MarkLogic

The NoSQL Generation: Embracing the Document Model

May 2014

MarkLogic

Table of Contents

Introduction

NoSQL databases are a new generation of databases that have gained significant market traction because they solve major challenges with the volume, variety, and velocity of big data. NoSQL represents a fundamental change in thinking about how data is stored and managed that is counter to the relational database approach used for Oracle Database 12c, Oracle MySQL, Microsoft SQL Server, IBM DB2, Postgres, and many others. [1]

The term "NoSQL" is a broad descriptor covering a wide range of new databases, generally broken down into four main categories: document, key-value, column-family, and graph databases. Among these categories, document databases are the best general purpose databases. Document databases have a more logical, human approach to modeling data, are generally the most flexible and easy to use, and are the most popular.

Among document databases, MarkLogic differentiates itself as an "Enterprise NoSQL" database because in addition to qualifying as a NoSQL database, it has all of the critical features that enterprises need to run mission-critical applications. This means ACID transactions, high availability, disaster recovery, government grade security, elasticity and scalability, and performance monitoring tools. With MarkLogic, enterprises can embrace the document model and securely move forward into the next era of databases.

The History of "NoSQL"

MarkLogic is now known as an Enterprise NoSQL database, but was originally known mostly for its ability to store and search XML. The original patent filings in 2002 had to do with a new way of storing data using the XML tree structure, including a new way to query that data. These patents were filed by MarkLogic's founder Christopher Lindblad long before anyone coined the term NoSQL. MarkLogic has since adopted the term "NoSQL" as a broad descriptor, also adding the "Enterprise" part to differentiate it from the many new databases invented more recently that have not yet evolved to include enterprise features.

The term "NoSQL" has only been in use since 2009, just five years ago. The term was initially chosen as a twitter hashtag to promote a meetup group to discuss new database technologies in San Francisco. The meetup was organized by Johan Oskarsson, a developer visiting from London, and the term was suggested by Eric Evans, a developer at Rackspace. The term was meant to only have a short lifespan, but it quickly caught on. With the explosion of new databases such as Cassandra, MongoDB, and CouchDB that followed

[1] Gartner, "Hype Cycle for Big Data, 2013", July 31, 2013.

in the wake of Google's Bigtable and Amazon's Dynamo, the market needed a term to describe the new technologies.[2]

One of the big misconceptions is that the term NoSQL means "*No* SQL", and that NoSQL databases do not use SQL (structured query language) as a query language. But, many NoSQL databases do use SQL, often as one option among many supported query languages. For example, MarkLogic supports Java, SQL, XQuery, and SPARQL. For this reason, NoSQL is now generally referred to as "Not Only SQL." Despite the fact that NoSQL does a better job at describing what it is *not* rather than what it *is*, the term is still very

useful for describing a broad class of databases ideally suited for the data problems we deal with today.

Types of NoSQL Databases

NoSQL databases handle the volume, variety, and velocity of big data very well. But, they each handle those three V's very differently depending on their data model. For this reason, NoSQL databases are grouped according to their data model, and include document, key-value, column family, and graph databases. MarkLogic is a document database but can also store RDF triples (a feature called semantics), which gives MarkLogic some graph database capabilities.

Document Databases

Sometimes called "document stores" or "aggregate databases," document databases use documents as the central entity for storage and queries. The term "document" does not necessarily mean a PDF or Microsoft Word document. The document can also be a single block of XML or JSON. An XML document does not

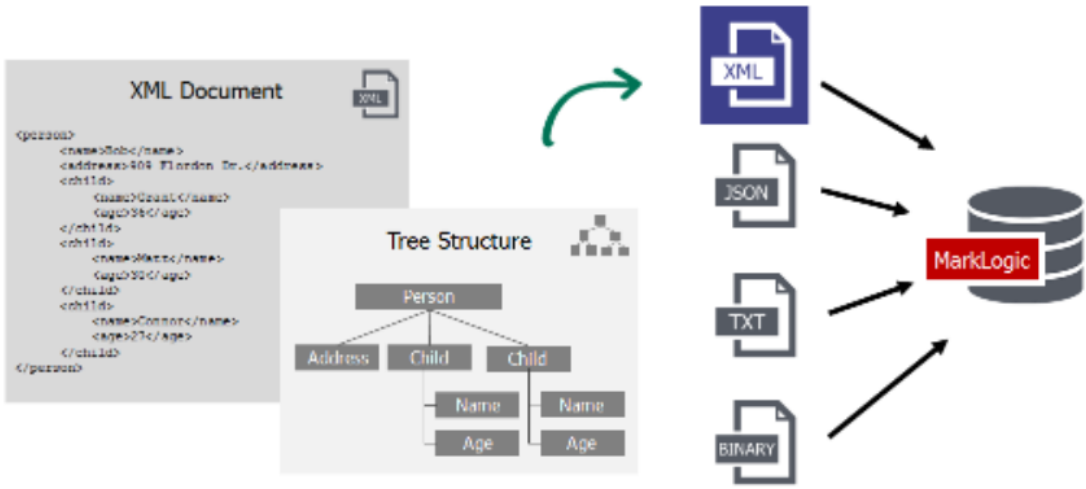

Figure 1: MarkLogic is a document database that can store XML, JSON, text, and large binaries such as PDFs and Microsoft Office documents

[2] Fowler, Martin. *NoSQL Distilled*. Pearson Education, Inc. 2013.

require pre-defined fields and it can also store nested data, often taking on a distinctive tree-like structure that can be queried. Document databases are ideal for storing large amounts of text information such as books or publications, though they can also be used for storing a wide variety of other types of information such as financial data, patient records, or metadata. Put another way, a document could contain all of the information that you would find in the row of a relational table. Because of their flexibility, document databases are the most popular kind of NoSQL database.

Key-Value Databases

Key-value databases have the simplest data model among NoSQL databases—they use a searchable index key associated with a value. Relational key-value databases have been around for many years, but the

newer key-value databases fall into the NoSQL category because they are purpose-built for speed and scale by sacrificing some functionality. For example, there are generally no alternate keys and no foreign keys, no implicit ordering, and no text searching capabilities against the values. These databases are often used for caching website visits and one of the more popular key-value databases, memcache, is named specifically for this purpose. Other uses include storing user preference settings for an application or storing large streams of non-transactional data.

Column Family Databases

A column family database is similar in theory to a table in a relational database, except that it can scale to zillions of rows, and each row can have any number of columns. Each column family associated with a row (i.e., a column family) consists of a key-value pair (a column key and a column value).

Column families became well-known after Google published their Bigtable paper, and has been spurred on by the popularity of Cassandra and HBase. Popular uses for column family databases are for application event monitoring, content management systems, and blogging platforms. Column family stores are not the best choice when ACID transactions are required or when queries are complex or changing.

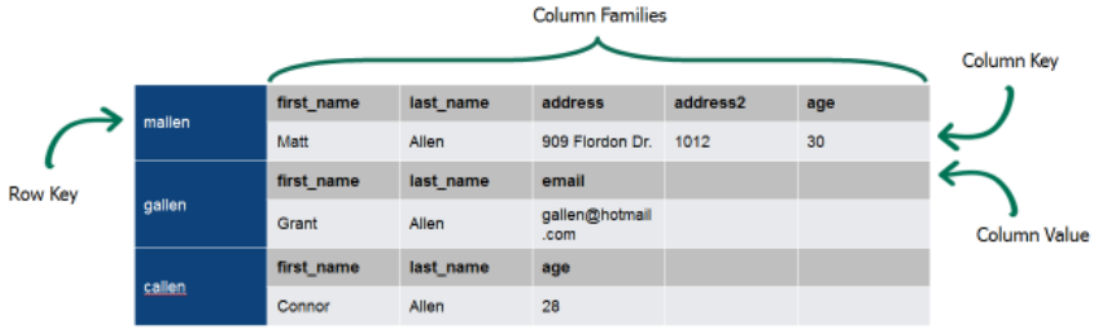

Figure 2: Column family databases such as Cassandra organize data by a row key that is associated with any number of columns

Graph Databases

Graph databases focus on the relationships between the data, which is why graph data is often referred to as "linked data." Data points are called nodes, and the relationship between one data point and another is called an edge. These relationships make graph databases ideal for social media sites such as LinkedIn, Facebook, and Twitter where questions are asked about "degrees of separation" between people.

One way to store linked data is with a distinct kind of graph database called an "RDF triple store." RDF stands for Resource Description Framework, and a triple is the combination of a subject, predicate, and object – for example, "Bo [subject] knows [predicate] baseball [object]." There are a few subtle but important differences between RDF triple stores and general purpose graph databases.

	Graph Databases	RDF Triple Stores
Examples	Neo4j, Titan, OrientDB	MarkLogic, AllegroGraph, Sesame
Types of Data Stored	Unlabeled graphs, undirected graphs, weighted graphs, hypergraphs	RDF triples
Query Language(s)	Cypher, G, GraphLog, GOOD, SoSQL, BiQL, SNQL, and more	SPARQL

	and more	
Other Attributes	Optimized for graph traversals	Graph traversals can be slow
	Cannot do inferencing (i.e., does not infer new triples based on existing data)	Can do inferencing (e.g., if humans are a subclass of mammals and man is a subclass of humans, then it can be inferred that man is a subclass of mammals)

MarkLogic has semantic web capabilities and graph database characteristics because it can store RDF triples and query them using SPARQL. The example below illustrates how MarkLogic Semantics can be used to create an interactive visualization—a distinguishing feature made possible with linked data.

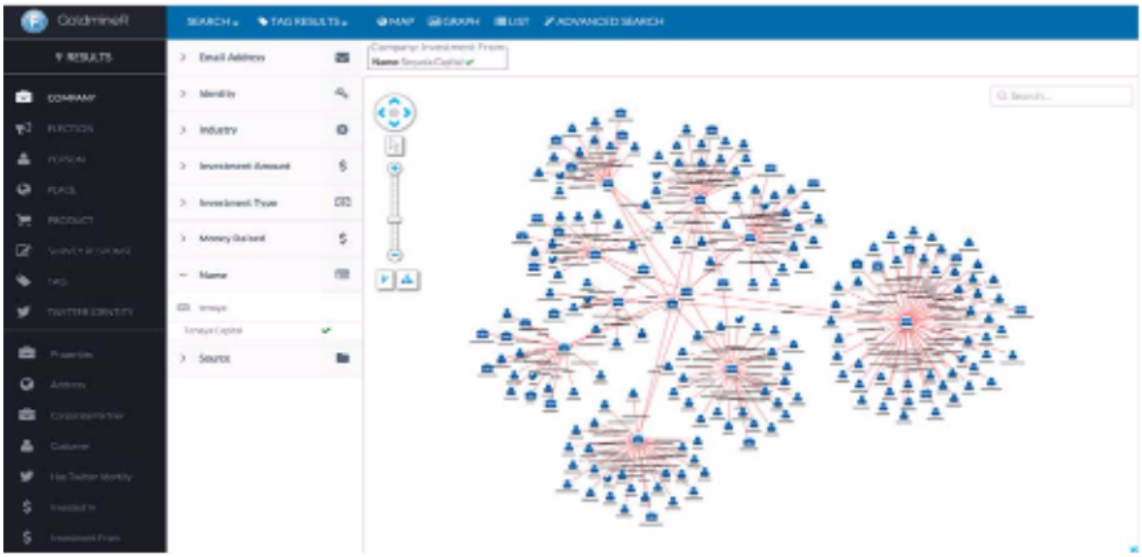

Figure 3: FactGem is an application that uses MarkLogic semantics to show associations such as investment relationships among venture capitalists

Embracing the Document Model

Document databases are the most popular type of NoSQL databases because they are both powerful and flexible enough to serve as a general purpose database. Although MarkLogic does have some graph database capabilities, at its core it is a document database. This has proved to be the right direction because it is much easier to add graph capabilities to a document database than the other way around. The five main reasons to embrace MarkLogic's document model are below.

A More Logical and Human Structure

People naturally organize information using hierarchies and groupings—the structure of documents. This is evident even in industries such as financial services or healthcare where one would think data is always structured. Derivative trades and health data can be easily modeled as documents. And yet, we tried for years to shred this data into relational schemas that no one could agree on. The document model makes it easier to understand what the data is about from a *human* perspective, and fortunately, MarkLogic also makes it easy to understand from a *computer's* perspective.

For a deeper look at how MarkLogic takes a new approach to data modeling, watch the presentation, Data Modeling in NoSQL with XML, RDF, and JSON.

Schema-Agnostic, Structure-Aware

Schema-Agnostic, Structure Aware

Document databases are schema-agnostic but they can enforce a schema when needed because they are also structure-aware. Investment banks frequently need to enforce schemas when handling financial transactions. But, if the bank decides down the road that the schema needs to change, it is a change that can be done rather rapidly. This approach—having schema when you need it—is a huge change from the relational world where it might take months of work to manage changes to schema design.

When it comes to loading data, not much has to be known about the data before loading. It helps if you know how you will structure your queries, because this may impact the primary ID's that are given to a group of documents, but it is not an absolute requirements. With MarkLogic, data is indexed and can be queried immediately after ingesting, regardless of the schema.

All of the data within a document is self-contained and does not rely on data in other documents within the database. This means no foreign keys, and no normalization. Because each document is self-contained, it is easy to distribute data across clusters, making it trivial to setup a cluster and scale a document database. With MarkLogic, you can standup or take down a cluster in the cloud in just a few minutes. The document

model also enhances performance because a group of documents can appear as a contiguous set of content for querying on-disk.

For a deeper look at how MarkLogic handles legacy schemas for investment banks, watch the presentation, Schema on Read in Financial Services.

Easy Application Development

It is not surprising that developers are usually the champions of NoSQL within most IT departments. NoSQL simplifies their lives. The greatest benefit is the time savings from not having to do relational modeling on unstructured information, or on aggregated multi-structured data. The document model, in particular, saves time because data is often already in a document format as XML or JSON.

For example, Founder's Online, an application built by the University of Virginia Press in collaboration with the National Archives, contains almost 150,000 searchable documents that were tagged with XML and then loaded to MarkLogic. This application was created by two developers in a number of months and achieves serious scale, supporting 120ms response times with five thousand concurrent users.[3]

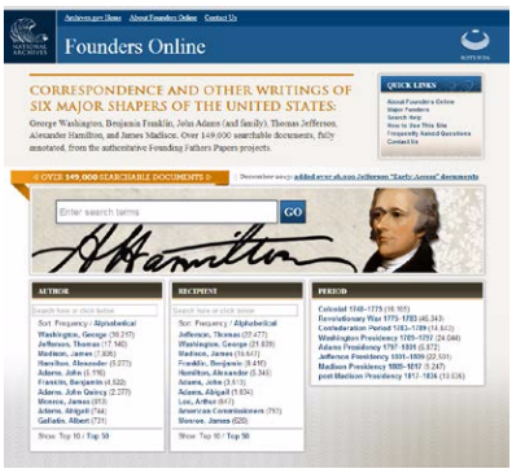

Figure 4: Founder's Online, a powerful search app built by two developers

Developers also favor the document model because it plays well with the languages that developers love—PHP, Ruby, and JavaScript—which are primarily object-based. It is easy to think of the object as the document with these languages. When documents are stored natively as JSON in the database, it is it is

possible to use JavaScript and JSON in the database, server, and on the client in the front-end. That simplicity means data does not have to be transformed when moving between tiers, which reduces the workload on the server and makes development much smoother. This simplicity also creates flexibility because the application and business logic can be put in any tier. If a mistake is made, the expense of changing it later on is minimal.

For a deeper look at one company's fast approach to developing applications with MarkLogic, watch the presentation, Building Applications on MarkLogic Fast and Easy.

[3] For more information, watch the presentation, Planning For Growth With and Without Performance Metering, delivered by David Sewell, Editorial and Technical Manager at University of Virginia Press.

Advanced Search

One of the major drawbacks of more simplistic NoSQL databases like key-value stores is that queries usually only apply to the primary key. In a document database, queries apply to all of the data, including the document ID and the document's contents. Document databases can also rely on indexes to support search. MarkLogic has almost 30-different indexes that can be toggled on-and-off to provide a rich and customizable search experience, including faceted search and real-time alerting. These search features were built-in to MarkLogic from the start, and in fact, MarkLogic's founder has a deep background in search: Christopher Lindblad was the architect of Ultraseek Server.

MarkLogic also supports numerous other search features including word and phrase search, Boolean search, proximity, wildcarding, stemming, tokenization, de-compounding, case-sensitivity options, punctuation-sensitivity options, diacritic-sensitivity options, document quality settings, numerous relevance algorithms, individual term weighting, topic clustering, faceted navigation, and custom-indexed fields.

These many features are made possible by MarkLogic's use of the document model, but only MarkLogic has search built-in. Other document databases must rely on bolt-on technology like Lucene or Solr to provide search capabilities, which adds complexity to the technology stack. Another differentiator is that documents can be searched immediately upon loading them into MarkLogic when their contents are indexed.

For a deeper look at how MarkLogic conquers database search, watch the presentation, Search, Relevance, and Context: Getting the Most out of MarkLogic Search.

Enormous Variety of Potential Use Cases

Enterprise-grade document model databases are flexible and powerful enough to serve as a general purpose database for an enormous variety of use cases. MarkLogic is a perfect fit anytime there is a need to eliminate data silos, use a single platform for search and analytics, reduce storage costs, better secure data, or develop an application faster. This applies to almost any industry, from media and publishing to financial services and healthcare:

- **Media and Publishing:** This industry was the first to adopt document databases. One large publisher, LexisNexis, was the first MarkLogic customer and continues to use MarkLogic today. Another publisher, Wiley, has used MarkLogic to consolidate 4 Million articles, 9,000 books, and thousands of reference works. They gained a 50% growth in usage, and after strategic acquisitions of content libraries, were able to quickly absorb and monetize that new material.
- **Financial Services:** Investment banks require strong governance policies, and need to respond quickly to regulators. A tier-1 bank had trouble developing risk profiles and conducting post-trade reporting because of the disparate heterogeneous data sources in legacy mainframes and Sybase

databases. But, with MarkLogic, they were able to bring that data into a single system, helping them save millions of dollars in IT costs and respond faster to regulators.

- **Healthcare:** Healthcare is another regulated industry that struggles to manage the variety of data, and is squeezed by tightening margins and government oversight. One MarkLogic customer, Zynx Health, partners with different hospitals across the United States to provide personalized plans of care. Despite the challenge of partnering with over 2,000 hospitals, they were able to build an application in less than a year that each of those hospitals now relies on to improve care quality and meet meaningful use requirements.
- **Government:** Government agencies love documents. But, when budgets get squeezed and the pressure mounts to move services online, they frequently run into the problem of developing applications in a timely and efficient manner. Government agencies are also wary of getting locked into building an entirely new system or replicating their data again and again for each new application—and of course they have serious data security needs. MarkLogic has helped solve this problem for the FAA, CMS, FDA, DoD, and the intelligence community.

While relational databases and other types of NoSQL databases will continue to serve specific purposes, document databases such as MarkLogic will help solve the most pressing big data challenges organizations face today.

To learn more about the many potential opportunities to use MarkLogic, watch the presentation, Reimagine: Data, Applications with MarkLogic.

Defining Enterprise NoSQL

There is a misconception that NoSQL is not for serious applications—that NoSQL is just for startups, or just a place for businesses to put their non-critical data. As the examples above illustrate, that is simply not true anymore. "Enterprise NoSQL" means a database that has the ability to handle the volume, variety, and velocity of data like all NoSQL solutions, *AND* has the necessary features to run at the heart of the business. Unless a NoSQL solution has the following features, it is not enterprise grade, and should not be used for mission critical applications:

- **ACID Transactions:** ACID transactions are not just for banking. Without ACID transactions (atomicity, consistency, isolation, durability), there is also a high probability of data loss. And, if the network fails for any reason, the result can be catastrophic for the database. Enterprises need support for multi-record transactions and rich, multi-term queries—additional features made possible with ACID transactions.

- **High Availability and Disaster Recovery:** Organizations should not have to implement entirely new procedures and governance structures to manage data in a NoSQL database. Enterprises need High Availability (HA) with local disk automatic failover, point-in-time recovery, and asynchronous

cross data center replication for Disaster Recovery (DR). This is necessary so that if the data center does go down, data is not lost and the database does not have to be rebuilt.

- **Government Grade Security:** It is not just governments that need security. The risk of not securing data is simply too high, which is why, according to Gartner, investment in IT security will increase by around 39%, to $93 billion, by 2017. Government grade security means having a top certification from the National Information Assurance Partnership (NIAP) Common Criteria Evaluation and Validation Scheme (CCEVS) for supporting key security functions such as audits, user data protection, security management, data protection, TOE (target of evaluation) access, and identification and authentication (including third-party support for LDAP and Kerberos).
- **Elasticity and Scalability:** Enterprises should be able to scale up or down in minutes to meet data volume and access demands, while also avoiding over-provisioning and over-spending. This needs to be done without downtime, inconsistency, or risk of data loss. The database should run easily on Amazon Web Services or other Cloud providers but should also have flexibility for deployment in other virtualized environments or on premises.
- **Monitoring and Performance Tools:** Great tools for monitoring and management ensure that the IT team is just as happy with the platform choice as the developers. Enterprises need automatic rebalancing and cluster monitoring tools and rich APIs for management, process automation, access controls, database cloning, and audit trails. They also need out-of-the-box interfaces that link to common tools such as Nagios and HP OpenView.

MarkLogic is built from the ground up to include all of these features, and continues to focus on building enterprise features that no other NoSQL solution has. If you are interested in learning more, you can find additional resources online at MarkLogic.com. In particular, you can take a deeper dive by reading the white paper, Inside MarkLogic.

If you want to talk to us about installing MarkLogic in your organization, give us a call at +1-877-992-8885, or email a sales representative at sales@marklogic.com.

About MarkLogic

For more than a decade, MarkLogic has delivered a powerful, agile, and trusted Enterprise NoSQL database platform that enables organizations to turn all data into valuable and actionable information. Organizations around the world rely on MarkLogic's enterprise-grade technology to power the new generation of information applications. MarkLogic is headquartered in Silicon Valley with offices in Washington D.C., New York, Chicago, London, Frankfurt, Utrecht, and Tokyo. For more information, please visit www.marklogic.com.

999 Skyway Road, Suite 200, San Carlos, CA 94070 › US: +1 650 655 2300 › INT'L.: +1 877 992 8885
sales@marklogic.com › www.marklogic.com