spread over distance.

Making a good clustering choice requires a clear understanding of cluster methodology, including replication. It also requires a clear understanding of what applications actually need from clusters. Let’s take marketing campaign automation, for example, which helps companies set up marketing campaigns across channels like email, social media, or websites and then track responses.

Marketing campaign processing can generate thousands of transactions per second for large organizations, so data volumes as well as transaction throughput are often enormous. Campaign management includes a wide mix of transactions, ranging from single row updates to operational reports to batch operations that may change millions of rows. Users have little patience for slow performance or downtime of any kind. Therefore, successful clustering must completely hide failures as well as maintenance from users while delivering uncompromising high performance across the entire workload spectrum.

In deciding whether a clustering approach cuts the mustard, two questions are of paramount importance: whether the cluster uses a multi-master or primary/secondary design and whether replication is synchronous or asynchronous.

In the multi-master approach, applications can read and write to any replica in the cluster. In the primary/secondary approach, applications read and write to a single primary replica but at most perform reads on the secondary replicas.
Synchronous replication applies changes to either all or at least the majority of the replicas (a “quorum”) within the transaction before signaling to client applications that updates were successful. Asynchronous replication sends the update as quickly as possible but does not make the application wait for transactions to replicate to multiple locations after commit.

At first glance, it would appear to most programmers that synchronous multi-master is the best choice for building clusters. Multi-master means applications can update at any node. Synchronous replication means that committed transactions will not be lost if a single replica fails and that all master nodes are consistent, which greatly simplifies the problems of load balancing application reads.

The trouble is that these desirable properties have high practical costs, especially for large applications.

Let’s first look at synchronous replication. The speed at which an update completes is dependent on the speed of the slowest replica, which means application connections have to wait longer for each commit. Synchronous replication works best when replicas are within a single data center and can respond very quickly. Unfortunately, it is a common best practice to put replicas in different data centers to ensure high availability. But doing this can destroy application performance. If the round-trip time between data centers is just 2 milliseconds, application transaction rates on single connections can drop by an order of magnitude compared with accessing a local DBMS. For cross-region replication, the effect is far greater. Single DBMS connections may be limited to a best-case maximum of 10 transactions per second or less.

What’s worse is that production systems rarely operate at best case throughput due to load, network congestion, and a host of other issues. The result is that fully synchronous replication is not used widely in the field, especially for applications that require very high transaction throughput. Outside of special cases at companies like Google, clusters that operate over distance or at scale tend to use asynchronous replication.

Multi-master operation has its difficulties as well. Updating data in multiple locations creates the potential for conflicts, which can lead to replicas becoming inconsistent. This is a very difficult problem to solve, but an approach that is growing in popularity in several new clustering products is known as “optimistic locking.” The cluster allows transactions to be handled on any replica but includes a check to ensure no other replica has processed a transaction that might include conflicting changes. If so, one or both of the conflicting changes must roll back. Optimistic locking seems promising at first glance, but turns out to be old wine in new bottles for many applications because such rollbacks can trigger a well-known problem with distributed database locking, which is prone to appear as transaction activity scales up.

Transactions that affect many rows, including data definition language (DDL) changes, may cause large numbers of conflicts with corresponding numbers of failures for user applications. Failures become most severe when operating at scale or when there is distance between replicas that extends the window for conflicts. This behavior has been
well understood for many years, and Jim Gray described the details in a famous 1996 paper, “The Dangers of Replication and a Solution.” Such failures may require extensive application fixes to resolve, which can be extremely painful if they occur in businesses with deployed systems and rapidly increasing load.

The somewhat counter-intuitive result is that asynchronous, primary/secondary clustering works better than synchronous multi-master for a wide variety of applications. Primary nodes behave like a normal DBMS instance with no limitations on SQL and minimal performance penalty for replication. Asynchronous replication also means that application performance remains essentially the same, regardless of the number of replicas. Moreover, you can speed up the primary by load-balancing read-intensive operations to replicas. This is not hard to program and is a common feature in many MySQL applications. Clustering products based on the primary/secondary model can even help route reads to secondary replicas, further simplifying deployment.

Asynchronous replication has another strength. A secondary node can lose its connection for days at a time and then reconnect when the master is once again available. All you have to do is ensure your logs do not run out. For the same reason, primary/secondary replication works extremely well over wide area networks (WANs) and is commonly used for constructing disaster-recovery sites. The ability to be disconnected for prolonged periods of time also means that such clusters offer good support for rolling maintenance.

One practical drawback of primary/secondary clustering is replica failures. Detecting that the primary has failed, selecting a suitable secondary to replace it, and reconfiguring the system with minimal disturbance to applications is complex and rife with difficult corner cases. Failover is a problem best solved with a mature clustering solution based on solid principles of distributed computing.

The state of the art in shared-nothing clusters is changing as business requirements evolve, leading to new approaches to balance high availability, data protection, and performance. One interesting mix is to use synchronous replication between replica pairs located a short distance apart while using asynchronous replication over longer distances. This approach largely eliminates data loss in cases where it is a problem while minimizing performance effects.

Shared-nothing clustering belongs in the palette of IT architects looking to maximize throughput and availability of applications. Like any powerful tool, users need to understand how to apply and manage it properly to ensure successful growth of their data-driven businesses.

Robert Hodges is Chief Executive Officer of Continuent, a leading provider of open source database clustering and replication solutions. For more information about Continuent, email the company at info@continuent.com, visit their web site, or call 866-998-3642.
How to Innovate Using Multi-Source, Multi-Structured Data

by Harmeek Bedi

It’s pretty easy to build a house if you have nails, wood, plaster, and insulation. It’s not so easy if you have nails, flour, yarn, and fertilizer.

In some ways, this is the challenge software developers face in today’s world of multi-source, multi-structured data. Innovating in an environment where data is exploding in its variety, size, and complexity is no simple task. Social data (comments, likes, shares, and posts), the Internet of Things (sensor data, motion detection, GPS), and advertising data (search, display, click-throughs) are just a few of the thousands — perhaps millions — of heterogeneous, dynamic, distributed, and ever-growing devices, sources, and formats that are driving the big data revolution.

It would be convenient to ignore the immense benefits of cross-utilizing all these disparate data sets. After all, like the raw materials needed to build a house, task-specific data is best suited for the job immediately at hand. Yet bringing together data from multiple sources can provide insights much more powerful than those from each source separately. For example, drawing data from satellites, city engineering data for buildings and roads, topographic information, user inputs, and so on makes accurate turn-by-turn navigation through maps applications possible. Similarly, e-commerce companies can leverage local customer, product, and store information as well as weather and location data to optimize inventory management and enable real-time, customer-specific offers.

Products and services leveraging multi-source, semi-structured data allow businesses to compete better and organizations to function more efficiently. They facilitate new business models and provide a deeper understanding of customers and constituents, problems, and phenomena. Most of all, they allow developers to innovate and uncover the possibilities of an interconnected world.

New Methods Needed

The challenges of analyzing this new amalgam of heterogeneous data are as complex and multi-layered as the data themselves. At its core, this is a software engineering problem: no one piece of software can do everything that developers will want to do. New methods are needed to overcome the size, bandwidth, and latency limitations of conventional relational database solutions.
In recent years, several critical technologies have evolved to address the problem of cross-utilizing semi-structured data. JavaScript Object Notation (JSON), a platform- and application-independent format, has gained popularity in virtually any scenario in which applications need to exchange or store data. Because the schema is carried along with the data, JSON allows data structures to evolve as the application changes without triggering downstream modeling changes. However, it must be collected and stored — and traditional Relational Database Management Systems (RDBMS) are not designed for this. NoSQL databases that allow developers to store and retrieve large collections of JSON documents quickly are fast becoming the new Online Transactional Processing (OLTP) stores.

So the transmission, collection, and storage of semi-structured data in native formats is now possible, as is the ability to scale this infrastructure to use data in a cost-effective manner as volumes and formats grow, without upfront planning. But this brings us to the next fundamental challenge: how to enable today’s business analysts to analyze the semi-structured data interactively, despite its complexity and diversity, for the discovery of insights.

**New Age of Analytics**

The Internet is awash with hype around RDBMS vs. Hadoop vs. NoSQL DBMS, with no clarity about when one should be used over the other — and for what kinds of analytical workloads.

Hadoop, one of the best-known processing frameworks, has the power to process vast amounts of semi-structured as well as structured data. Its appeal lies in its versatility, its high aggregate bandwidth across clusters of commodity hardware, and its affordability (at least in its pure, open-source form). However, Hadoop was designed for batch processing in a programming language familiar only to developers—not interactive ad hoc querying using a declarative language like SQL.

For these reasons, there is growing interest in developing interactive SQL engines on top of the same Hadoop cluster. There are open-source projects that attempt to set up Hadoop as a queryable data warehouse, but these are just getting started. Their task is daunting — no less than trying to re-invent a database on top of Hadoop.

Such projects offer very limited SQL support (HiveQL) and are typically lacking in SQL functions such as subqueries, “group by” analytics, etc. They rely on the Hive metastore, which requires defining table schemas up front for the semi-structured data attributes that you want to analyze, in order to allow an SQL-like language to manipulate this data. This is a self-defeating strategy. To explore and understand your multi-source data, you first must know it well enough to define its attributes and schema up front.

NoSQL databases like MongoDB have a built-in query framework to interrogate semi-structured data. But now you are burdening an operational database with the overhead of data access for longer-running analytical queries. This will cause conflicts as the data
and usage grow. Additionally, Mongo’s query framework requires an understanding of how the data is physically laid out to avoid running into syntax, memory, and performance limitations on large data sets.

Things we take for granted in investigative analysis, such as joining data stored in two separate tables directly from within a query, queries with multiple values, and conditions or ranges not known up front are simply not possible using Mongo’s native analytics capabilities.

DWS for Semi-Structured Data

An advanced analytics platform for multi-source, semi-structured data sets in which ad hoc queries require scanning and joining of data across billions of records requires a more sophisticated approach. In particular, SQL as a query language is a must, to support broad use and deliver insights to decision makers quickly.

The answer can be found in a new class of Data Warehouse Service (DWS) designed for fast, low-latency analytics. In these services, data is stored in its original format with support for JSON, XML, and Key Values as native data types. This preserves the richness of the data and also circumvents the need for complex extract, transform, load (ETL), or any up-front modeling, before analysis.

With this class of DWS, developers can create solutions that not only leverage multi-source data for current needs, but also actually support as-yet undiscovered solutions that involve data not yet leveraged. In fact, in the more high-performing DWS offerings, analysts can access and work with data directly, using their favorite SQL-based BI tool and user-defined functions. This is because such offerings speak ANSI SQL, and SQL-2011 online analytical processing (OLAP) operators work directly over JSON, allowing workers with knowledge of SQL-based BI tools, modeling techniques, and accompanying domain expertise to operate in the semi-structured world. “What if” questions take on a whole new dimension because data can be cross-referenced from literally dozens of sources to provide insights never before possible.

DWS offers multiple benefits. First, because it exists in the cloud, all the cost and manpower savings of a cloud-based service accrue, including a huge drop in hardware expenditures, lower administrative effort, and the ability to scale on commodity hardware as data and performance needs grow. DWS not only eliminates the need to create and maintain a separate analytics infrastructure alongside an organization’s operational systems, but also reduces impact on transactional stores such as MongoDB, thus helping the organization meet its dashboard and reporting SLAs.

Second, by storing JSON datasets directly, DWS takes away the need to write custom code to build, collect, and integrate data from various streaming and semi-structured sources. Fresh, detailed data is available immediately on load for quick analytics and discovery.
Analytics is resilient to schema changes and provides the same application flexibility as NoSQL, while data is preserved in its native format — it doesn’t lose semantic value through conversion (for example, when converting JSON to text).

Third, the ability to use SQL as the query language delivers powerful business advantages. It allows business users to do their own analysis, thereby freeing up developers from having to write Java or other code to get data for every question an analyst might have. Furthermore, the SQL capability of DWS lets organizations leverage the well-established ecosystem of SQL-based analytics tools.

The future of big data lies in the advent of tools and techniques that derive value out of multi-source, semi-structured data. Developers need to innovate with these solutions that support ad hoc queries with multiple values, conditions and ranges—the kind of intelligent questions made possible when “sum of all knowledge” systems have finally arrived. When speed and flexibility are paramount, developers must look to new solutions like advanced DWS to provide the answers. Like an architect given the rare opportunity to create using a whole new set of raw materials, when heterogeneous data is assembled from the new world of information, there’s no telling what amazing structures might arise in the future.

Harmeek Singh Bedi is CTO of BitYota. Harmeek brings 15+ years of experience building database technologies at Oracle and Informix/IBM. At Oracle, he was a lead architect in the server technology group that implemented partitioning, parallel execution, storage management, and SQL execution of the database server. Prior to BitYota, he spent 2+ years at Yahoo! working on Hadoop and big data problems. Harmeek holds 10+ patents in database technologies.
Modernizing M2M Analytics Strategies for the Internet of Things

by Don DeLoach

A jet airliner generates 20 terabytes of diagnostic data per hour of flight. The average oil platform has 40,000 sensors, generating data 24/7. In accordance with European Union guidelines, 80 percent of all households in Germany (32 million) will need to be equipped with smart meters by 2020.

Machine-to-machine (M2M) sensors, monitors and meters like these will fuel the Internet of Things. M2M is now generating enormous volumes of data and is testing the capabilities of traditional database technologies. In many industries, the data load predictions of just 12 to 24 months ago have long been surpassed. This is creating tremendous strain on infrastructures that did not contemplate the dramatic increase in the amount of data coming in, the way the data would need to be queried, or the changing ways business users would want to analyze data.

To extract rich, real-time insight from the vast amounts of machine-generated data, companies will have to build a technology foundation with speed and scale because raw data, whatever the source, is only useful after it has been transformed into knowledge through analysis. For example, a mobile carrier may want to automate location-based smartphone offers based on incoming GPS data, or a utility may need smart meter feeds that show spikes in energy usage to trigger demand response pricing. If it takes too long to process and analyze this kind of data, or if applications are confined to predefined queries and canned reports, the resulting intelligence will fail to be useful, resulting in potential revenue loss.

Investigative analytics tools enable interactive, ad-hoc querying on complex big data sets to identify patterns and insights and can perform analysis at massive scale with precision even as machine-generated data grows beyond the petabyte scale. With investigative analytics, companies can take action in response to events in real-time and identify patterns to either capitalize on or prevent an event in the future. This is especially important because most failures result from a confluence of multiple factors, not just a single red flag.

However, in order to run investigative analytics effectively, the underlying infrastructure must be up to the task. We are already seeing traditional, hardware-based infrastructures
run out of storage and processing headroom. Adding more data centers, servers and disk storage subsystems is expensive. Column-based technologies are generally associated with data warehousing and provide excellent query performance over large volumes of data. Columnar stores are not designed to be transactional, but they provide much better performance for analytic applications than row-based databases designed to support transactional systems.

Hadoop has captured people’s imaginations as a cost-effective and highly scalable way to store and manage big data. Data typically stored with Hadoop is complex, from multiple data sources, and includes structured and unstructured data. However, companies are realizing that they may not be harnessing the full value of their data with Hadoop due to a lack of high-performance ad-hoc query capabilities.

To fully address the influx of M2M data generated by the increasingly connected Internet of Things landscape, companies can deploy a range of technologies to leverage distributed processing frameworks like Hadoop and NoSQL and improve performance of their analytics, including enterprise data warehouses, analytic databases, data visualization, and business intelligence tools. These can be deployed in any combination of on-premise software, appliance, or in the cloud. The reality is that there is no single silver bullet to address the entire analytics infrastructure stack. Your business requirements will determine where each of these elements plays its role. The key is to think about how business requirements are changing. Move the conversation from questions like, “How did my network perform?” to time-critical, high-value-add questions such as, “How can I improve my network’s performance?”

To find the right analytics database technology to capture, connect, and drive meaning from data, companies should consider the following requirements:

**Real-time analysis.** Businesses can’t afford for data to get stale. Data solutions need to load quickly and easily, and must dynamically query, analyze, and communicate M2M information in real-time, without huge investments in IT administration, support, and tuning.

**Flexible querying and ad-hoc reporting.** When intelligence needs to change quickly, analytic tools can’t be constrained by data schemas that limit the number and type of queries that can be performed. This type of deeper analysis also cannot be constrained by tinkering or time-consuming manual configuration (such as indexing and managing data partitions) to create and change analytic queries.

**Efficient compression.** Efficient data compression is key to enabling M2M data management within a network node, smart device, or massive data center cluster. Better compression allows for less storage capacity overall, as well as tighter data sampling and longer historical data sets, increasing the accuracy of query results.

**Ease of use and cost.** Data analysis must be affordable, easy-to-use, and simple to implement in order to justify the investment. This demands low-touch solutions that
are optimized to deliver fast analysis of large volumes of data, with minimal hardware, administrative effort, and customization needed to set up or change query and reporting parameters.

Companies that continue with the status quo will find themselves spending increasingly more money on servers, storage, and DBAs, an approach that is difficult to sustain and is at risk of serious degradation in performance. By maximizing insight into the data, companies can make better decisions at the speed of business, thereby reducing costs, identifying new revenue streams, and gaining a competitive edge.

Don DeLoach is CEO and president of Infobright. Don has more than 25 years of software industry experience, with demonstrated success building software companies with extensive sales, marketing, and international experience. Don joined Infobright after serving as CEO of Aleri, the complex event processing company, which was acquired by Sybase in February 2010. Prior to Aleri, Don served as President and CEO of YOUcentric, a CRM software company, where he led the growth of the company’s revenue from $2.8M to $25M in three years, before being acquired by JD Edwards. Don also spent five years in senior sales management culminating in the role of Vice President of North American Geographic Sales, Telesales, Channels, and Field Marketing. He has also served as a Director at Broadbeam Corporation and Apropos Inc.
Is SQL-on-Hadoop Right for Your Real-Time, Data-Driven Business?

by Monte Zweben

As many enterprises start focusing on gaining better business insights with Big Data, it would be careless for them to overlook the advances that Hadoop is making with SQL.

SQL-on-Hadoop brings the most popular language for data access to the most scalable database framework available. Because SQL queries traditionally have been used to retrieve large amounts of records from a database quickly and efficiently, it is only natural to combine this standard interactive language with Hadoop’s proven ability to scale to dozens of petabytes on commodity servers. As a result of these converging technologies, there is plenty of optimism that the promise of Hadoop can be realized. Previous challenges of not being able to get data in and out of Hadoop are now mitigated through the flexibility of SQL.

But access and flexibility are only part of the SQL-on-Hadoop marriage. We’ve seen firsthand how customers interact with real-time, transactional SQL-on-Hadoop databases. They may be familiar with a variety of databases, from traditional relational database management systems (RDBMS) such as MySQL and Oracle to a new generation of highly scalable NoSQL options such as Cassandra or MongoDB, but SQL-on-Hadoop solutions offer a best-of-both-worlds approach.

Even with their potential benefits, it’s important to tread carefully because not all SQL-on-Hadoop solutions are created equal. Choosing the right one is a critical decision that can have a long-term impact on a company’s application infrastructure. In working closely with enterprises to solve their Big Data challenges, we have identified the top five issues enterprises need to be mindful of when choosing a SQL-on-Hadoop solution:

1. Supporting real-time applications. This includes real-time operational analytics and traditional operational applications such as web, mobile, and social applications as well as enterprise software. Like many companies, one of our customers had real-time applications that required queries to respond in milliseconds to seconds. While this demand can be handled by traditional RDBMS systems, the client also faced growing data volume, which was making its Oracle databases expensive to maintain and scale. When real-time support cannot be compromised, it leads companies to either scale up at great costs or try to
re-create functionality while scaling out. In this instance, our SQL-on-Hadoop database allowed them to execute in real time with an almost 10x price-performance improvement.

2. **Working with up-to-the-second data.** This means real-time queries on real-time data. Some solutions claim to be real-time because they can do real-time ad-hoc queries, but it is not real time if the data is from yesterday’s ETL (Extract, Transform, Load). For example, an e-commerce company we worked with evaluated many SQL-on-Hadoop solutions but found many of them lacking the ability to update data in real-time. This was a critical requirement as they needed to analyze real-time order, pricing, and inventory information to trigger real-time discounts and inventory replenishment orders. While it may not be mission-critical for all applications, up-to-the-second data streams can enable companies to derive maximum business value from their SQL-on-Hadoop investment.

3. **Maintaining data integrity.** Database transactions are required to reliably perform real-time updates without data loss or corruption. They are a hallmark of traditional RDBMS solutions, but we have heard of many enterprises that made the switch to NoSQL solutions and missed the reliability and integrity of an RDBMS. Working with a large cable TV provider, we discovered that transactions are even important in analytics, as data and secondary indices need to be updated together to ensure consistency. For its operational analytics applications, this customer found that it could not reliably stream updates in its SQL-on-Hadoop database without having ACID transactions.

4. **Preserving SQL support.** Many companies have made large investments in SQL over the years. It’s a proven language with published standards like ANSI SQL. This has led to many companies trying to retain standard SQL in their databases, causing them to forgo the NoSQL movement. However, even in some SQL-on-Hadoop solutions, the SQL provided is a limited, unusual variant that requires retraining and partially rewriting applications. One of our customers in the advertising technology space switched from Hive because its limited variant of SQL, known as Hive Query Language (HQL), could not support the full range of ad hoc queries that the company’s analysts required. More and more SQL-on-Hadoop vendors are moving to full SQL support, so it’s important to check SQL coverage when making a decision.

5. **Supporting concurrent database updates.** Many operational and analytical applications are receiving data from multiple sources simultaneously. However, not all SQL-on-Hadoop solutions can support concurrent database updates. This not only can interfere with the recency of the data, but also can lock up the database. One of our customers evaluated an analytic database that provided transactions, but any update or insert would lock the entire table. This meant that a table could support only one update at a time and made it impractical to do significant updates more than a few times a day. For applications with many streaming data sources (such as a website or a sensor array), a reduced frequency of updates can greatly hinder the value it can create for users.
SQL-on-Hadoop applications will play a large role in fueling the growth of the Big Data market. According to IDC, the Big Data industry will surge at a 27 percent compound annual growth rate and reach $32.4 billion in 2017. This increase is indicative of the universal demand to realize the promise of Big Data by becoming real-time, data-driven businesses. Companies able to leverage emerging solutions like SQL-on-Hadoop databases stand to gain the biggest benefits by driving business processes resulting in better customer insight, improved operational efficiencies and smarter business decisions. For companies that see SQL-on-Hadoop solutions as a critical component of their long-term data management infrastructure, it is important that they ask the right questions to ensure that their chosen SQL-on-Hadoop solution can adequately address all of their application needs, including real-time operational and analytical applications.

Monte Zweben is co-founder and CEO of Splice Machine. He founded Red Pepper Software and Blue Martini Software, and is currently on the Board of Directors of Rocket Fuel. He holds a B.S. in Computer Science/Management from Carnegie Mellon University, and an M.S. in Computer Science from Stanford University.
How Database Environments Lose Performance and What You Can Do About It

by Marc Linster

Data volumes are fast expanding and usage patterns are changing just as rapidly. As a result, database environments that were thought to be well-tuned and well-configured quickly become congested, cluttered and inefficient with disparate platforms, sprawling systems, a bloated application portfolio, growing costs and less-than-satisfactory performance. Even the best among us are finding ourselves asking, “How did our well-tuned applications turn into resource hogs that stopped meeting expectations?”

If you took the time to review many database environments — examining everything from hardware and operating systems configuration to the database architecture, and cataloging components that affect scalability, high availability, disaster recovery, manageability, security and tuning — you would find many infrastructures suffer the same maladies.

In the reviews my database company, EnterpriseDB, has conducted, we have found common problems: the usage pattern has changed, more users stay connected for longer, certain data tables have grown faster than expected, some data partitions or indexes are out of balance, a minor software upgrade impacted a configuration parameter, or a well-intended “quick fix” has disastrous long-term performance impacts. For example, one such quick fix that we frequently encounter is a result of an indiscriminant indexing strategy. DBAs create too many indexes, some of them redundant, which may increase read performance, but they will also almost certainly have a calamitous impact on write performance.

Another cause of dwindling database performance is failing to maintain a commercial software package, such as an ERP system. Over the past year, for example, we worked with a number of large and mid-sized health care providers that were using a commercial package for managing medical practices, offices and small clinics without paying enough attention to the underlying database. This particular package embodies a database that, just like all databases, if not well maintained, over time tends to develop problems with performance and data integrity. This happens slowly in the background as the medical practice grows, specifically in situations where users fail to conduct proactive maintenance or upgrades. Users were reporting such problems as data loss, data corruption and recovery issues. In
some cases, it was unclear if these problems were a result of intrusions that were allowed to happen because the software wasn’t upgraded regularly.

**Tactics to Improve Performance**

There are a number of places a data professional can target when things seem to be going awry. When application performance starts to falter or data discrepancies emerge, here’s where to start looking:

- **Memory configuration**: Wrong memory configuration settings can slow the server down dramatically. Memory gets assigned to the database globally, and separately to every database connection. These allocations must be realigned regularly with new usage patterns. For example, more concurrent users require more available memory. At a certain point, the database should adopt a connection pooler to improve the ratio of users to connection memory. And larger tables that are accessed frequently for read-only should be held in-memory for quick access.

- **OS parameters**: Operating system parameters need to be set optimally to support new releases and changing usage profiles.

- **Partitioning Strategy**: Data loads change over time as an application is used, and outdated partitioning strategies that may have been terrific for the data load you had 18 months ago may no longer support the data load you have today. As a result, queries can become painfully slow. Database architects (DBAs) need to review partitions regularly to make sure they are well balanced and that the distribution of the data across the partitions meets business requirements. Furthermore, it’s important that DBAs verify regularly that queries continue to make efficient use of partitions.

- **SQL queries and indexes**: Poorly written SQL queries and misaligned indexes on commonly executed tasks can have significant impact. A thorough analysis of where queries perform sequential scans, have sub-optimal execution plans or could be supported with better indexing strategies, often works wonders. A good indexing strategy eliminates overlapping indexes, to the degree possible, and finds the right balance between write performance (slowed down by indexes) and read performance (enhanced by indexes). A good strategy also considers index-only scans, returning results from indexes without accessing the entire heap.

- **Misaligned Java queries**: Often queries are generated by the Java code, but the queries may not be well supported by the table structures, the partitions or the indexes. Like every other query, these machine-generated queries must be reviewed and the output of the query planner must be analyzed to identify optimization opportunities, such as long sequential scans that could benefit from an index.

By targeting just these areas for adjustment, we have seen data professionals in multiple situations improve performance by 1,000 times.

But while performance degradation is often the first visible sign of data management
problems, by far the worst scenarios result from deficient backup and recovery strategies. Backup strategies must plan for catastrophic failure of a device (for example, a storage system gets corrupted), operator error (a database architect accidentally deletes a table), data corruption (a virus or other defect causes a table corruption that went unnoticed for days or weeks), and compliance requirements (know the rules for archiving and retention).

Given how quickly data volumes have grown and how data usage has changed in recent years, it’s not hard for our databases to become congested, cluttered, inefficient and corrupt. The good news is that simple fixes can do wonders for your performance so getting back on track to good data health can be just as easy.

Marc Linster is senior vice president, products and services at EnterpriseDB. Marc holds a Ph.D. in computer science and has 25 years experience in the technology industry, serving in multiple leadership roles for global firms in engineering, business intelligence and process improvement. He can be reached at marc.linster@enterprisedb.com.
Fast Database MapD Emerges from MIT Student Invention

by Ian B. Murphy

Todd Mostak’s first tangle with big data didn’t go well. As a master’s student at the Center for Middle Eastern Studies at Harvard in 2012, he was mapping tweets for his thesis project on Egyptian politics during the Arab Spring uprising. It was taking hours or even days to process the 40 million tweets he was analyzing.

Mostak saw immediately the value in geolocated tweets for socio-economic research, but he did not have access to a system that would allow him to map the large dataset quickly for interactive analysis.

So over the next year, Mostak created a cost-effective workaround. By applying his analytical skills and creativity, taking advantage of access to education and using hardware designed for computer gamers, he performed his own version of a data science project, developing a new database that solved his problem. Now his inventive approach has the potential to benefit others in both academia and business.

While taking a class on databases at MIT, Mostak built a new parallel database, called MapD, that allows him to crunch complex spatial and GIS data in milliseconds, using off-the-shelf gaming graphical processing units (GPU) like a rack of mini supercomputers. Mostak reports performance gains upwards of 70 times faster than CPU-based systems.

Mostak said there is more development work to be done on MapD, but the system works and will be available in the near future. He said he is planning to release the new database system under an open source business model similar to MongoDB and its company 10gen.

“I had the realization that this had the potential to be majorly disruptive,” Mostak said. “There have been all these little research pieces about this algorithm or that algorithm on the GPU, but I thought, ‘Somebody needs to make an end-to-end system.’ I was shocked that it really hadn’t been done.”

Mostak’s undergraduate work was in economics and anthropology; he realized the need for his interactive database while studying at Harvard’s Center for Middle Eastern Studies program. But his hacker-style approach to problem-solving is an example of how attacking a problem from new angles can yield better solutions. Mostak’s multidisciplinary background isn’t typical for a data scientist or database architect.

Sam Madden, the director of big data at MIT’s Computer Science and Artificial Intelligence Lab (CSAIL), said some faculty thought he was “crazy” for hiring Mostak to work at CSAIL; he has
almost no academic background in computer science. But Mostak’s unconventional approach has yielded one of the most exciting computer science projects at MIT.

Madden said that when a talented person with an unusual background presents himself, it’s key to recognize what that person can accomplish, not what track he or she took to get there. “When you find somebody like that you’ve got to nurture them and give them what they need to be successful,” Madden said of Mostak. “He’s going to do good things for the world.”

Using Tweets to Challenge Assumptions

The start of MapD’s creation was Mostak’s master’s thesis, which tested the theory that poorer neighborhoods in Egypt are more likely to be Islamist. He looked at geocoded tweets from around Cairo during the Arab Spring uprising. He examined if the tweet writer followed known Islamist politicians or clerics.

Mostak lived in Egypt for a year, learning Arabic at American University in Cairo and working as a translator and writer for the local newspaper Al-Masry Al-Youm. He knew the situation leading up to the Arab Spring uprising better than most.

He cross-referenced the language in the tweets with forums and message boards he knew to be Islamist to measure sentiment. He also checked the time stamps to see if Twitter activity stopped during the five daily prayers.

Then he plotted the Islamist indicators from 40 million tweets, ranging from August 2011 through March 2012, against 5,000 political districts from the Egyptian census.

In his first attempt to plot the points on the map using his laptop, he discovered it would take several days to run the analysis. Mostak said at this point he was far from an expert at crunching that size of data, but even with the optimized code the data was too big to get reasonable performance.

“You could do it if you had this big cluster, but what if you’re a normal guy like me?” Mostak said. “There really is a need for something to do this kind of workload faster.”

This choropleth map displays the work from Mostak’s Harvard thesis. It shows the relative frequency of people following Islamists on Twitter in each voting district around Cairo.

Mostak’s Harvard professors helped him get access to better computing resources to finish his thesis. His results, which he plotted on a choropleth map, were suggestive that more rural — not poorer — areas leaned towards Islamism. His thesis won first prize in the department’s annual contest.

After graduating from his master’s program in May 2012, he began a six-month fellowship at the Ash Center for Democratic Governance and Innovation at the John F. Kennedy School of Government with his advisor, Prof. Tarek Masoud, expanding his effort to analyze social media for insight on Egyptian political changes.
Database Class at MIT

But Mostak had found a new problem to solve: looking for a better way to do spatial or GIS analytics. For his last semester at Harvard, he registered for a course on databases taught by Sam Madden at the Massachusetts Institute of Technology.

Mostak said that when he first signed up for the course, he hadn’t yet encountered his big data problem. Mostak graduated from University of North Carolina at Chapel Hill with degrees in economics, anthropology and a minor in math; he was looking to take advantage of the opportunity to learn at MIT while he still had the chance through Harvard.
But as his thesis project began to encounter problems analyzing millions of tweets, Mostak said he saw the class as a chance to better understand how to organize and query data for mapping projects.

“I wanted to know what was going on under the hood, and how to better work with my data,” he said. “That was pretty serendipitous.”

Mostak had already dabbled in programming for 3D graphics with the language OpenGL, when he was making iPhone apps as a hobby. He knew how powerful the top graphical processing units, or GPUs, that hardware companies made were for high end gaming computers.

During the class in the spring of 2012 he learned the graphics programming language CUDA, and that opened the doors for tweaking GPUs to divide advanced computations across the GPUs massively parallel architecture.

He knew he had something when he wrote an algorithm to connect millions of points on a map, joining the data together spatially. The performance of his GPU-based computations compared to the same operation done with CPU power on PostGIS, the GIS module for the open-source database PostgreSQL, was “mind-blowing,” he said.

“The speed-ups over PostGIS … I’m not an expert, and I’m sure I could have been more efficient in setting up the system in the first place, but it was 700,000 times faster,” Mostak said. “Something that would take 40 days was done in less than a second.”

That was with a $200, mid-level consumer graphics card. With two GeForce Titan GPUs made by Nvidia, the fastest graphics card on the market, Mostak’s database is able to crunch data at the same speed of the world’s fastest supercomputer in the year 2000. That machine computer cost $50 million at the time, and ran on the same amount of electricity it took to power 850,000 light bulbs. Mostak’s system, all told, costs around $5,000 and runs five light bulbs worth of power.

Mostak said his system uses SQL queries to access the data, and with its brute force GPU approach, will be well suited for not only geographic and mapping applications but machine learning, trend detection and analytics for graph databases.

Building the GPU database became his final project for Madden’s class.

### MapD, At A Glance:

MapD is a new database in development at MIT, created by Todd Mostak.

- MapD stands for “massively parallel database.”
- The system uses graphics processing units (GPUs) to parallelize computations. Some statistical algorithms run 70 times faster compared to CPU-based systems like MapReduce.
- A MapD server costs around $5,000 and runs on the same power as five light bulbs.
- MapD runs at between 1.4 and 1.5 teraflops, roughly equal to the fastest supercomputer in 2000.
- MapD uses SQL to query data.
- Mostak intends to take the system open source sometime in the next year.
The database was sufficiently impressive; Madden offered Mostak a job at CSAIL. Mostak has been able to develop his database system at MIT while working on other projects for the university.

Madden said there are three elements that make Mostak’s database a disruptive technology. The first is the millisecond response time for SQL queries across “huge” datasets. Madden, who was a co-creator of the Vertica columnar database, said MapD can do in milliseconds what Vertica can do in minutes. That difference in speed is everything when doing iterative research, he said.

The second is the very tight coupling between data processing and visually rendering the data; this is a byproduct of building the system from GPUs from the beginning. That adds the ability to visualize the results of the data processing in under a second. Third is the cost to build the system. MapD runs on a server that costs around $5,000.

“He can do what a 1,000 node MapReduce cluster would do on a single processor for some of these applications,” Madden said.

**MapD and the Harvard World Map**

Mostak’s MapD database system was built initially to solve problems involving millions of points on maps, and Mostak’s development efforts have started with GIS applications.

At his project’s conceptual stage, Mostak found a sounding board and technical advisor in Ben Lewis, the project manager for Harvard’s open source World Map project. The World Map is housed in Harvard’s Center for Geographic Analysis, and it serves as a free collaborative tool for researchers around the world to share and display GIS data.

Mostak’s database came at a great time for Lewis and World Map, just as the number of users began to increase and Lewis was starting to think about how the system would scale. It’s a web service; nearly all of the processing is done on Harvard’s servers.

Lewis has been running a Hadoop system, but at best all the batch-oriented database can do is preprocess data and prepare it for display while running in the background. Mostak’s MapD is instantaneous.

“The thing is this is a whole different animal,” Lewis said. “It’s not using [open source search database] Solr, or not using a different database. It’s truly parallelizing this stuff, and it’s actually searching on the fly through 100 million things without preprocessing. That may seem like a subtle difference, but in terms of what it can enable, in terms of analytics, it’s completely different.”

Lewis has helped Mostak get some funding for hardware to run the system, and with projects to help fund the development of MapD. Through World Map, Mostak worked for the Japan Data Archive, a project to collect data from the 2011 earthquake and tsunami. The project uses MapD to display several data sets on a map instantly.
Mostak is working with Harvard to visualize the Kumbh Mela, a 55-day Hindu religious festival that happens only once every 12 years that will see more than 80 million people attend. Mostak and MapD will visualize anonymized cell phone data to analyze crowd flow and social networks.

World Map also serves as a platform for Mostak’s first visualization project, TweetMap, which allows users to look at Twitter heat maps from 125 million tweets sent in a three week span in December 2012.

It’s open for anyone to use and explore. The project is still in alpha, but users can enter terms and see where and when the highest density of people included in those terms in tweets. “Hockey” lights up the northern United States, Canada and Sweden; “hoagie” shows the term for a sandwich is almost exclusively used in New Jersey and Pennsylvania.

The heat map is a good example of the system’s horsepower. The visualization reads each 125 million geocoded and time-stamped tweet and relates it to those sent nearby, gauging which areas used the terms most often and displaying it on World Map in milliseconds.

**Open Source Plans**

Mostak said he struggled with how to commercialize his idea. He filed the paperwork for a provisional patent, he said, but said he’s “99 percent sure” he’s going to take MapD open source instead. He plans to keep certain parallel processing algorithms he’s written for the system proprietary, but the base of the data processing system and the computation modules will be open to everyone.

This division between open source and proprietary technologies is Mostak’s way of keeping the full-time pressures of running a for-profit entity at a distance from his research interests. “The business side was just stressing me out so much,” Mostak said. “I wanted to slow it
down, and I didn’t even have time to focus on my [research] work because I was so stressed with the business side of things.

“It wasn’t any fun to me,” Mostak said. “Say [seeking private funding] had a greater income potential, I still think you can do well with open source. If it’s open source I can still work here at MIT, write research papers around some of this and then finally, maybe at some point in the future I can build a company doing consulting around this. There are a lot of companies on this model, like 10gen and MongoDB.”

Mostak said he has to clean up his 35,000 lines of code before publishing them on an open source license, which will take months. In the meantime, keeping the project in the university setting and working towards an open source license will open up some public and academic funding opportunities, he said.

Once the project does go open source, then he can rely on a community to help him build out the system.

“It’s much more exciting that way,” Mostak said. “If you think of it as the idea that people could really benefit from a very fast database that can run on commodity hardware, or even on their laptops all the way up to big scalable GPU servers, I think it would be really fun for people.”

By opening the code, Mostak opens up the possibility of having a competitor rapidly develop a system and force him to the edge of the market. That possibility doesn’t bother him, he said.

“If worse comes to worst, and somebody steals the idea, or nobody likes it, then I have a million other things I want to do too, in my head,” Mostak said. “I don’t think you can be scared. Life is too short.”

“I want to build the neural nets,” he said. “I want to do the trend detection, I want to do clustering. Maybe only one or two of them are novel, and a lot of them have been done, but they’d just be very cool to do on the GPU, and really fast. If I had a few people to help me that would just be awesome.”

This version of the story has been updated to correct the power requirements for the world’s fastest super computer in the year 2000, and to include the name of Mostak’s thesis advisor, Prof. Tarek Masoud. The article has also been updated to correctly state that MongoDB is a project curated by the company 10gen.
Innovative Relational Databases Create New Analytics Opportunities for SQL Programmers

by Ian B. Murphy

A new crop of high-performing relational databases are entering the market to reach out to millions of data professionals who are fluent in SQL. These new databases use the interactive querying language, while changing how transactional data is stored, to address performance limitations of traditional relational databases.

Competing with Hadoop and NoSQL, these new databases, such as NuoDB, Splice Machine, VoltDB, Cloudera’s Impala and Hadapt, are focused on scalability and predictability, with the ability to perform with distributed data in the cloud or on-premise.

Matt Aslett, the research manager for data management and analytics at 451 Research, said the new database market represents an unprecedented level of innovation.

Several new entries into the market are reimagining the transactional database to ensure ACID compliance, which stands for atomicity, consistency, isolation and durability. Those four traits guarantee reliable database transactions.

“Maintaining the predictable performance at scale is the inherent issue. Scalability can be achieved, distributed environments can be supported, but the relational database tends to fall down in terms of that,” Aslett said. “What is so interesting and exciting about this space right now: A lot of people are thinking anew about how to solve these problems, but with foundational SQL and ACID transactions, stuff that has been proven.”

With so many developers and data professionals familiar with SQL, and business intelligence tools and other applications written for SQL, the interactive query language isn’t going anywhere. Many new database creators are looking for ways to connect SQL to highly scalable distributed file systems such as Hadoop.

Barry Morris, CEO of NuoDB, said so many professionals know SQL that it makes more sense to create scalable and predictable databases using SQL than to teach every professional new programming and query-writing skills.

“Every one of the global 2000 companies has SQL everywhere,” Morris said. “They’ve got thousands of employees they’ve trained, they’ve got tools, and they’ve got business
processes and applications. Everything is very dependent on SQL. If it’s good enough for that, don’t change it.”

NuoDB, which launched for general availability on Jan. 15, has targeted those SQL users by coming up with a new architecture for a relational, ACID-compliant database for the cloud or on-premise deployments. The database takes advantage of in-memory technology to avoid data caching, and it keeps the data close to the application layer to boost performance. The transaction layer and the storage engine are separated, so while the data is responding to interactive SQL queries, transactions are recorded to any number of permanent storage nodes, either in the cloud or on-premise.

Traditional relational databases are like a library, where each patron must check out books at the front desk so a proper record can be kept of each transaction. According to Morris, NuoDB’s architecture is more like a sports contest, where different data key values are the players interacting on the field while permanent storage nodes are reporters in the press box keeping track of who scores, when and how.

Morris said the company calls this database architecture concept “emergence,” where each bit of data, called atoms, reacts together like a flock of birds moving together in flight. NuoDB holds several patents for its new emergent technology; Morris said the company built the new database from scratch with 12 rules for the future of cloud data management in mind.

“Let’s think about the Web 10 years out, when there are 50 billion devices and we’ve got billions more users, most of them on mobile,” he said. “When we’ve got free bandwidth, when your motor car and your television set are on the web, and there are going to be millions more applications. On the back end of that, there are going to be databases. Identify what the requirements of these systems are, and then let’s compare that to what people are actually building, and we think that we’ve cracked it.”

Splice Machine also uses SQL to connect to a distributed file system, using Hadoop and HBase for storage for availability, scalability and fault tolerance, according to Monte Zweben, the company’s CEO.

The Splice Machine database is transactional and by using SQL, the database gains interactivity. While that’s key for data scientists looking to do iterative exploration, it’s even more important for new big data applications, he said.

“We believe there is a new class of application that is emerging that is generating orders of magnitude more data than the previous generations,” Zweben said. “Your response time needs to be very quick in an application setting, and more importantly it needs to be transactionally safe.”

Zweben pointed to an e-commerce shopping cart, or a record of medical treatments, as an example of applications where data transactions have to be faithfully recorded.
Zweben said NoSQL databases that sought to fix performance issues without including SQL have “thrown the baby out with the bath water.”

“The entire IT community has dedicated many, many years of application development and data analytics and business intelligence utilizing SQL,” he said. “There are many tools out there for it, and there are great deal of skillsets and organizations out there that are well trained.”

There are several other projects looking to connect SQL to distributed systems. Top Hadoop distribution Cloudera announced its Impala project, which uses SQL to query Hadoop in October. Hadapt also allows SQL queries on data stored on Hadoop.

The open source Apache project Drill is working on interactive queries for Hadoop.

VoltDB, like NuoDB, is using an in-memory approach to boost performance for online transaction processing, or OLTP. VoltDB announced its version 3.0 on Jan. 22, with increased SQL and JavaScript Object Notation (JSON) support.
Advice for Evaluating Alternatives to Your Relational Database

by Martin LaMonica

Choosing an alternative to the traditional relational database can almost feel like ending a long relationship or choosing to root for the Yankees after growing up as a Red Sox fan. But businesses would do well to explore emerging database technologies, both for specific projects and with an eye towards the future.

The combination of Web-scale computing, powerful commodity hardware, and big data has ushered in an array of new products that are shaking up the database and analytics field. The result is far more choice, but also a potentially confusing environment for making technology decisions.

NoSQL, or “not only SQL,” databases have become viable alternatives to relational database, particularly for applications that store unstructured or semi-structured data. There is also a crop of new relational databases, sometimes called NewSQL databases, designed to run on many cheap servers, rather than on one single appliance or a very large server.

Right now, it’s mostly the leading-edge technology adopters who have jumped on the Hadoop and NoSQL wave, particularly for Web-facing applications with high volumes of data. eBay, for example, has been running the open-source distributed database system Cassandra across dozens of nodes to enhance online shopping with more customized data for its users.

Implications of New Database Technologies for Analytics

But more mainstream IT organizations should have these emerging technologies on their radar screens because they represent a potentially major shift in analytics, analysts say. Instead of the relational database undergirding most enterprise applications, computing systems are becoming more mixed with multiple specialized data stores emerging.

“In the last year, we’ve seen a big shift toward an acceptance to look at alternatives to relational databases,” said Matthew Aslett, analyst at consultancy 451 Research. “For new applications and new projects, particularly Web-facing ones that are perhaps not mission critical, organizations are looking at them as options for their next-generation database platforms.”
Relational databases — think of MySQL, Oracle, IBM DB2, Microsoft SQL Server, Teradata and Sybase — are mature, have a rich ecosystem of third-party products and vendor support, and all speak SQL so training is typically not an issue. So why bother to look around?

One of the primary drivers is economic. Many of the newer products are open source and designed to run on clusters of commodity hardware. This means administrators can add more servers or storage devices to scale up, rather than buy a more expensive high-end server. Hadoop, after all, was specifically designed by Internet giants, led by work at Yahoo and research from Google, to run Web searches in data centers filled with racks of commodity servers.

But a powerful draw toward NoSQL databases is the flexibility they can bring over the relational model of storing data in rows and a predetermined number of columns.

Craigslist is a MySQL shop but it decided to pull the plug on MySQL for its archive database because of the trouble engineers had making changes to postings, according to engineer Jeremy Zawodny, who worked extensively with MySQL as an engineer at Yahoo.

Altering a table to, for example, change the number of photos in a posting was difficult and time consuming. The company decided to go with the open-source document database MongoDB because it offered more flexibility to make changes. The shift eases administration and frees people to work on improvements to the system, such as adding real-time analytics, Zawodny said in a video done by MongoDB.org.

“There are parts of the other systems we have that may benefit from adopting more of a document model than a relational model. We even look at our main database and some of us squint at that and say, ‘Why is even that relational?’” he said. “It kind of opens up your mind to say, ‘What else can we improve by doing this?’

### How to Choose a Relational Database Alternative

- **Consider the business problem first.** Match the technology to the business problem at hand, rather than survey the entire confusing array of options.
- **Focus on the data model.** NoSQL databases are best for applications that store unstructured data or require a flexible data model.
- **Look at cloud-based services.** These can ease integration issues but won’t work when data security and privacy concerns are paramount.
- **Choose products with a reasonable learning curve.** Look for products that are professionally supported and work with SQL or a similar language.
- **Ask whether your application requires a highly scalable cluster.** Many new products are designed for scale-out architectures, but that does add management overhead.
Don’t Unplug That Relational Database

Growing interest in alternative databases does not necessarily mean unplugging relational databases. Instead, Hadoop and NoSQL databases will often perform a specific task, perhaps one that was wasn’t done before, in concert with existing systems. Companies could start to collect network or Web log data, for example, to improve the performance of their ecommerce application or analyze unstructured social media data.

“Many of these systems coexist where Hadoop acts as a landing strip for lower density data and then the data is put into a structured database or queried directly using something like HiveQL. Some typical use cases are around customer churn, fraud, product optimization, and capacity planning, but we’re seeing more each day,” said Tony Cosentino, an analyst at Ventana Research.

The right approach is to understand the business problem that needs solving and then look at which technology maps best, rather than surveying the bewildering array of options, analysts say.

Other important considerations are product maturity and the skills required to implement and manage new technologies. There are a number of companies formed to offer support and services around open-source products and a number of new products support SQL or a SQL-like language for queries and analytics.

For the most part, tying Hadoop and NoSQL databases into existing computing systems is not a major problem because there are connectors from Hadoop into all existing databases and support from established vendors, said Aslett. But more sophisticated applications, such as one that requires real-time data movement, will require custom work, he said.

The decision to go with a NoSQL database should be driven primarily by the data model and whether you will benefit from the flexibility to make changes over time, said Max Schireson, the president of 10Gen, which provides support services for MongoDB. Using a document database to store purchase orders, for example, makes it easier to later alter the required fields, or add a Twitter handle to documents that contain contact information. Learning a new product that fits the business problem is better than using a relational database for a task it’s not designed to do, Schireson argued.

Using a database, whether it’s relational or not, that is built to run on large clusters means IT organizations can support petabyte-size data sets. That means businesses can take on big data applications on a cheaper and more flexible hardware platform.

Moving to a different technology architecture inherently carries risks, such as lower-than-hoped performance or higher costs than anticipated, but the products and services are developing quickly. Because these technologies are maturing, more companies are willing to try alternative technologies for some applications, said 451 Research’s Aslett.

“There are a number of things coming together that mean people can really do this at a professional level rather than hack something together themselves,” he said.

Martin LaMonica is a technology journalist in the Boston area. Follow him on Twitter @mlamonica.
How an In-Memory Database Management System Leads to Business Value

by Puneet Suppal

Data proliferates at a much faster pace today than it did just a few years ago. Add to that the impact of social media and we now have data proliferation that is also rich in variety. Intense competitive pressures demand that businesses become more agile than ever before — and this translates into the need for being able to adapt business models and business processes much faster. This means that decisions must be based on rich granular information, covering a wide variety of sources, and often in real-time.

In my conversations with Fortune 500 CXOs and others, it is clear that they see these trends strengthening in the future. In fact, they want to be able to predict outcomes better using real-time data in order to buy them the agility to stay ahead of the competition and stay relevant for the customer. Thus, there is a growing need for possessing the ability to always select the most appropriate action dynamically — often in real-time — to address the business question. This is the central issue that enterprises would like to have addressed.

Let us take a look at the case of Bigpoint, a retailer of online gaming. They have demonstrated how their use of a real-time data platform allows them to process more than 5,000 events per second and make targeted offers to gamers (their customers) based on historical and real-time data. In order to deliver these personalized offers to individually targeted gamers while they are online, the solution leverages a real-time predictive modeling system in addition to comprehensive in-memory processing. Bigpoint now projects a significant increase in revenue by applying this solution to its business model. This is an example of how businesses are increasingly looking at getting between the end-customer and the cash register in dynamic ways.

Moving to a More Digital Enterprise

“Most C-level executives say the three key trends in digital business — namely, big data and analytics, digital marketing and social-media tools, and the use of new delivery platforms such as cloud computing and mobility — are strategic priorities at their companies,” according to a recent McKinsey Quarterly article.
The challenge before enterprises is to take advantage of these trends—any organization that succeeds in this will have moved closer to being a more digital enterprise. This is where a true in-memory data platform can make a difference. Ideally, it should be a platform that enables the organization to go deep within their data sets to ask complex and interactive questions, and at the same time be able to work with enormous data sets that are of different types and from different sources. Such a platform should be able to work with recent data, without any data preparation such as pre-aggregation or tuning, and preferably in real-time.

This is not a trivial undertaking. Many database management systems are good at transactional workloads, or analytical workloads, but not both. When transactional DBMS products are used for analytical workloads, they require you to separate your workloads into different databases (OLAP and OLTP), and expend significant effort in creating and maintaining tuning structures such as aggregates and indexes to provide even moderate performance.

A system that processes transactional and analytical workloads fully in-memory can transcend this problem. There are differences among in-memory systems, and an important consideration is whether a system requires a business to prepare the data to be processed — which can take a lot of work — or not. There are software vendors today that claim to do some in-memory processing, but deliver on this front only in a limited way. Some of them deliver on speed by finding ways to pre-fabricate the data and speed up the data-crunching — this often runs the risk of missing some key element that might become necessary to decision-makers, while also killing any chance of working with real-time data. One CTO I met recently put it succinctly: There should be a resident ability to report live against transactions as they happen, such that cost-saving or revenue enhancing steps can be taken in real time.

In my conversations with various customers, it is clear that they can't wait for the day when they can run their entire landscapes on this new type of data platform as opposed to traditional database systems. Such a real-time platform has the potential to bring dramatic process benefits to an enterprise.

Puneet Suppal is a member of SAP’s Database and Technology Platform Adoption team focused on the SAP HANA in-memory computing platform. Follow on Twitter @puneetsuppal and connect to him at LinkedIn. Bigpoint is an SAP HANA® customer.
How the Financial Services Industry Uses Time-Series Data

Kx Systems’ Database Drives Real-Time Decision Making in the Financial Industry

By Scott Etkin

In the fast-paced, data-intensive world of financial services, the difference between now and a few seconds from now can be millions of dollars. In this industry, the ability to discover insights within massive inventories of continuously arriving data is crucial.

Kx Systems’ kdb+ is a column store time-series database system with a built-in programming language called q. Brokerages, banks, hedge funds, and other financial institutions have been using kdb+ in their electronic trading businesses for the past two decades.

At some of these companies, trading systems have grown to hundreds of terabytes of data. These systems rely on kdb+ for a stable and scalable foundation on which to build analytics that process real-time, streaming, and historical market data for algorithmic trading strategies.

Simon Garland, Chief Strategist at Kx Systems, fielded questions about kdb+ and how financial services and other industries are leveraging the platform for real-time insights.

Data Informed: What types of data does kdb+ typically handle, and how is the data handled?

Simon Garland: The type of data that kdb+ works with is structured data, as well as semi-structured data. In the financial services industry, this is often in the form of market data, which comes from exchanges like the NYSE, dark pools, and other trading platforms. This data may consist of many billions of records of trades and quotes of securities with up to nanobyte precision — which can translate into many terabytes of data per day.

The data comes in through feed-handlers as streaming data. It is stored in-memory throughout the day and is appended to the on-disk historical database at the day’s end. Algorithmic trading decisions are made on a millisecond basis using this data. The associated risks are evaluated in real-time based on analytics that draw on intraday data that resides in-memory and historical data that resides on disk.
Why don’t financial services companies use typical RDBMS?

Garland: Traditional databases cannot perform at these levels. Column-store databases are generally recognized to be orders of magnitude faster than regular RDBMS, and a time-series optimized columnar database is uniquely suited for delivering the performance and flexibility required by Wall Street.

What benefits do fast database processing speeds bring to users?

Garland: Orders-of-magnitude improvements in performance will open up new possibilities for “what-if” style analytics and visualization, speeding up their pace of innovation, their awareness of real-time risks, and their responsiveness to their customers.

The Internet of Things in particular is important to businesses that now can capitalize on the digitized time-series data they collect, like from smart meters and smart grids. In fact, I believe that this is only the beginning of the data volumes we will have to be handling in the years to come. We will be able to combine this information with valuable data that businesses have been collecting for decades.

How does the q programming language compare to SQL?

Garland: The q programming language is built into the database system kdb+. It is an array programming language that inherently supports the concepts of vectors and column-store databases rather than the rows and records that traditional SQL supports.

The main difference is that traditional SQL doesn’t have a concept of order built in, whereas the q programming language does. Unlike traditional SQL, the q language contains a concept of order.

This makes complete sense when dealing with time-series data.

The q language is intuitive and the syntax is extremely concise, which leads to more productivity, less maintenance, and quicker turn-around time.

What other industries are using kdb+, and why?

Garland: Utility applications are using kdb+ for millisecond queries of tables with hundreds of billions of data points captured from millions of smart meters. Analytics on this data can be used for balancing power generation, managing blackouts, and for billing and maintenance.

Internet companies with massive amounts of traffic are using kdb+ to analyze Googlebot behavior to learn how to modify web pages to improve their ranking. They tell us that traditional databases simply won’t work when they have 100 million pages receiving hundreds of millions of hits per day.
In industries like pharmaceuticals, where decision-making is based on data that can be one day, one week, or one month old, our customers and prospects say our column store database makes their legacy data warehouse software obsolete. It is many times faster on the same queries. The time needed for complex analyses on extremely large tables literally has been reduced from hours to seconds.

What are the key similarities among the industries using kdb+? What are some differences?

**Garland:** The shared feature is that all of our customers have structured, time-series data. The scale of their data problems is completely different, as are their business use cases. The financial services industry, where kdb+ is an industry standard, demands constant improvements to real-time analytics.

Other industries, like pharma, telecom, oil and gas, and utilities, have a different concept of time. They also often are working with smaller data extracts, which they often still consider to be “big data.” When data comes in one day, one week, or one month after an event occurred, there is not the same sense of real-time decision making as in finance. Having faster results for complex analytics helps all industries innovate and become more responsive to their customers.
Find other articles like these and more at Data Informed:

www.data-informed.com

Data Informed gives decision makers perspective on how they can apply big data concepts and technologies to their business needs. With original insight, ideas, and advice, we also explain the potential risks and benefits of introducing new data technology into existing data systems.

Follow us on Twitter, @data_informed