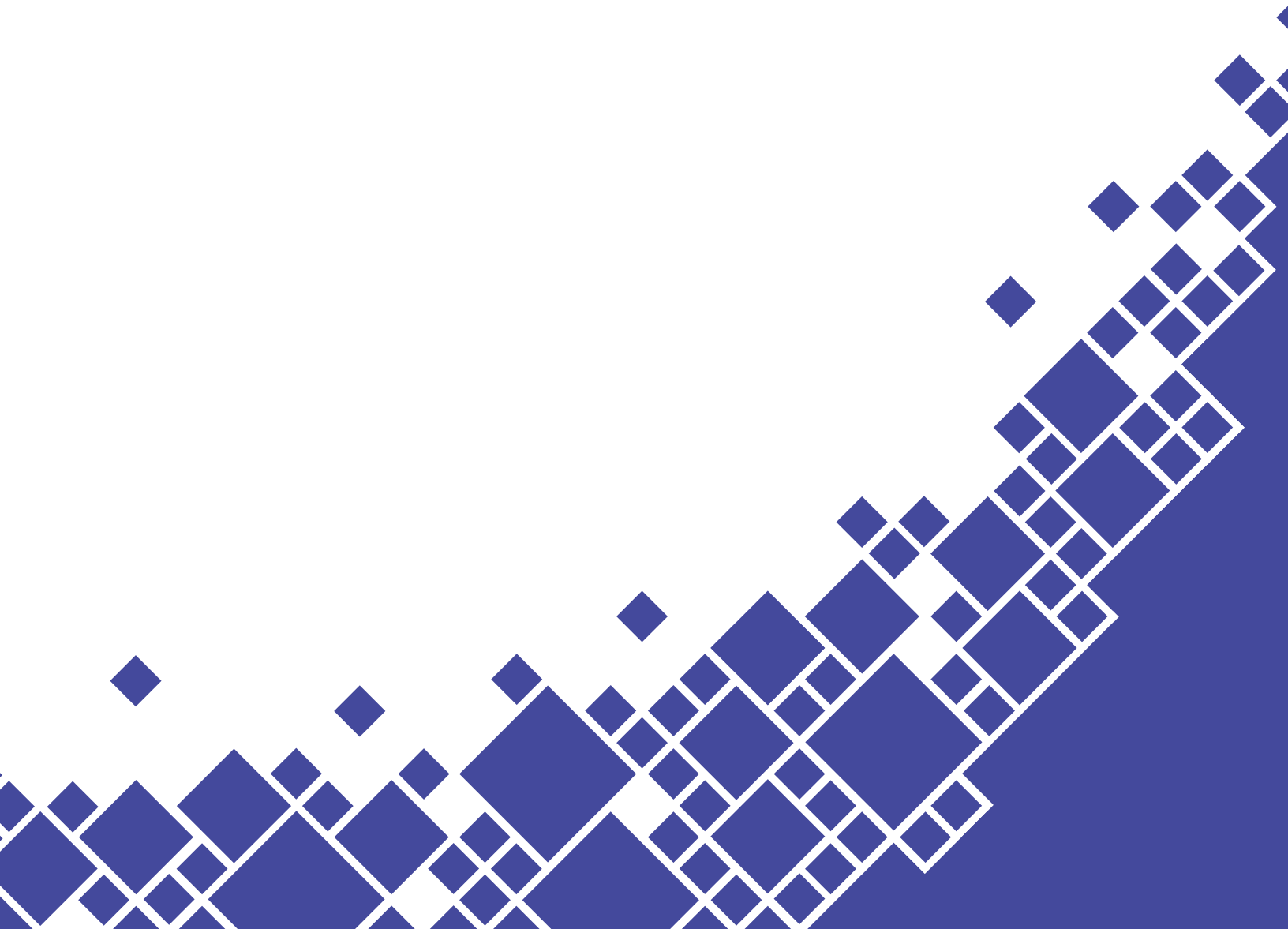
MarkLogic®

RETHINK DATA MODELING

MARKLOGIC WHITE PAPER • MARCH 2016

Traditional data modeling is inadequate. Today, organizations are constrained by relational technology and they need a better approach to data modeling in order to integrate data faster and build smarter applications. For that reason, organizations are now choosing a multi-model approach using NoSQL and semantics.



CURRENT STATE ASSESSMENT

Is your organization in need of change, or do you think you are doing all right with your data? The answers to the questions below provide a baseline assessment to address that question. 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 related data, but no integrated view of that data?	✓	✗
	3. Are there numerous data sets spun off from core systems and no longer centrally managed or governed?	✓	✗
	4. Are there large IT projects that have been behind budget or failed to launch due to data integration challenges?	✓	✗
	5. Are there database schemas so complicated that no one wants to touch them anymore?	✓	✗
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 were 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. Is important metadata or reference data stored outside of the database, in an Excel spreadsheet or some other place?	✓	✗
	10. Are there ever performance problems or bugs that may have resulted from complicated middleware?	✓	✗

Contents

INTRODUCTION

Data modeling is crucial for every organization. Data models define the details of how information is stored, documenting real life people, places, and things and how they relate to one another. For example, a company has customers, and customers have purchases. How these entities and relationships are modeled forms the basis for using and sharing data and directs how organizations build applications. At a high level, data models represent how organizations think about the world in which they operate.

Unfortunately, the traditional approach to data modeling is inadequate. The process for data modeling involves developing a conceptual model of entities and relationships based on the domain of interest, translating that into a logical model, and then further translating that into a physical model that can be implemented in the database. This approach, known as entity-relationship modeling ("ER modeling"), has been a standard since it was first proposed in 1976.

But in practice, database designers ignore conceptual modeling. According to one study, there was not a single instance of conceptual ER modeling among the Fortune 100 companies surveyed.[1] Why is that? The problem is that the world has too much complexity to fit into rows and columns of a relational database. Looking at an ER diagram for a relational database, it is not possible to discern much about the business or the logical whole of what is being described. There is a disconnect and the physical models just end up as convoluted, poor depictions of the world they are meant to portray.

Organizations make valiant efforts to build and maintain foolproof relational databases perfectly connected together. But eventually a change is required—a new data source comes around, a different question is asked, or data must be integrated into a new system. Relational databases show their weaknesses when these events occur, but such events are now common. For the past 30 years, experts have tried to make relational databases work, but today, data modeling remains an unsolved problem.

To integrate data faster and building smarter applications, organizations are adopting an alternative multi-model approach using NoSQL and semantics. NoSQL and semantics provides a more flexible, descriptive, and useful model. In the words of one MarkLogic customer, NoSQL and semantics "removes the shackles of relational technology."[2]

Currently, there are many other new databases on the market, but MarkLogic® is the only enterprise-grade, multi-model database that combines all of the benefits of a NoSQL document database and semantics in a single platform. It is for this reason that leading organizations such as the British Broadcasting Company (BBC), NBCUniversal, Broadridge Financial Solutions, Amgen, and others are rethinking data modeling with a multi-model approach using MarkLogic.

THE NEED FOR A BETTER APPROACH

Data integration is one of the most pressing challenges for organizations today. It matters to banks that need better reporting due to increased oversight, companies undergoing mergers and acquisitions, and governments that must improve national security.

Regarding national security, the 9/11 Commission report stated the importance of data integration, stating that, "A 'smart' government would integrate all sources of information to see the enemy as a whole."[3] Unfortunately, traditional database design does not capture enough information to enable data integration—it falls short of even capturing the kind of information that would be valuable for data integration. To integrate data faster and easier, organizations need a different kind of database.

1 M. L. Brodie and J. T. Liu. "The power and limits of relational technology in the age of information ecosystems." Keynote at On The Move Federated Conferences, 2010.

2 Watch the interview with Paolo Pelizzoli, SVP and Global Head of Architecture at Broadridge Financial Solutions, online at https://www.youtube.com/watch?v=TB1tLrM_z1k.

3 National Commission on Terrorist Attacks on the United States, The 9/11 Commission Report: Final Report of the National Commission on Terrorist Attacks upon the United States: Official Government Edition (Washington, DC: U.S. G.P.O. 2004) p.401,416. <https://www.gpo.gov/fdsys/pkg/GPO-911REPORT/pdf/GPO-911REPORT.pdf>

TRADITIONAL DATA MODELING IS INADEQUATE

WHAT WE WERE PROMISED

In a famous computer science paper published in 1976, Peter Chen put forward the idea of capturing information about the real world as entities and relationships.[1] The new approach, called ER modeling, was intended to unify multiple storage and transaction models to better represent the real world. It soon became a standard for data modeling.

In ER modeling, database designers look at the most import entities (e.g., objects with a physical existence such as an employee, car, or house; or an object with a conceptual existence such as a company or job). Next, they distill out the attributes (e.g. the name, age, address, and salary of the employee). This information then guides implementation of the physical database. This process involves three different diagrammatic models, described below.

CONCEPTUAL DATA MODEL

The conceptual data model identifies the general entities and relationships at a high level. It uses non-technical terms that executives and managers understand, and serves as a point of reference for the technical specifications that follow.

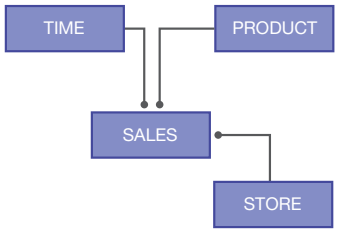


Figure 1: Example of a simple conceptual data model
(Source: http://www.1keydata.com/)

LOGICAL DATA MODEL

The logical data model is more detailed than the conceptual data model. It standardizes people, places, things, and relationships—and the rules and events among them. It does not go so far as to specify the technology for implementation, but it does anticipate the physical model that follows. It includes a normalized view of the entities (tables), attributes (columns/fields) and relationships (keys).

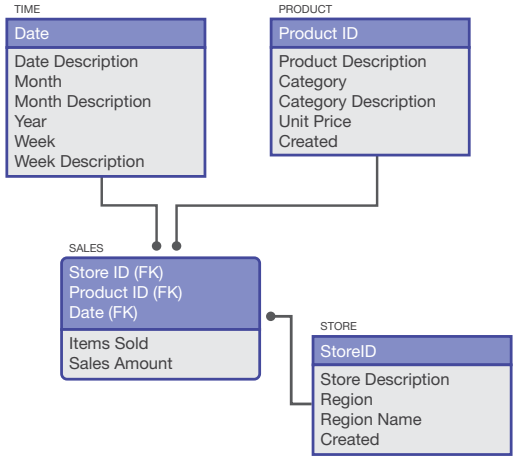


Figure 2: Example of a simple logical data model
(Source: http://www.1keydata.com/)

PHYSICAL DATA MODEL

The physical model is the closest representation to how the data is actually stored in the database. It is detailed, technical, and technology (and even vendor) dependent. It includes the actual table and column names, and takes into consideration performance goals, indexes, constraint definitions, data distribution, triggers, stored procedures, and more.

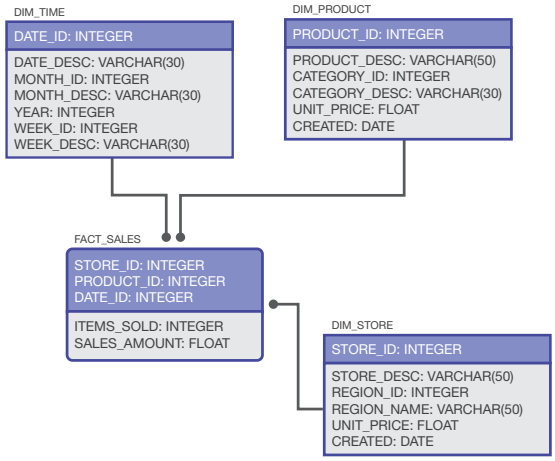


Figure 3: Example of a simple physical data model
(Source: http://www.1keydata.com/)

1 P. Chen. The entity-relationship model - toward a unified view of data. ACM Transactions on Database Systems, 1(1):9–36, 1976. <http://www.inf.unibz.it/~nutt/IDBs1011/IDBPapers/chen-ER-TODS-76.pdf>

> “The main problem is that the world has too much complexity, and businesses operate faster than IT can keep up.”

WHAT WE GOT

Unfortunately, when put into practice, the conceptual ER diagrams meant to describe entities and relationships get immediately supplanted by physical models designed according to the hard constraints of relational databases. The conceptual models are often not even produced. Database designers jump straight to implementing physical models, without building the foundation to guide their development.

In one study, researchers could not find a single instance of conceptual ER modeling among the ten Fortune 100 companies they surveyed, and they also found significant problems with data modeling in general. They found that, “A typical Fortune 100 company has about 10 thousand different information systems, a typical relational database is made of over 100 tables, each containing between 50 to 200 attributes. Formalized conceptual models, as well as the theory developed around normalization, are not used. Physical modeling is frequently delayed until performance problems arise.”[2] Another database expert reports that, “[the database design] for 50 percent of applications is substantially flawed.”[3]

Why are database designers not doing proper data modeling?

The main problem is that the world has too much complexity, and businesses operate faster than IT can keep up. Conceptual models that depict the real world do not translate well into the physical models that DBAs and architects use to build their relational databases. Database designers focus enormous amounts of time on the physical models, designing ways to persist data in relational databases in ways that relational databases can handle. This traditional approach results in a rigid system of tables connected together using primary keys, foreign keys, and joins. Unfortunately, it is a poor representation of the world it was designed to portray. It is not ideal for application development due to the problem of impedance mismatch. And, it is also problematic when a change is required, a new data source comes along, or different data sources need to be integrated together from silos—events that are all quite common today.

NOW WE’RE PAYING THE PRICE

What typically happens in most organizations is that IT lags behind the pace of the business. With the database architecture, tables and columns start accumulating in the database until no one knows quite how the data model relates to the actual business entities, and no one can be sure how a change in one place may affect other aspects of the model. The database becomes overly complex, difficult to reason through, virtually impossible to change, and no one wants to touch it anymore.

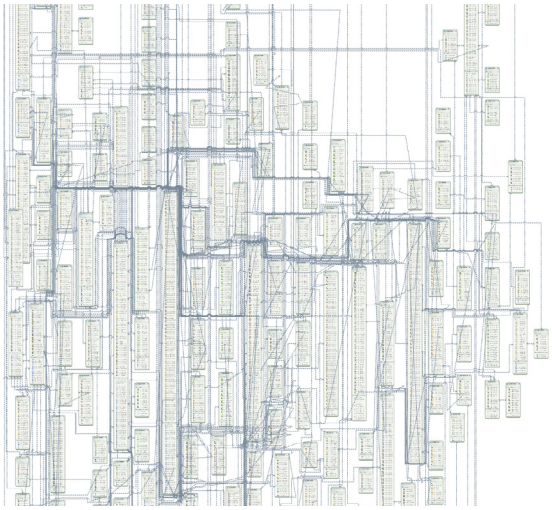


Figure 4: Snapshot of a complex relational model (Source: http://www.plandora.org/docs/mer.png)

Most organizations do not have just one schema, either—they have multiple schemas across various silos. Additional schemas often appear as a result of a merger, acquisition, or integrating a new data provider. These schemas accumulate and grow over

2 M. L. Brodie and J. T. Liu, 2010.

3 Roberto V. Zicari, “How good is UML for Database Design? Interview with Michael Blaha.” ODBMS.org, July 25, 2011. <http://www.odbms.org/blog/2011/07/how-good-is-uml-for-database-design-interview-with-michael-blaha/>

> “40% of the cost associated with information systems is due to data integration problems.”

time, creating layers upon layers of complexity. So, it becomes more expedient to just spin off a subset of the data to a new data mart needed for a particular application or report. Of course, this only results in more copies of the data floating around, more security and data governance challenges, and more problems integrating data and building applications.

To integrate data from such disparate silos, the organization must go through the painful process of determining the similarities between attributes, usually without any associated metadata. At one large Swedish bank, for example, it was discovered that across their many databases, there were 31 different definitions for “nominal amount”—and no additional information to indicate what the data actually represented. The confusion with just that one important term results in many trickle-down costs and risks. Now imagine that problem happening with hundreds or thousands of terms.

These problems are not just headaches for DBAs. They create serious risks for the entire organization, higher costs, and longer project timelines. One study reports that, “40% of the cost associated with information systems is due to data integration problems.”[4] And, in 2015 alone, organizations spent $5 billion on data integration software.[5]

Over the past 30 years, experts have tried to make relational databases work, but traditional data modeling is inadequate. The fundamental mismatch between relational databases and the increasingly varied data that today’s organizations handle will only become more problematic if left unchanged. Forrester Research sums up the problem: “As data structures get more complex and data volume grows, traditional relational databases—and their need for a pre-defined schema are falling short.”[6]

4 M. L. Brodie and J. T. Liu, 2010.

5 Gartner. Forecast: Enterprise Software Markets, Worldwide, 2011-2018, 4Q14. 2014. <https://www.gartner.com/doc/2944023/forecast-enterprise-software-markets-worldwide>; Includes: Data Integration Tools, Data Quality Tools, and Other Data Integration Software.

6 The Forrester Wave™: NoSQL Document Databases, Q3 2014; An Evaluation Of Four Enterprise-Class NoSQL Document Databases Vendors. Noel Yuhanna with Leslie Owens, Emily Jedinak, and Abigail Komlenic.

SPECIFIC PROBLEMS WITH RELATIONAL MODELING

In this section, we elaborate on the specific challenges that make relational databases ill-equipped for modern day data modeling. If you are interested in a broader discussion covering all of the key reasons why relational databases are not working for today’s data, read the white paper Beyond Relational, which discusses other aspects such as application development and scalability.

DIFFICULTIES MODELING ENTITIES AND RELATIONSHIPS

When looking at an ER diagram for a relational database, it is not possible to discern much about the real life business model or the logical whole of what is being described. That data model does not translate well into today’s world of dynamic, complex, connected relationships.

When doing relational modeling, the process typically involves the following steps:

1. Give each attribute its own field
2. Group attributes into tables
3. Assign one primary key to each table
4. Eliminate duplicate attributes

The problem is that relationships between entities are defined only inherently via the rows and columns in tables, or by using pointers between tables based on foreign key relationships and constraints. Some types of relationships such as associations or inherited relationships are implicit, documented and governed outside the database, or just ignored. The more complex the information is, the more rows, columns, and pointers. This gets overly burdensome to model, and even more difficult to query.

As a business changes, the schema must also be updated. Unfortunately, making changes to relational schemas is usually time consuming and expensive. For one MarkLogic customer, “Even a simple change like adding or replacing a column in a table might be a

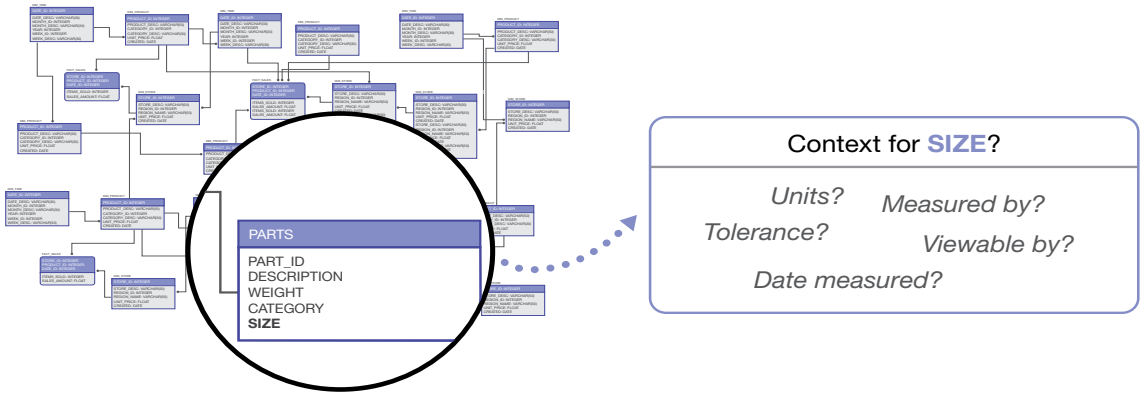


Figure 5: Relational databases are not designed to store context

million dollar task."[7] The process of managing change with relational databases usually involves building out separate join tables to capture many-to-many relationship and doing a lot of defensive programming to avoid problems downstream. Such manual tasks are error prone and do not provide a long term solution for managing change.

Relational databases also have no universally accepted standards for modeling and querying people, places, and things. Each attribute in a relational database stands on its own, and every DBA models it differently. Furthermore, because of data variety, database designers are often working with tables that have been populated with rows and rows of NULL cells ("sparse data") or with column names described with generic VARCHAR data types that are part of tables with enigmatic names that only a few people understand. These issues destroy cohesion and cause problems for normalization and querying. Ultimately, the end result of relational shortcomings is that database designers are left to convey, usually externally, how to find meaning and join distinct artifacts into a logical whole.

DIFFICULTIES MAINTAINING CONTEXT

Relational databases provide a mathematically consistent view of the data, but the valuable context around the data—the *semantics* of the data—is lost.

This was one of the main points in Peter Chen's paper on ER modeling, which stated that, "The relational model is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world."[8]

As an example, consider a database that has a table of parts with a column named "Size," and a column value of "42" in one of the rows in that column. But, where is the contextual information: *What are the units of "42"? What is the tolerance? Who measured it? Is it an estimate? When was it measured? Who can see this column?*

Things are even more problematic when handling data that is more qualitative, less structured, or has class or property relationships that must be accounted for. Consider a table with a column name labeled "Customer." *What does that label mean? How are customers related to other things that are important to the organization? Is this group of customers a subclass of a broader group of customers? Which systems can generate customers? How are customers represented to the applications? How many customers does the organization know about? Which customers do not adhere to the business rules?*

Unfortunately, the context of the data is not in the database. If it does exist, it may be stored in

7 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.

8 P. Chen, 1976.

> “With MarkLogic, the new iPlayer system handled over 3 billion program requests in the first year. Performance increased, with SQL queries that used to take 20 seconds only taking 20 milliseconds with MarkLogic.”

SharePoint*, a Microsoft Excel* spreadsheet, or an ER diagram printed out a few months ago and hung on a DBA’s office wall—everywhere except for the database where the data is stored. Making sense of the data within one database is difficult. Across data silos it can be impossible. This makes getting and reconciling metadata and reference data a brittle and expensive process.

LEADING ORGANIZATIONS ADOPTING NEW APPROACHES

Relational databases provide robust technology and there are many reasons why they are the most used type of database. But it is a mistake to think that relational databases should be used to do things they were not designed for. Driven by the need to change, organizations today are fundamentally rethinking their approach, and MarkLogic provides an alternative using NoSQL and semantics that solves many of the challenges with traditional data modeling. In this section, we quickly highlight some of the successes that organizations have had in transitioning to a new approach with MarkLogic.

MANAGING PROGRAMS METADATA FOR THE BBC’S iPLAYER STREAMING SERVICE

The BBC was the first global media company to embrace using NoSQL and semantics for a mission-critical application at scale. For the 2012 Olympics, the BBC moved from a relational database to a new architecture built using NoSQL and semantics in order to automate the aggregation, publishing, and repurposing of content.

Since then, the BBC has expanded its use of NoSQL and semantics. One example is with the iPlayer TV streaming service, which was relaunched in 2014 using MarkLogic as the main component for storing and delivering metadata about the BBC’s programs. They made the change because their legacy system, based heavily on MySQL and Memcached, showed performance and reliability problems. And, the whole end-to-end process for making new content available on iPlayer was too slow, often taking days.

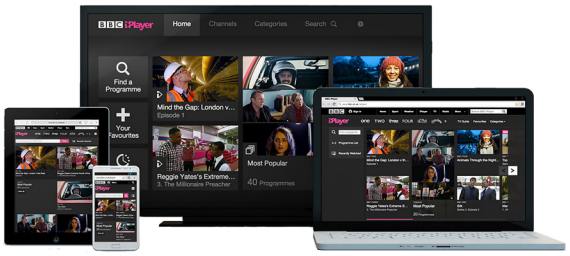


Figure 6: BBC iPlayer TV streaming service

With the increasing volumes of users and need for a simpler and more reliable system, the BBC knew they needed a change. After struggling to solve the problem with relational technology and spending a lengthy period of time prototyping and testing alternatives with major database vendors, the BBC chose MarkLogic. With MarkLogic, the new iPlayer system handled over 3 billion program requests in the first year. Performance increased, with SQL queries that used to take 20 seconds only taking 20 milliseconds with MarkLogic. And, the end-to-end process for making content available dropped from days to minutes. The Lead Technical Architect at the BBC stated that, “MarkLogic makes things so simple that the architects struggle to get their head around it!”

BUILDING A SEMANTIC METADATA HUB AT A LEADING ENTERTAINMENT COMPANY

A large entertainment company uses MarkLogic to manage hundreds of thousands of product titles, characters, distribution rights, and technical information that are all vitally important. Previously, the data was held in disconnected information silos, resulting in data quality issues, inconsistent governance, and an inability to re-use data efficiently. They even had to employ couriers to hand-carry physical assets between buildings.

> “With alternative approaches, the organization would have needed to integrate at least three different technologies to achieve the same level of functionality.”

The solution to their problem was to establish a centralized, semantic metadata hub using MarkLogic. The hub makes it possible to store and access all of the organization’s metadata about its assets in one place, enabling rapid response to emerging business challenges and changes in the competitive landscape.

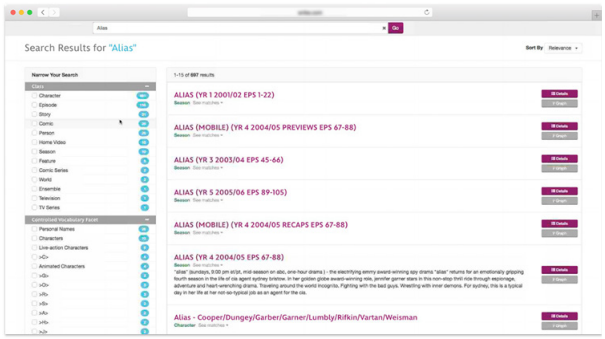


Figure 7: Search application built using semantics at a large entertainment company

The hub uses MarkLogic to store and search the data and also Smartlogic*, a MarkLogic partner, for classification, publishing, taxonomy and ontology management, and semantic enrichment functions. The hub works by ingesting data from multiple silos and then providing a natural language search interface to query the data, in addition to providing the data to downstream systems.

Because the data model is driven by NoSQL and semantics, users can leverage the relationships in the data. For example, it is easy to see how a movie had a certain star appearing in it, that she played a specific character in that movie, that the character was from a specific place, and that the movie was animated. A relational database, by contrast, would have difficulty managing all of these relationships, and the costs would be prohibitive. The organization was also able to create a simpler information architecture since MarkLogic combines a database, search, and application services in one platform. With alternative approaches, the organization would have needed to integrate at least three different technologies to achieve the same level of functionality.

INTELLIGENT ANALYTICS FOR ACADEMIC PUBLISHING AT THE APA

The American Psychological Association (APA) gets over 70 percent of its revenue from publishing. Their “first to information” advantage is crucial. But they had problems with their large database of 57 million articles, books, journals, dissertations, videos, tests, measures, etc.—to be accessed by over 150,000 users. The APA’s legacy solution based on Oracle and Lucene/Solr (a relational database and a search engine) showed inconsistent search results, a poor user experience, and slow content turnaround. It was cumbersome to manage the large database with millions of rows. And, although highly structured, the data was not structured in a way that was suitable for relational databases. The APA also had unnecessary costs due to having too many servers and too many developers focused on maintaining the legacy system.

Figure 8: Visualizations of APA data using semantics

The APA transitioned to MarkLogic to manage their data pipeline. This simplified their architecture, as MarkLogic is a database with built-in search. They also took the next step to enrich their data using semantics so that their users can explore data and analyze relationships between authors, subjects, journals, and

> “JSON and XML documents provide a more natural approach to modeling the variable and complex data that today’s organizations work with.”

sponsors. MarkLogic also enabled the APA to run a higher volume of content through their systems and maintain higher quality, integrity, agility, and scalability in step with the increasing size of their business. DBAs were happy with the new system because there was less infrastructure to maintain, and developers were happy because development cycles became faster, allowing them to focus more time working on new features and functionality.

THE BENEFITS OF NOSQL DOCUMENT DATABASES

One of the main reasons the organizations in the preceding section chose to use NoSQL and semantics was that they needed a more flexible, useful, and descriptive model for their data that would allow them to integrate data better and build smarter applications—all at a much faster pace than they could previously. The remainder of the paper discusses the specific benefits of this new approach.

To start, we discuss the benefits of NoSQL document databases. NoSQL databases fall into four distinct types: document, graph, column, and key-value. Among these types of NoSQL databases, document databases are by far the most popular, and in this section, we focus only on the benefits of document databases since MarkLogic stores data as documents. If you are interested in learning more about NoSQL in general, download the eBook, Enterprise NoSQL for Dummies.

FLEXIBILITY OF THE DOCUMENT MODEL

One of the main benefits of document databases is that they use a flexible, schema-agnostic data model that brings agility and richness when it comes to modeling data and building applications. Rather than normalizing data across tables in a relational model, the document model keeps all of the data in JSON or XML documents. JSON and XML documents provide a more natural approach to modeling the variable and complex data that today’s organizations work with, they are more human-readable, and they more closely map to the conceptual or business model of the data—the real-world entities. For example, if modeling a financial trade, a patient record, or a surgical procedure at a hospital, a single document can contain the necessary information.

```
{
  "hospital": "Johns Hopkins",
  "operationType": "Heart Transplant",
  "surgeon": "Dorothy Oz",
  "operationNumber": 13,
  "drugsAdministered": [
    { "drugName": "Minicillan",
      "drugManufacturer": "Drugs R Us",
      "doseSize": 200, "doseUOM": "mg" },
    { "drugName": "Maxicillan",
      "drugManufacturer": "Canada4Less",
      "doseSize": 400, "doseUOM": "mg" },
    { "drugName": "Minicillan",
      "drugManufacturer": "Drugs USA",
      "doseSize": 150, "doseUOM": "mg" }
  ]
}
```

Figure 9: Example of a JSON document representing a surgical procedure at a hospital

A row in a table in a relational database can be roughly compared to a single document, though documents are not as rigid and can be used for both structured and unstructured data. Each document has its own schema of structure and attributes. That schema can easily evolve, independently of other documents. If all of a sudden a new data feed is identified that has a different schema, the data can still be ingested and stored without having to pre-define its structure or having to adjust any of the existing documents.

Document databases have another benefit when the data is consumed in applications—they do not have an impedance mismatch problem. With relational databases, there is an impedance mismatch between

> “A document database makes it possible to load data as-is and perform any necessary transformations either during the ingestion process or later on after it is loaded into the database.”

the objects that contain the data and code used in application programming and the normalized rows and columns in the database. This architecture eventually leads to performance loss and more opportunities for buggy code. With NoSQL, developers can simply work with data as JSON, XML, or even rich objects throughout the technology stack without having to maintain complex transformations from the relational model. This is a more straightforward approach and it means developers can avoid the risks of developing the wrong application because they can use a more iterative approach to building a system that is more resilient to changes later on.

THE IMPACT OF A BETTER APPROACH

Today, organizations are frequently managing data across silos and they need a database better suited for data integration. When it comes to data integration, the flexibility of the document model is a huge advantage. Rather than spending unnecessary time identifying and profiling data and managing complex ETL processes, a document database makes it possible to load data as-is and perform any necessary transformations either during the ingestion process or later on after it is loaded into the database. Often, transformations are needed to get data to conform to a certain schema. Other times, certain metadata needs to be added to enrich a document. And, these processes must often happen without destroying the integrity of the original source data. The document model makes it possible to do all of these tasks faster and simpler than with a relational database.

At one large healthcare company now using MarkLogic, there was a data integration project involving over 140 human resources-related data feeds, consisting primarily of complex, structured data such as payroll data, employee evaluations, promotions, benefits data, and more. That data needed to be ingested, transformed, and then delivered in real-time to over 50 downstream systems. The project was estimated to take over 40,000 hours of development and multiple years using traditional relational database technology. With MarkLogic, the company was surprised when the project was completed successfully and the new system put into production in less than a year. The new system can handle the complex data ingestion and complex output, and it is more resilient to future changes. The company now relies on MarkLogic as the data layer for all of their HR data, and at a reduced cost compared to their previous system that involved a myriad of data silos.

MARKLOGIC COMPARED TO OTHER DOCUMENT DATABASES

Compared to other well-known document databases, MarkLogic stands out for the following key reasons:

- **Multiple Data Formats** – MarkLogic provides native storage for JSON, XML, and RDF (which we discuss next). In addition, MarkLogic can also store large binaries (e.g., PDFs, images, videos). Most document databases only store JSON.
- **Built-in Search** – MarkLogic has a Universal “Ask Anything” Index for words, phrases, structure, values, security, and more—and even more indexes on top of that (range, geospatial, and triple indexes). With other databases, indexing capacity is limited, and require another bolt-on solution for full-text search.
- **Enterprise Capabilities** – MarkLogic has 100% ACID transactions. Most NoSQL databases do not have ACID transactions, and may lose data or end up with corrupt data. In addition, MarkLogic also has certified security and proven HA/DR.

THE ADDED BENEFITS OF SEMANTICS

Document databases serve as great general purpose databases, but there are still some things about data modeling they are not optimized for. In this section, we discuss how semantics is specifically optimized for storing facts and relationships, and what this capability means for data modeling. If you are interested in a more in-depth overview of what semantics is, including more examples of semantics in practice, grab a copy of Semantics for Dummies.

A SIMPLE AND POWERFUL DATA MODEL

Graph databases have risen quickly in popularity in recent years, and triple stores—where semantic data is stored—are considered a type of graph database. When data starts to take on a graph structure in which entities (people, places, and things) and the relationships between them are the most important thing, it is better to use semantics.

Semantic facts and relationships are expressed with a subject-predicate-object construction, and are known as RDF triples (RDF stands for "Resource Description Framework"). An example of a triple is "John lives in London." These triples are queried using a standard language called SPARQL, which is a lot like SQL.

Using triples to describe facts and relationships is both simple and powerful. As one expert puts it, "Increasingly models are enterprise wide, and contain a high degree of variability for which SQL is just too cumbersome to handle well"… "RDF (and by extension SPARQL) becomes more important as the data models themselves become more complex, more associational, and more heterogeneous, simply because the variety of information will dominate over factors such as volume or velocity."[9]

Some of the unique advantages of semantics include the following:

- Triples are universally understood and can be easily searched and shared
- Triples connect together to form graphs that are machine readable, and can even be used to infer new facts

9 Cagle, Kurt. "Why SPARQL Is Poised To Set the World on Fire." June 4, 2016. <https://www.linkedin.com/pulse/why-sparql-poised-set-world-fire-kurt-cagle>

SEMANTICS IN PRACTICE

The British Standards Institution, or BSI, publishes standards and provides standards-related services for businesses around the world. BSI developed an online application, British Standards Online (BSOL), for searching standards.

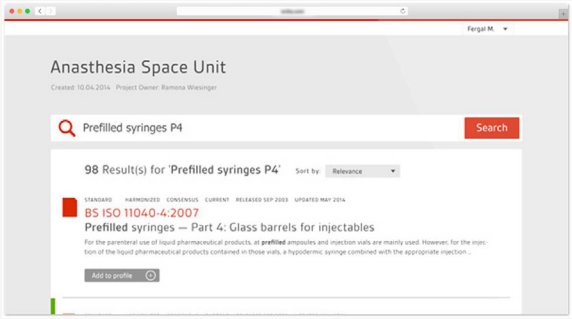


Figure 10: Screenshot of BSI app

Unfortunately, traditional keyword searches were unhelpful. For example, searching for "cardiac catheter" brought back zero results. Google was not helpful either, as a search for "cardiac catheter manufacturing standards" only brought back general links based on the keywords, not the specific, applicable standards.

Using MarkLogic, BSI built a new application incorporating semantics in order to make searching standards better and faster. Semantics makes it possible to expand queries so that a user can just search for "cardiac catheters" and then see results based on conceptual relationships. So, even though a document about medical standards may not specifically mention "cardiac catheter," it may mention "implantable devices" that are semantically related to cardiac catheters.

> “Rather than relying on the relatively abstract and complex ways in which relationships are defined in relational databases, triples describe facts and relationships explicitly in the database.”

- Common standards are defined by W3C for RDF triples and the query language, SPARQL
- Triple stores can scale to hundreds of billions of facts and relationships
- Triple stores can leverage ontologies to organize and categorize data (ontologies are like taxonomies, but are richer and more useful)

THE IMPACT OF A BETTER APPROACH

MANAGING DATA VARIABILITY

The unique features of triples and triple stores are generally put to use by organizations in two ways, all of which are aimed at providing context for data. One approach is to use triples to describe an almost limitless number of facts and relationships about an organization, a domain, or the world at large. This in turn can be used to enhance search.

Having a standard approach to defining facts and relationships helps manage data variability. There is variability in how organizations define entities (e.g., different column names for the same thing, or the same column name for different things). There is also variability in natural language (e.g., the word “sub” may refer to a naval submarine, or to a Subway® sandwich depending on the context).

Rather than relying on the relatively abstract and complex ways in which relationships are defined in relational databases, triples describe facts and relationships explicitly in the database. This is really beneficial for data integration because triples can connect data more easily than through joins or complex transformations. For example, a triple could say the entity `cust123` is the same as `cus_id_456` or is related to that `cus_id_456` in a certain way. Relationships such as class and property relationships can be captured, going beyond what a traditional physical ER diagram shows.

With these features, semantics provides a great data model for expanding searches. For example, it is possible to take a search input, such as “cardiac catheter” and expand that search to include results that have anything to do with “implantable devices” based on the semantic taxonomy or ontology. See the call-out-box above for more information on this type of search expansion.

MANAGING METADATA AND REFERENCE DATA

Triples can be used as metadata or reference data to describe data lineage (provenance and on-going history), data retention and temporality, security, relevance, or a wide variety of other facts about data. Rather than having to go elsewhere in search of the context for data, that context can now be connected to the data in the database. Returning to the earlier example of a database with various sizes for parts, it is possible to use triples to define the units (cm), the tolerance (h17, 0-1.20mm), that John measured it, that the measurement was taken on December 1, 2015, and that only employees in the manufacturing division can see the information.

This approach is superior to the traditional relational approach in which metadata is usually non-existent or difficult to manage, and also has benefits when compared to a solely document-oriented approach. While metadata and reference data can be managed in JSON or XML documents, the document model does not have the ability to leverage the graph structure and machine readability of triples. These benefits, combined with the ability to do semantic inference and describe complex class and property relationships, make semantics ideal for storing metadata. When the data and metadata can be kept together in the same database, integrating data from disparate silos is a much smoother, less error-prone process.

> “Polyglot persistence is best achieved by storing many kinds of data in one place rather that storing many types of data in many places.”

THE MULTI-MODEL APPROACH COMBINING NOSQL AND SEMANTICS

There is no single data model that works well for every use case. By necessity, organizations will require more than one model to support different types of data and workloads. The idea of using multiple models in combination is known as “polyglot persistence.”

We take the position that polyglot persistence is best achieved by storing many kinds of data in one place rather that storing many types of data in many places, an approach that requires bolting together various different technologies. Other database vendors are coming to the same conclusion; and, as one analyst states, the “multi-model approach is the future of NoSQL.”[10]

Multi-model databases allow organizations to get the benefits of each model in a single platform, and get additional unique benefits that are only possible when both models exist in the same database. Currently, MarkLogic is the only enterprise-grade, multi-model database that combines a document database with semantics. In this section, we discuss the benefits of MarkLogic’s approach in more depth, but here is a quick summary:

- Document database (JSON, XML)
  - Flexibility and agility
  - Massive scale
- Triple Store (RDF)
  - Designed for entities and relationships
  - Designed to provide context
- Documents & Triples (JSON, XML, RDF)
  - *Benefits of each of the above, and...*
  - One single platform
  - Even more flexible data modeling
  - Improved query capabilities

10 Aslett, Matt. Toward a converged data platform, part one: SQL, NoSQL Databases and data grid/cache. 451 Research. Dec. 3, 2015.

Figure 11: MarkLogic is an enterprise-grade, multi-model database that makes it possible to store data, documents, and triples all in a single unified platform

SIMPLICITY OF A UNIFIED DATABASE

By combining a NoSQL document database and triple store in MarkLogic, organizations get the benefits of both models without having to maintain separate systems. This alone is a huge benefit. It means drastically reducing the footprint for backup and recovery, development and testing, and search. It also means only having to maintain one security model and has implications for the size of the hardware footprint as well. So, rather than spending the majority of time and resources managing legacy data silos, organizations can focus on value-adding activities.

MORE FLEXIBILITY, MORE POWER

With MarkLogic, organizations can store data as documents or as triples, all depending on the use case. For example, one common pattern already discussed is using triples to expand document searches with semantic taxonomies or ontologies.

When deciding which data type to use, it is easiest to think of JSON as the best data type for objects (e.g., customer, stock trade), XML for text (e.g., blog post, news article), and RDF triples for facts and relationships

	JSON	XML	RDF	JSON/XML + RDF
Usage	Ideal for structured data that is stored as objects	Ideal for structured and unstructured data or text	Ideal for facts and relationships	Ideal for systems of data, text, and relationships
Description	• Schema-agnostic • Query with JavaScript • Compact and fast to parse • Six kinds of values: objects, arrays, floats, strings, booleans, nulls • Avoids namespaces, comments and attributes • Common data format for the web	• Schema-agnostic • Query with XQuery • Can store objects, sets, and many data types such as dates, durations, integers, and more • Uses namespaces (for embedding object types), comments, and attributes (for adding metadata) • More maturity than JSON as a data model	• Define entities and relationships • Atomic structure (cannot be broken down further) • Uses universal standards for data and querying (RDF and SPARQL • Used for reference data, metadata, provenance	• Documents can contain triples • Triples can annotate documents • Graphs of triples can contain documents • Enhanced querying: - Expand a document search using graphs - Enhance graph search by linking to documents - Restrict document search using triples metadata

Table 1: Usage and descriptions of different data models in MarkLogic

(e.g., J.J. Abrams directed Star Wars, a story by George Lucas). Table 1 above has more detail on each data model. With this optionality, organizations can choose the right tool for the job rather than having to deal with the cumbersome constraints that relational databases apply to the increasing volume of multi-dimensional data.

IMPROVED QUERY CAPABILITIES

With a better data model, it is possible to ask harder questions of the data (and usually with less code). There are new kinds of queries that are possible with MarkLogic that are not possible with SQL, and other queries can be written more simply than with SQL.

MarkLogic's document model provides some unique query capabilities such as being able to search the free text within documents and rank data by relevance. MarkLogic's triple store provides the ability to navigate a graph of relationships and do semantic inference. With inferencing, new triples can be added to the graph and then leveraged based on how existing triples are defined and related.

Combining both models together in MarkLogic opens up the door for more types of queries. MarkLogic provides APIs to mix SPARQL queries with traditional MarkLogic document search queries, making it possible to expand a document search using graphs, enhance a graph search by linking to documents, and restrict document searches using triples metadata. With more options for how to ask questions of the data, the better the answers will be.

GET GOING WITH A BETTER DATA MODEL

This paper is only scratching the surface of the benefits of NoSQL and semantics and what is possible when starting with a better data model. Making the move from the old world of data modeling to the new may seem daunting at first, but there are many leading organizations that have already adopted the multi-model approach and are seeing the benefits. Here, we provide some links to more information so that you can get the process started.

> “With a multi-model approach, organizations can choose the right tool for the job rather than having to deal with the cumbersome constraints that relational databases apply to the increasing volume of multi-dimensional data.”

MORE INFORMATION

Enterprise NoSQL for Dummies
po.st/nosqlfordummies
Read the book to get a good general overview of the different types of NoSQL databases and what to look for when evaluating them.

Semantics for Dummies
po.st/semantics
Get a deeper dive into the world of semantics, learning about the various use cases and the top ten things to look out for when considering semantics.

Customer Presentation
po.st/multimodeldataintegration
Hear how one customer analyzes data models, and how his large organization is applying NoSQL and semantics.

Modern Approach to Data Modeling
po.st/modeling
Watch this presentation by MarkLogic engineers that goes into depth about multi-model data integration in the real world.

MARKLOGIC CORPORATION
999 Skyway Road, Suite 200 San Carlos, CA 94070
+1 650 655 2300 | +1 877 992 8885 | www.marklogic.com | sales@marklogic.com