BEYOND RELATIONAL

MARKLOGIC WHITE PAPER • SEPTEMBER 2015

Organizations face a growing inability to handle the massive volumes of disparate, varied, and changing data with the relational databases that have been relied on for the past three decades. For this reason, leading organizations are going beyond relational to embrace a new generation database: MarkLogic.

CURRENT STATE ASSESSMENT

Is your organization in need of change, or do you think you're doing all right with your data? The answers to the questions below provide a baseline assessment to help answer that questions. The more "YES" answers, the more likely your current database(s) are not meeting your organization's needs.

		YES	NO
BUSINESS QUESTIONS	1. Is there data that is important to your organization that is not in a database?	✓	✗
	2. Are there multiple databases with essentially the same data in them?	✓	✗
	3. Are there numerous data sources that are not centrally managed by the IT department?	✓	✗
	4. Are there large IT projects that have been behind budget or failed to launch?	✓	✗
	5. Are there database schemas so complicated that only a small handful of experts can adequately answer questions about them?	✓	✗
TECHNICAL QUESTIONS	6. Does data modeling ever slow down or hinder the process of application development?	✓	✗
	7. Are there relational tables in which column names have changed or been assigned new meaning "just to make it work"?	✓	✗
	8. Are there frequent database schema changes each month, and are some of the changes unsuccessful?	✓	✗
	9. Are significant time and resources spent figuring out how to scale?	✓	✗
	10. Are there ever performance problems or bugs that may have resulted from complicated middleware?	✓	✗

Contents

INTRODUCTION

In the past few decades, technology has evolved rapidly and every aspect of doing business has changed. Today, more data is collected than ever before and organizations have great ideas for bigger, smarter applications. One would think that databases—where organizations store their most prized asset, *data*—would have also changed. But to a large degree, they haven't at all.

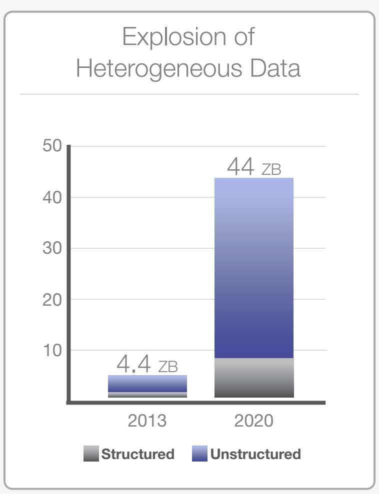
The dominant technology for storing and managing data—the relational database—still looks pretty much the same as it did when first released during the Cold War, over thirty years ago. Back then, data was perceived as small, neat, structured, and static because that was the only way it could be stored. However, data is not like that. In today's world, organizations are confronting the reality that data is big, fast, varied, and changing. Organizations are no longer just managing a small handful of systems, but hundreds of systems and petabytes of data.

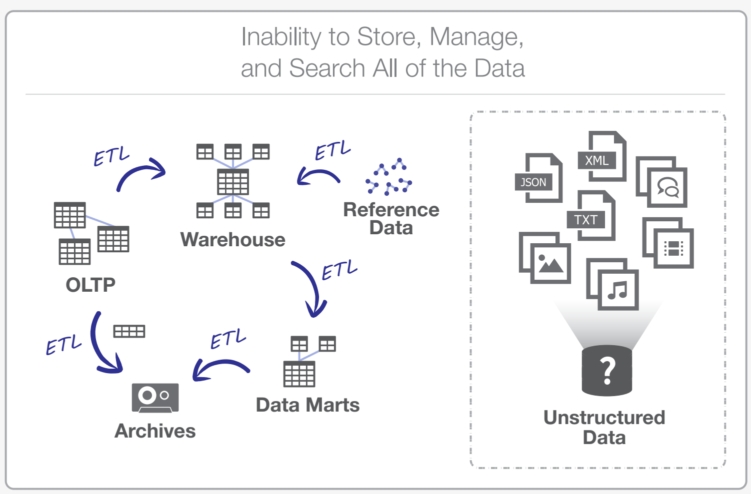
The new world of "Big Data" creates an exciting opportunity, but too often it is just seen as yet another challenge that IT departments must handle. Today, IT departments spend most of their time just keeping their heads above water, managing a complex web of data silos and frequent ETL (Extract, Transform, Load) processes to shuffle data around. Then, as individuals and departments seek their own solutions, "shadow IT" and security lapses start to appear. Across industries, high costs and lengthy project timelines are the norm. Stuck in a state of constant maintenance cycles, organizations are unable to focus on getting the most value from all of their data. To some degree, these challenges have all come about as a result of trying to use relational databases to solve problems they were never designed to solve.

Today, organizations cannot rely on just using the one-size-fits-all relational model. Motivated by the need to change, organizations are embracing new kinds of databases. MarkLogic is at the forefront of this generational shift, providing a database that is a better fit for all of today's data. MarkLogic has a more flexible data model for storing, managing, and searching massive volumes of varied data, while also maintaining all of the enterprise capabilities that organizations require. It is this unique combination that has enabled leading organizations to go beyond relational and get more value from more data than ever before.

WHY CHANGE?

Organizations face a growing inability to handle their disparate, varied, and changing data. Only by adopting new approaches can organizations address this problem and reduce their risk, build smarter applications faster, and get more business value from their data.





> “Taken as a whole, the ‘Vs’ of big data can be summed up in one truth: *today’s data is big, fast, complex, and changing*.”

TODAY’S WORLD OF BIG DATA

Limited by the technologies of the time, data used to all look the same. It came into data centers slowly and orderly, and it was neatly organized as tabular data that fit into rows and columns joined together across pre-configured tables. The pace of change was idle, both for the business and for IT, and that was okay. But that was the 1980s, and today is much different.

The world of Big Data that we operate in today is well-characterized by the oft-mentioned three “Vs”—volume, velocity, and variety. Additionally, two more “Vs” are increasingly relevant—veracity and variability. Taken as a whole, all of these “Vs” can be summed up in one truth: *today’s data is big, fast, complex, and changing*.

VOLUME

The digital universe is growing 40 percent a year, and is expected to grow from 4.4 zettabytes in 2013 to 44 zettabytes in 2020 (a zettabyte is 1 trillion gigabytes).[1] Paper files are no longer the system of record, databases are—and that means storing *everything*. Organizations today must now handle more data in more forms across a greater number of systems, and are expected to manage it efficiently, securely, and with low overhead. The cost of data storage continues to drop, and consumers and regulators expect that organizations can and should store everything.

VELOCITY

Everything is faster in today’s world. Data is created faster and data changes faster. And, the questions asked of the data also change faster to meet new business requirements laid out to handle rapid changes in market dynamics, new management, on-demand services, or acquisitions and spin-offs. Today, decisions get made in minutes, not days, and data to support those decisions must be delivered the right format with reduced latency and greater efficiency. Whether delivering data for sporting events or detecting fraud at a bank, the need to get data in *real-time* is no longer on the wish-list, but is a requirement. The timeline for application development is also much faster, measured in weeks, not years. And, those applications must hold up to torrents of users—users that have a decreased tolerance for waiting, decreased loyalty, and an increased desire for personalization.

VARIETY

Variety is one of the biggest challenges of all the “Vs.” Today’s data is much more varied, or heterogeneous, than it used to be—about 20 percent is structured (e.g., transactional, tabular data) and 80 percent unstructured (e.g., documents, text, emails, images, video).[2] The new unstructured data sources available are certainly problematic. One study found that according to 64 percent of businesses, the primary reason for considering a new approach to Big Data was the diverse, new, and streaming data sources they now have to handle.[3] But the variety of structured data is perhaps even more problematic as organizations struggle to handle the many shapes, sizes, and types of structured data that are quickly growing in volume and changing. New applications, mergers and acquisitions, and repurposing of data are common reasons that lead to the disparity of structured data.

VERACITY

Veracity has to do with the truthfulness of data, or data integrity. Data is a highly prized asset and organizations take great lengths to ensure that their data is accurate and not corrupted in any way. For this reason, it is becoming more important to track the data lineage, or lifecycle of data, including when and where data originated (its provenance), its on-going history (how

1 IDC. *Digital Universe*. April 2014 <http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>

2 Khan et al. *Big Data: Survey, Technologies, Opportunities, and Challenges*. Scientific World Journal, 2014 <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4127205/#B53>

3 New Vantage Partners. *Big Data Executive Survey: Themes & Trends*. 2012 <http://newvantage.com/wp-content/uploads/2012/12/NVP-Big-Data-Survey-Themes-Trends.pdf>

> “Data siloes are reported to be the number one impediment to Big Data success.”

it changed and by whom), its retention (how long it should be kept available), and its relevance (what data provides the best answer). Additionally, organizations must have strong data governance policies in place to guard users’ access to the data at a granular level. These issues are all becoming more important for audit and compliance reasons, in addition to providing the ability to run more advanced analytics.

VARIABILITY

Variability refers to the variations in meaning that data can have depending on context. Variability has been discussed by a Principal Analyst at Forrester, Brian Hopkins, who defined it as, “the variability of meaning in natural language and how to use Big Data technology to solve them.”[4] One example is with the word “sub”—does it refer to a naval submarine or to a Subway® sandwich? This problem is about more than natural language. There are also differences in how users and data modelers describe basic entities. An example is with the state of “North Carolina” that sometimes appears as “N Carolina” or simply “NC.” How would a database know they are referring to the same thing, or what the concept of a “state” even is? People have an easy time deciphering meaningful knowledge from context, but databases have difficulty with these semantic challenges. With more data comes more variations in how people, places, and things are described, and so the problem is further amplified.

DROWNING IN A SEA OF COMPLEXITY

Today’s world of Big Data should be an enormous opportunity, but all too often it is just seen as yet another challenge. Today, IT departments spend most of their time keeping their heads above water, and if they do not tackle the complexity head on, it has the potential to sink the whole enterprise.

DATA SILOS AND ETL

Data silos are reported to be the number one impediment to Big Data success.[5] This is obvious when looking at most complex enterprise architecture diagrams that show incompatible legacy systems woven together with other legacy systems to create a complex, brittle architecture in which data is un-shareable and un-usable. In most organizations, only a few experts may still be around to understand how it maps to the intricate business rules of the organization. It is thus not surprising that for most business intelligence initiatives, the majority of time is just spent identifying and profiling data sources.[6]

Data silos were not an intentional part of the design, but the result of short-term solutions. Most databases are only designed to support a certain application or certain type of data. To get data out of those databases and use it for another purpose in another database, a process of ETL (Extract, Transform, Load) must occur to ensure it matches the schema of the new target database. ETL occurs frequently in most organizations, creating another data silo each time it is done.

As silos proliferate it becomes more and more difficult to maintain and connect them. Eventually, developers begin using “duct tape” maintenance code to connect various applications together, avoiding the true source of the problem. This only creates more complexity and eventually something either stops working, developers get too frustrated and leave, or new projects are slowed down to such an extent that progress becomes hopeless.

4 Hopkins, Brian. “Blogging From the IBM Big Data Symposium - Big Is More Than Just Big,” 2011 <http://blogs.forrester.com/brian_hopkins/11-05-13-blogging_from_the_ibm_big_data_symposium_big_is_more_than_just_big>

5 Oracle. *IT Assessment Complexity Survey*. 2015 <http://www.oracle.com/us/corporate/features/it-complexity-assessment-survey-2281110.pdf>

6 Boris Evelson. *Boost Your Business Insights By Converging Big Data And BI*. Forrester, March 25, 2015 <https://www.forrester.com/Boost+Your+Business+Insights+By+Converging+Big+Data+And+BI/fulltext/-/E-RES115633>

> Relational database vendors are still offering users a 1990s-era product, using code written in the 1980s, designed to solve the data problems of the 1970s, with an idea that came around in the 1960s.”

“SHADOW IT” AND SECURITY LAPSES

Oversight of enterprise data has continued to slip away from CIOs as employees and departments fix their own problems by using software that is not overseen or managed by a centralized IT department. Most CIOs think they have a few dozen “shadow IT” apps in use, but more often it is a few hundred. In one survey, organizations were found to be using an astounding 923 distinct cloud services, and only 9.3 percent met enterprise security requirements.[7] This change is a direct result of the perceived unresponsiveness to the needs of the business, and creates enormous risk and inefficiencies for the organization as a whole.

This is happening as the cost of a lapse in security continues to grow and cybercriminals use more sophisticated attacks. An organization’s reputation can be severely damaged with just one breach, and a data breach can also be costly. One study found that a single cybersecurity incident can cost a company $5.4 Million on average, or $188 per record.[8] Unfortunately, protecting data is harder than ever with the proliferation of data silos that create more entry points, vulnerabilities, and data leakage.

HIGH COSTS, FAILED PROJECTS, AND AN INABILITY TO INNOVATE

It is a sad expectation that many IT projects will not meet deadlines and will be over budget. High costs and failed projects are the norm, and in fact, half of IT projects with budgets of over $15 Million run 45 percent over budget, are 7 percent behind schedule, and deliver 56 percent less functionality than predicted. Even worse, about 17 percent of IT projects go so bad that they can threaten the very existence of the company.[9]

This is despite the lengthy planning, enormous resources, and large teams of brilliant individuals that work on these projects.

While teams are working on over budget projects that under deliver, they are not spending time on the innovative projects that are so critical to the organization’s success. How can an organization get value out of Big Data if they cannot devote any resources to it? Today, 95 percent of all database spending goes towards relational databases, which only store about 20 percent of enterprise data.[10] That leaves only 5 percent to spend on managing the other 80 percent of enterprise data. Without change, CIOs and IT will be left tending a host of legacy systems and responsibilities as they get outrun by competitors that devote more resources to innovation and business transformation.

WHY RELATIONAL DATABASES AREN’T WORKING

Many of the challenges with Big Data and resulting complexity seen today can be traced back to relational databases. There is nothing inherently wrong with relational databases—they were just never designed to handle today’s data. This is why the former CIO of the Federal Government, Vivek Kundra, said back in 2009 that, “this notion of thinking about data in a structured, relational database is dead.”

First invented in the late 1970s, relational databases took over from hierarchical mainframe systems to become the database of choice by the early 1990s. Relational databases met the needs of early computing very well. They made it possible to decouple applications from the data and use less custom code, and gave users more control over querying the data by using SQL as the common query language.

7 Skyhigh. *Cloud Adoption and Risk Report - Q1 2015*. June 2015 <http://info.skyhighnetworks.com/rs/skyhighnetworks/images/WP%20CARR%20Q1%202015.pdf>

8 Ponemon Institute. *2013 Cost of Data Breach Study: Global Analysis*. Symantec. 2013 <https://www4.symantec.com/mktginfo/whitepaper/053013_GL_NA_WP_Ponemon-2013-Cost-of-a-Data-Breach-Report_daiNA_cta72382.pdf>

9 McKinsey & Company. “Delivering large-scale IT projects on time, on budget, and on value,” October 2012 <http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value>

10 Carl Olofson. *Worldwide Database Management Systems 2014–2018 Forecast and 2013 Vendor Shares*. IDC, June 2014 <http://www.idc.com/getdoc.jsp?containerId=248952>

> "This notion of thinking about data in a structured, relational database is dead."

Vivek Kundra, Federal CIO, July 21, 2009; Open Government and Innovations Conference

---

Throughout the almost 40-year existence of relational databases, they have continued to improve and the ecosystems around each vendor's product have grown, but the fundamental model for managing data has remained unchanged. The fact is, relational database vendors are still offering users a 1990s-era product, using code written in the 1980s, designed to solve problems of the 1970s, with an idea that came around in the 1960s.[11] Today, organizations require more than what relational databases can offer, and the following sections discuss why.

RELATIONAL DATABASES ARE NOT DESIGNED TO HANDLE CHANGE

Relational databases organize data in tables with rows and columns, much like spreadsheets in Microsoft Excel. Each row represents a unique entry and each column describes unique attributes. One column is chosen as the primary key to uniquely identify each row in the table.

So, for example, if you modeled a relational database for customers and products they ordered, you might start by creating a "Customers" table with a column called "CustomerID" to be used as the primary key. You would create additional columns for each attribute about each customer, such as "FirstName," "LastName," and "Address," defining the type of data that will be stored in each.[12] You then link the "CustomerID" to another table, "Orders," that stores information about a customer's purchases. Each row in the "Orders" table would have its own unique identifier and also a reference to the primary key of the "Customers" table.

11 It was in 1969 that Edgar "Ted" F. Codd first published his famous paper internally to IBM. Later, in 1970, the paper was made publicly available. (E.F. Codd, "A Relational Model of Data for Large Shared Data Banks." Communications of the Association for Computing Machinery, Vol 13 No 6 Pgs. 377-387)

12 In the 1980's, relational databases limited column names to eight characters and had to be one case. So, column names would be "fname" or "lname." Now, the SQL standard is to have 30 character limits for column names.

You continue this process of creating various tables, ensuring your design meets all of the entity and referential integrity constraints, and everything is properly "normalized" so that there are no repeating columns, that columns are all dependent on their primary key, and no tables duplicate any information. Those constraints are what maintain data consistency and ensure fast queries—hallmarks of the relational model. This process of designing the data model, or schema, involves a dedicated team getting together to decide what tables should be created and what the column names will be. It is an important process, and the end result is often proudly depicted with a large entity-relationship diagram (ERD) that gets printed out and hung prominently in the hallway.

The problems with this approach are twofold. First, the process can take months, if not years, depending on the size of the database. Relational schemas are complex, and all of the modeling must be done *in advance* of loading any data or building the application. Second, if a change is required after applications are built on top of the database, it is a time and resource intensive process that can take another few months or years. The relational model is like a sensitive, complex rainforest ecosystem in which one small change can cause detrimental effects with cascading impacts across the database and through the application stack. Even a simple change like adding or replacing a column in a table might be a million dollar task.[13]

Today, change occurs frequently, and data modeling is a huge challenge because of the time and resources that relational databases require. Each year, billions of dollars are spent on data modeling and ETL processes to create and recreate more "perfect" models that will never change. But they always do.

13 According to one customer at a leading Fortune 100 technology company, the task of adding a column could take them up to a year and cost over a million dollars. For other more complex data modeling projects involving master data management, even lengthier timelines of over five years have been reported.

> “It is surprising that 95 percent of total database spend is on relational databases but relational databases are only designed to handle the 20 percent of data that is structured.”

RELATIONAL DATABASES ARE NOT DESIGNED FOR HETEROGENEOUS DATA

It is surprising that 95 percent of total database spending is on relational databases but relational databases are only designed to handle the 20 percent of data that is structured.[14] Organizations face a growing inability to handle that structured data, while the other 80 percent, the unstructured data, becomes completely orphaned despite having enormous value that is locked up inside it.

Organizations used to only store some key transactional data and a few basic things about their customers. Today, however, organizations can no longer cherry-pick a few key pieces of data. They need to store just about everything. As the cost of getting the infrastructure to do that has become reasonable, organizations can take advantage of the opportunity to reduce risk and lower costs. There is also an expectation from customers, partners, and regulators that the organization should store everything in a usable format that will benefit them as well.

The growing amount of structured data is a problem for relational databases because the structure of each data source is different. The changes that are required to handle a new data source, as already noted, are cumbersome and result in more schema complexity. This is true even when new data represents the same domain or concepts.

The growing amount of unstructured data also presents a problem for relational databases. The rows and columns of a relational database are ideal for storing sets of values, but most information is composed of much more than that. Consider something like a person’s medical record. It includes **values** (name, date of birth), **relationships** (to family members or care providers, to symptoms and medications), **geospatial data** (addresses), **metadata** (provenance, security attributes), **images** (CAT scan), and **free text** (doctors’ notes, transcripts).

Now, imagine putting all of that data into a Microsoft Excel spreadsheet. This task would require a lot of ingenuity, and many difficult choices: *Should large blocks of text be broken up or stuffed into a cell in the table? What about storing new data sources that come in later? How many columns should there be for metadata? What about the relationships between various entities? What about the structure within the document? What indexes should be created? What if I want to filter the data by an element that is not defined by a row or column?*

Regardless of the amount of labor and compromise that has been put into trying to make the relational model work for everything, the fact remains that it was not designed for heterogeneous data.

RELATIONAL DATABASES ARE NOT DESIGNED FOR SCALABILITY AND ELASTICITY

Today, organizations have millions of users and petabytes of data. They run their applications in the cloud to deliver dynamic content to millions of desktop, tablet, and mobile devices across various geographical locations. To handle this new reality, organizations need scalability (adding capacity for more data and more users) and elasticity (the *ease* in which the system scales, typically referring to the ability to scale back down when user demand dissipates).

Unfortunately, scaling relational databases is challenging. Relational databases are designed to run on a single server in order to maintain the integrity of the table mappings and avoid the problems of distributed computing. With this design, if a system needs to scale, customers must buy bigger, more complex, and more expensive proprietary hardware with more processing power, memory, and storage. Upgrades are also a challenge, as the organization must go through a lengthy acquisition process, and then often take the system offline to actually make the change. This is all happening while the number of users continues to increase, causing more and more strain and increased risk on the under-provisioned resources.

14 IDC, June 2014

> “The problem is that the relational model forces complexity upon IT departments because it was not designed to deliver information to different sets of users in the right way at the right time.”

To handle these concerns, relational database vendors have come out with a whole assortment of improvements. Today, the evolution of relational databases allows them to use more complex architectures, relying on a “master-slave” model in which the “slaves” are additional servers that can handle replicated data or data that is “sharded” (divided and distributed among multiple servers, or hosts) to ease the workload on the master server. Other enhancements such as shared storage, in-memory processing, better use of replicas, distributed caching, and other new relational architectures have certainly made relational databases more scalable. Under the covers, however, it is not hard to find a single system and a single point-of-failure.[15]

The enhancements to relational databases also come with high costs and big trade-offs. For example, when data is distributed across a relational database it is typically based on pre-defined queries in order to maintain performance. In other words, flexibility is sacrificed for performance. Additionally, relational databases are not designed to scale back down—they are highly *in*elastic. Once data has been distributed and additional space allocated, it is almost impossible to “undistribute” that data.

RELATIONAL DATABASES ARE NOT DESIGNED FOR MIXED WORKLOADS

“Mixed workloads” refers to the ability to handle both operational and analytical workloads. Operational workloads encompass the day-to-day business transactions that are occurring in real-time, such as purchases being made by large numbers of customers. Analytical workloads are those operations intended for business intelligence and data mining, such as when an analyst wants to look at an aggregate of purchases over a specified time period.

15 For example, Oracle RAC is a “clustered” relational database that uses a cluster-aware file system, but there is still a shared disk subsystem underneath.

In the mid-1990s a split arose between databases optimized for operational workloads, known as OLTP systems (online transaction processing), and databases optimized for analytical workloads, known as OLAP systems (online analytical processing). In OLTP systems, the data is modeled to be optimal for the application built on it, requiring consistent, speedy transactions. In OLAP systems, the data is modeled to be optimal for slicing and dicing, including aggregates and trends. Before long, elegant models were developed as experts agreed and disagreed on the best ways to model data for different scenarios. This is when “star schemas,” “snowflake schemas,” and “OLAP hypercubes” entered the lingo of data modelers.

Unfortunately, the split between operational and analytical systems contributed to the creation of disparate data marts, data warehouses, reference data stores, and archives that have proliferated out of necessity. Data from operational systems was moved via ETL to a central data warehouse designed to be the warehouse for all business decisions. However, that broke down when it could not answer new and different questions that appeared. So, another ETL process was used to move a certain subset of data to a data mart. Other systems were set up to capture reference data. Then an archive system captured all the historical data from all of the systems. Each time a new question needed to be asked or a new application built, a newer, better model was created—and no model was ever the same. What was just a simple schema and handful of databases soon multiplied to hundreds.

This is one of the reasons why most IT departments today spend the majority of their time and money just maintaining the myriad of systems in the organization. The problem is that the relational model forces complexity upon IT departments because it was not designed to deliver information to different sets of users in the right way at the right time.

> The result of the traditional relational architecture is performance loss and more opportunities for buggy code."

RELATIONAL DATABASES ARE A MISMATCH FOR MODERN APPLICATION DEVELOPMENT

Modern applications are built using object-oriented programming languages such as Java, JavaScript, Python, and C# to name a few. These languages treat data structures as "objects" that contain data and code (i.e. attributes and methods). The problem is that this way of handling data is very different from how relational databases handle data, creating an impedance mismatch between the database and application programming.

To get around the impedance mismatch, developers use a technique called object-relational mapping (ORM), a bi-directional active-active mapping between the objects in the application layer and the data as it is represented in the relational database schema. With ORM, application developers get to work with business rules and logic and generate views of the data in a way that makes the most sense from an application development perspective. With this approach, databases are viewed more simply as the places where data is persisted and where stored procedures are kept. A wide number of ORM tools are available, helping simplify application development with relational databases. Some examples of ORM tools include Hibernate for Java, ActiveRecord for Ruby on Rails, Doctrine for PHP, and SQLAlchemy for Python.

Unfortunately, ORM is also seen as a poor workaround for a systemic problem with relational databases. ORM has even been called the "Vietnam of computer science" because it "represents a quagmire which starts well, gets more complicated as time passes, and before long entraps its users in a commitment that has no clear demarcation point, no clear win conditions, and no clear exit strategy."[16] Many other publications have continued to show how ORM does more harm than good.[17]

16 Ted Neward. Blog Post on "The Blog Ride," June 26, 2006 <http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx>

17 See *OrmHate* by Martin Fowler <http://martinfowler.com/bliki/OrmHate.html>, *Object-Relational Mapping Is the Vietnam of Computer Science* by Jeff Atwood, *ORM Is an Anti-Pattern* by Laurie Voss <http://seldo.com/weblog/2011/08/11/orm_is_an_antipattern>, *ORM is An Offensive Anti-Pattern* by Yegor Bugayenko <http://www.yegor256.com/2014/12/01/orm-offensive-anti-pattern.html>, and many others.

ORM, rather than preserving the interesting aspects of the data inside an object, instead extracts the data away, tearing apart the data and adding more overhead in the process. And this happens after the data was already split up across tables through the process of normalization to begin with. Going back to the example of a person's medical record, just consider all of the various data that is part of the record that must be split across tables in a relational database. After "shredding" the data across tables, the data must then be reassembled to display or aggregate the data in the application layer in order to be presented to the user. This imposes lots of overhead, lots of mapping, and a custom framework or lots of joins in order to get materialized views of the business entity (i.e. a form or piece of paper).

The result of the traditional relational architecture is performance loss and more opportunities for buggy code. In today's fast-paced application development cycles, with users demanding more interactivity and responsiveness, the relational model shows its flaws. Rather than having to find workarounds for the mismatched relational model, developers are embracing new models that involve less abstraction and show higher performance.

A NEW GENERATION DATABASE FOR TODAY'S DATA

MarkLogic is a NoSQL ("Not only SQL") database that is at the forefront of a shift away from the one-size-fits-all relational databases of the past three decades. There are many features of MarkLogic that make it a good fit for today's data, but there are four things MarkLogic has that make it really unique:

1. A **flexible data model** for storing today's varied, changing, and disparate data
2. **Built-in search and query** to get more value out of data at any point-in-time
3. **Scalability and elasticity** to handle the massive, changing volumes of data
4. **Enterprise features** required to run mission-critical enterprise applications

> "The first MarkLogic project took 60 days. It was estimated to take 3,000 days with existing technology."

Paolo Pelizzoli, Global Head of Architecture, Global Technology Operations at Broadridge Financial Solutions

OVERVIEW OF MARKLOGIC'S DIFFERENTIATING FEATURES

FLEXIBLE DATA MODEL

MarkLogic is a multi-model database designed to natively store and rapidly query JSON, XML, and RDF triples, geospatial data, and large binaries (e.g., images, video). This capability makes MarkLogic much more adept at handling a wider variety of data types and data structures than relational databases, and makes it easier to handle changes to the data model as the data changes.

JSON and XML are both document formats, and are the primary way that MarkLogic stores data. Contrasted with the tables in a relational database, documents are much more human-readable, and provide a more natural approach to modeling the rich, variable, and complex data that today's organizations work with. By using a richer data model, organizations can get more value out of their data.

Documents are also a better match for modern application development because they avoid the impedance mismatch problem. The document model allows developers to maintain the integrity of data throughout every tier of an application stack. For example, developers can have JSON in the database, in the application layer, and the user interface. This is a more agile approach, and is a good fit for the growing amount of JavaScript used in developing modern web applications.

Another advantage of MarkLogic's data model is that no schema, or structure, needs to be defined in advance of loading data to MarkLogic. The term for this is "schema-agnostic." MarkLogic allows users to store documents with different schemas and change specific schemas without disrupting others. This powerful capability also allows users to rapidly combine relational data tables with different models—all inside MarkLogic.

MarkLogic also has graph database capabilities because it natively stores RDF triples, the language of semantics. At a high level, semantics is a data model for linking together two entities (people, places, or things) based on the relationship between them to form a triple. When linked together, triples form a graph that is machine readable, and can be used to infer new facts. MarkLogic can store hundreds of billions of triples right alongside, or even inside, JSON or XML documents.

MarkLogic is the only enterprise database that combines a document store and triple store. This unique capability makes it faster and easier to model data in the format that makes the most sense, and enhances the value that organizations can get from

COMPARING NOSQL DATABASES

As a result of the pressing need for change there has been an explosion of new data management technologies in the past few years, all aimed at providing better options to handle today's data. There are many open source NoSQL databases of various types, including document stores, graph stores, column stores, and key-value stores. There are some general similarities between them such as the ability to scale on commodity hardware, but each type of database is actually quite different. If you are looking to better understand the NoSQL landscape, download the eBook, *Enterprise NoSQL for Dummies*, available for free at info.marklogic.com/nosql-for-dummies.html.

> MarkLogic is designed to handle the volume, variety, and velocity of Big Data like other NoSQL solutions, *AND* has the enterprise features that made last-generation relational databases so reliable."

their data. As a whole, MarkLogic's data model is much more flexible than the relational model. It provides a platform to build smarter applications faster, and more agility to handle changes as they occur.

The improvements over relational databases can be dramatic: "The first MarkLogic project took 60 days. It was estimated to take 3,000 days with existing technology," said Paolo Pelizzoli, Global Head of Architecture, Global Technology Operations at Broadridge Financial Solutions.

BUILT-IN SEARCH AND QUERY

To search data fast and accurately, a database requires indexes. Database indexes are similar to those in the back of books, providing a listing of information within the book that can be quickly referenced rather than scanning the entire volume. With most databases, indexing is typically seen as a secondary task to storing data. Indexing is a difficult process in which users must figure out which indexes need to be created to answer which questions, what the performance implications of each index are, and how the indexes will be maintained. Then, to get full-text search, full-text indexes are required. With relational databases, this means adding additional software that must be setup and maintained alongside the database.

MarkLogic works differently, having a best-in-class indexing capability and full-text search built-in as part of the product. MarkLogic indexes the content and structure of data as it is loaded, and has numerous indexes (e.g., range indexes, triple index, geospatial index) that can be toggled on and off. MarkLogic's indexes make it easy to answer both simple and sophisticated queries using a variety of query languages—JavaScript, XQuery, SPARQL (the query language for semantics), and of course, SQL. An example of a sophisticated query is, "Find all earnings and rankings of professional athletes who Michael Jordan played with during his career. Restrict the results to those athletes who live in New York and were mentioned in reliable news sources after January 2015. Rank the candidate results by relevance."

Answering such multi-dimensional questions is not trivial, and it would either be extremely difficult or impossible to achieve with a relational database. A relational database would have difficulty modeling the relationships between Michael Jordan and the athletes he played with, and finding the specific mentions within news sources, which would be text documents. A relational database would also not be able to do relevance ranking in a way that a search engine such as Google does relevance—it would just return a list of results based on a simple ordering of values.

On the other hand, this kind of sophisticated query is one that MarkLogic can handle with relatively little code. The issues of relevance and full-text are resolved by MarkLogic's rich data model and powerful indexes that are designed to answer the same types of questions anyone would ask with SQL in a relational database, and much more. Even if the questions change, that is okay. MarkLogic is prepared to handle new and unexpected queries that come along, without requiring users to reconfigure the data and indexes as in a relational database. And, MarkLogic returns the results with sub-second response times over hundreds of terabytes of data, all within a system in which the data is consistently and reliably maintained.

SCALABILITY AND ELASTICITY

Rather than be constrained by the limits of single server architectures, MarkLogic is designed for massive scale on distributed systems. MarkLogic scales "horizontally," meaning that it runs on multiple servers that work together, each sharing part of the load. Using this approach, MarkLogic can operate across hundreds of servers, petabytes of data, and billions of documents—and still manage to process tens of thousands of transactions per second. And it can do all of this on

> MarkLogic is *proven* in mission-critical systems at the U.S. Department of Defense, large investment banks, healthcare payers, global media organizations, and many other industries in which success is not optional."

inexpensive commodity hardware operating in any environment, whether it is in on hardware that is sitting on-premise or in a cloud environment such as Amazon Web Services.

Massive scale is impressive, but what is perhaps even more important is MarkLogic's *elasticity*. MarkLogic's unique architecture makes it possible to quickly and easily add or remove nodes in a cluster so that the database stays in line with performance needs without costly over-provisioning. There is not any complex sharding of data or architectural workarounds—data is automatically rebalanced across a cluster when nodes are added or removed. This is one of the reasons that MarkLogic is so easy to use when it comes to administration.

ENTERPRISE FEATURES

There are common misconceptions that NoSQL is not for serious applications, that NoSQL is just for startups or just a place for organizations to put their non-critical data. That is simply not true with MarkLogic.

MarkLogic has all of the critical enterprise features that made last-generation relational databases so reliable, and which are absolutely critical for storing and managing enterprise data. Some of MarkLogic's key enterprise features include:

- **ACID transactions** to ensure data consistency and avoid data loss or corruption
- **Certified security** that allows MarkLogic to run in enterprise data centers
- **High availability and disaster recovery** so that data is always available
- **Performance monitoring** to keep a close eye on how resources are provisioned and used
- **Enterprise management tools** that provide automated approaches to common tasks

With MarkLogic, all of these features are more than just boxes on a checklist. MarkLogic has *proven* all of them in mission-critical systems at the U.S. Department of Defense, large investment banks, healthcare payers, global media organizations, and many other industries in which success is not optional.

LEADING ORGANIZATIONS ACHIEVING MORE WITH MARKLOGIC

Hundreds of organizations have embraced change and innovation by using MarkLogic to power the future of their businesses. These examples highlight some of the many successes.

RUNNING AN OPERATIONAL TRADE SYSTEM AT A LARGE BANK

A top-5 investment bank replaced 20 relational databases with one MarkLogic database. MarkLogic now runs the derivative trade store at the bank, managing over 100,000 trades per day, and 32 million live deals in the system at any one time. This high volume generates cash flow risks in excess of $100 million. In addition to the huge cost savings from using MarkLogic, the bank also achieved a global, real-time, unified, and accurate view of their derivative trading business.

ENROLLING MILLIONS OF BENEFICIARIES THROUGH HEALTHCARE.GOV

The Centers for Medicare & Medicaid Services (CMS) provides access to health coverage for millions of Americans through HealthCare.gov—validating eligibility requirements for insurance plan enrollment across multiple federal data sources and handling hundreds of thousands of concurrent users, all with zero data loss. In the first two years, the system enabled 12 million Americans to sign-up for health insurance. This success is in stark comparison to some of the failed state health exchanges, failures that have even led to one state to take a relational database vendor to court.[18]

18 Shelby Stebens. "Oracle sues Oregon officials in healthcare website dispute," Reuters, February 27, 2015 <http://www.reuters.com/article/2015/02/27/us-usa-healthcare-oregon-idUSKBN0LV2LK20150227>

> “MarkLogic has helped us get more rapid product development and to engage the business users with the IT team. So IT is seen now as very much business critical, part of the solution rather than the problem to delivering the solution.”
>
> Andrea Powell, CIO of CABI (The Centre for Biosciences and Agriculture International)

PERFORMANCE INCREASES FOR THE BBC’s iPLAYER STREAMING SERVICE

The iPlayer is the BBC’s TV streaming service in the UK, handling over 3 billion program requests per year. To manage the massive scalability and performance requirements, the BBC team moved from using relational technology to MarkLogic as the main component for storing and delivering metadata about the BBC’s programs. The BBC already experienced success building a dynamic content delivery platform on MarkLogic for the 2012 Olympics, and wanted to leverage the scalability and flexibility of MarkLogic for iPlayer as well. Once implemented, queries that used to take 20 seconds with SQL only took 20 milliseconds with MarkLogic—orders of magnitude better.

TAKE YOUR DATA BEYOND RELATIONAL

Making the move from the old world to the new may seem daunting at first. For that reason it is often better to start with a smaller project and then ramp up. Below, we provide some recommendations to help plan the transition to using NoSQL.

RECOMMENDED NEXT STEPS

ESTABLISH A SENSE OF URGENCY

Seeing the need to change is critical in order to move past any complacency that may be preventing adoption of NoSQL. With a sense of urgency established, it is then possible to establish a vision and gain buy-in.

BUILD THE RIGHT TEAM

Success requires great technology, but it also requires great people and processes. It is important to identify both decision-makers and implementers in the organization, and understand what processes will inhibit and enable the initiative to take shape.

START WITH THE RIGHT PROJECT

Choosing the right use case to begin with is critical. Often times, it is best to start with a small but impactful project that avoids unnecessary disruption. MarkLogic experts can work with you to ensure that your project is the right fit for a MarkLogic solution.

ENGAGE MARKLOGIC EARLY

Dedicated MarkLogic experts are available who have more combined NoSQL experience than any team in the business. It is important to engage with them early-on during the development phase, when support is most critical.

MORE INFORMATION

- **NoSQL for Dummies eBook** – Get an overview of NoSQL databases with a free eBook info.marklogic.com/nosql-for-dummies
- **What is MarkLogic?** – Read more about MarkLogic’s unique set of features at marklogic.com/what-is-marklogic
- **Inside MarkLogic White Paper** – Understand the internals that make MarkLogic so powerful marklogic.com/resources/inside-marklogic-server
- **Schedule a Meeting** – Discuss your particular use case with a MarkLogic sales representative by contacting us at sales@marklogic.com