White Paper

embarcadero®

Does Data Modeling Still Matter, Amid the Market Shift to XML, NoSQL, Big Data, and the Cloud?

A companion document for the
O'Kelly Associates webinar

By Joe Maguire and Peter O'Kelly

Reprinted with permission.

Table of Contents

Introduction 3
A Framework for Information Management 3
Relations and Resources: Dichotomy or Continuum? 3
Separation of Concerns: Conceptual, Logical, and Physical 4
Applications, Data, and Application/Data Independence 6
Recap: The Value of an Information Management Framework 6
Database Market Dynamics 7
RDBMS Reconsidered 7
XDBMS for XML Information Management 8
NoSQL 10
Big Data 11
Cloud 12
Database Market Dynamics: Recap 12
Revisiting the Role of Information Modeling 13
Old, Unchanged Realities 13
Separation of Concerns 13
Distinguishing Data from Process 13
Contextual Consensus 13
User-Recognized Metamodels 13
Noteworthy New Realities 14
Data Modeling For Cloud Computing 14
Distinguishing Between Conceptual and Logical Modeling 15
Paradox: Underinvestment in Data Modeling 15
Recap and Recommendations 16
Recommendation: Establish strategic consensus 16
Recommendation: Refocus on Data Modeling 16
Recommendation: Appreciate the Interplay between Relations and Resources 17
Recommendation: Learn and Leverage XQuery 17
For More Information 17

Introduction

During the last few years, several database market dynamics have led many people to question the utility of data modeling[1] . In particular, the advent of XML information management, growing frustration with traditional relational database management system (RDBMS) capabilities and vendor relationships, and the expanding influence of cloud platforms have led many organizations to reconsider their commitments to the practice of data modeling.

Other database market dynamics, including NoSQL and Big Data systems, have also led many people to reassess the value of data modeling. Later sections in this document will explain why we believe the likely roles for NoSQL and Big Data are broadly misunderstood today, but there is no question that market enthusiasm for these loosely-defined domains has challenged traditional data modeling assumptions.

Overall, we believe data modeling is more important than ever before, and that organizations that seek to fully leverage database market dynamics must redouble their focus on data modeling. The rest of this document explains the reasoning behind our perspectives.

A Framework for Information Management

There has been a great deal of information management innovation during the last several years, as organizations seek to exploit new opportunities made possible by developments including the shift to Internet-centric computing and the advent of cloud platforms. We'll review the most significant market dynamics in a subsequent section, after we introduce a framework that can be used to understand how the various market dynamics relate to one another. The framework has a pivotal role in understanding current market dynamics and how they collectively influence the role of data modeling.

Relations and Resources: Dichotomy or Continuum?

The first framework dimension addresses the distinction between two types of information items, resources and relations. Resources are digital artifacts optimized to impart narrative flows (e.g., to share stories), and are usually organized in terms of narrative, hierarchy, and sequence. Examples of commonly-used resource types include books, magazines, documents (e.g., PDF and Word files), and hypertext documents such as Web pages and XBRL (the eXtensible Business Reporting Language) reports. As means of asynchronously sharing stories, resources are optimized for human comprehension.

Relations, in contrast, are application-independent descriptions of real-world things and relationships between things. Examples include popular database domains such as customer, sales, and human resources models. Relations are designed to be used by applications and tools (such as query/reporting tools), and, in the absence of such tools, are rarely useful for conveying stories or otherwise enhancing human comprehension of a given domain.

[1]Data modeling, for the purposes of this document, is not limited to traditional (typically relational) modeling; it is also applicable to domains such as enterprise content management and Web content management. It would be more precise to refer to the broader domain as "information modeling," encompassing both data modeling and a variety of types of content modeling.

Figure 1 depicts the resource/relationship continuum.

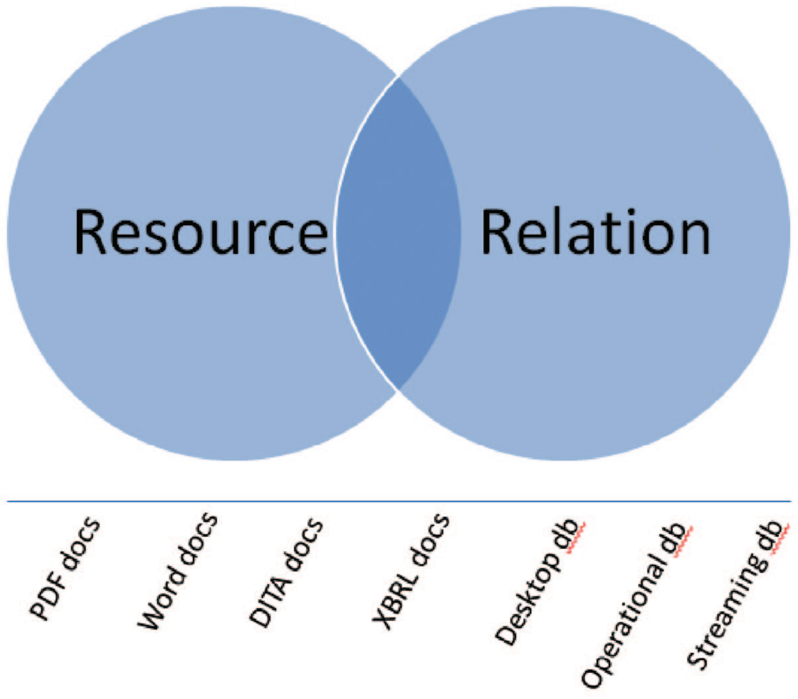


Figure 1: The Resource/Relation Continuum

While the model in Figure 1 is not meant to be precise or exhaustive, it helps to explain the distinctions among different categories of resources and relations.

Resources and relations are complementary. For example, an Internet-based order processing system is likely to include both relation-focused databases and resources such as XML documents capturing purchase order details, along with other document-centric concerns such as digital signatures used for authentication purposes.

Separation of Concerns: Conceptual, Logical, and Physical

The second dimension of our framework addresses three complementary levels of modeling abstraction:

- Conceptual, which is technology-neutral and used primarily to help establish contextual consensus among modeling domain stakeholders. Information workers ideally work at the conceptual model level of abstraction.
- Logical, which captures conceptual models in a technology rendering. The two most widely-used logical models today include hypertext (e.g., Web pages, but often much more elaborate, such as XBRL reports) and relational. Application developers ideally work at the logical level of abstraction.
- Physical, which includes implementation-level details such as indexing and federation. Physical modeling concerns are ideally limited to systems architects and administrators; information workers and application developers should not be burdened with physical model details.

Most organizations currently work with a combination of logical and physical models. Conceptual models are not as widely used or understood today. Figure 2 includes two conceptual model fragment examples, one for Evernote and the other for Microsoft OneNote, two popular personal information management tools.

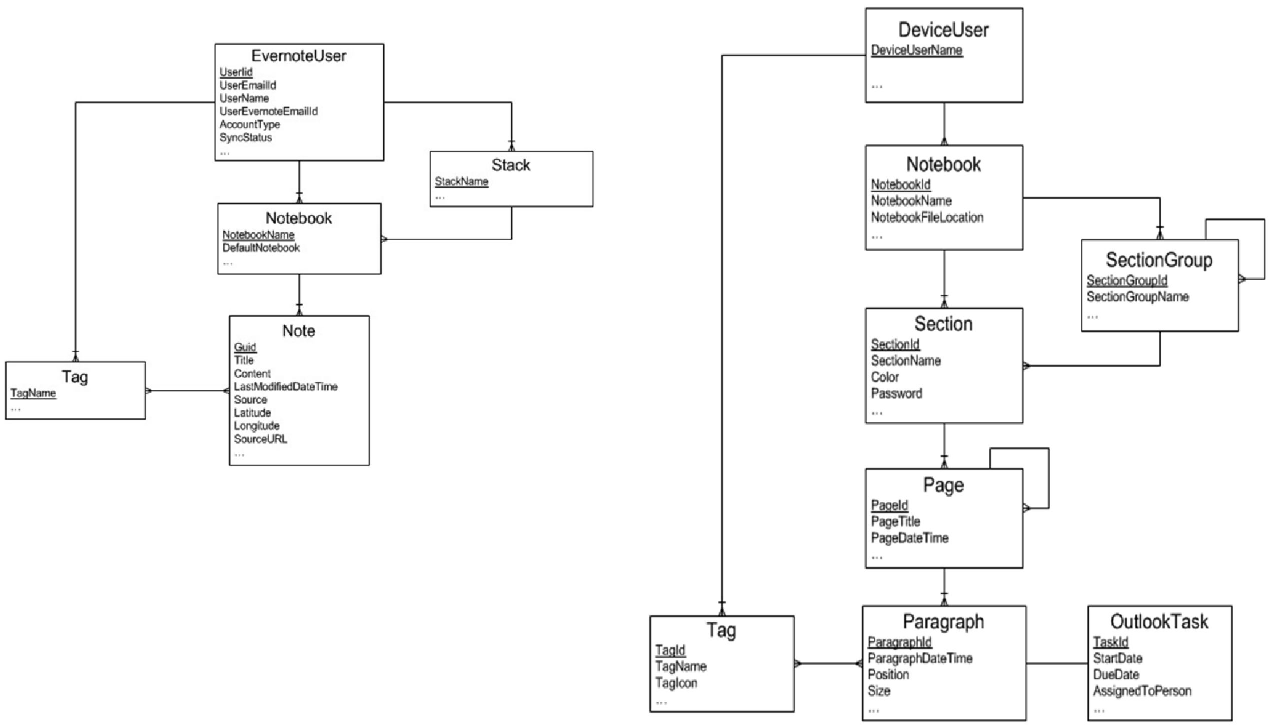
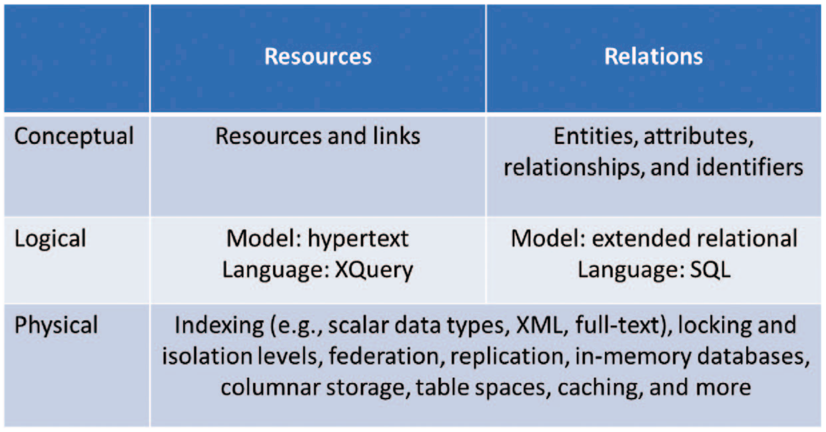


Figure 2: Conceptual Model Diagram Examples

Even without training in the modeling technique, it's easy to observe, looking at the examples in Figure 2, that OneNote has a more elaborate containment model than Evernote, and that OneNote supports tagging at the sub-page/paragraph level, while Evernote tagging is at the note/page level.

While the conceptual model conventions may seem familiar to people who have experience with logical modeling tools, note that they do not include details such as foreign key relationships, data types, or null constraints. Conceptual models are optimized to help establish contextual consensus among business domain stakeholders, so they are focused on entities, attributes, relationships, and identifiers, and do not assume a logical model preference, or include lower-level details that might distract or confuse business domain-focused people.

Figure 3 captures the dimensions in a unified framework.



	Resources	Relations
Conceptual	Resources and links	Entities, attributes, relationships, and identifiers
Logical	Model: hypertext Language: XQuery	Model: extended relational Language: SQL
Physical	Indexing (e.g., scalar data types, XML, full-text), locking and isolation levels, federation, replication, in-memory databases, columnar storage, table spaces, caching, and more	

Figure 3: The Unified Framework

The framework will be revisited in subsequent sections, when we review database market dynamics such as NoSQL and Big Data.

Applications, Data, and Application/Data Independence

A third and final framework dimension that's important to consider is the distinction between applications and data. Data management is not just about information, of course; information without applications or analytical tools to make the information useful is simply archival. However, although information management scope is guided by application requirements, application/data independence is a key goal, to ensure that databases are useful for multiple application domains.

In some database scenarios, application developer preferences can have what we consider to be disproportionate influence on information modeling. For example, organizations that are primarily focused on optimizing programmer productivity may seek to minimize what have been called "impedance mismatches" between programming language frameworks (typically manifesting the object-oriented paradigm) and storage concerns (typically XML and SQL). In extreme cases, this approach can effectively represent a return to the pre-DBMS era, reflecting a programs-have-files mentality that can greatly complicate information management. While paradoxical to database professionals, this approach is a common information management pattern that must be considered when reviewing data modeling concerns, and it's a topic we'll revisit as we review database market dynamics.

Recap: The Value of an Information Management Framework

By working with a framework focused on the complementary nature of resources and relations, and by distinguishing conceptual, logical, and physical models, it's possible to explore how current database market dynamics fit relative to one another. Organizations that opt to not work with such a framework risk:

- Uncertainty about what to use when, in terms of database systems and modeling tools, often resulting in application developers using the techniques and tools with which they're most familiar or comfortable, rather than what might be the best fit for the domain
- Conflict based more on miscommunication and/or misunderstanding that real information-model based issues, due to a lack of contextual consensus

- Insufficient focus on
    - Application/data independence, with the potential to inadvertently revert to the pre-DBMS era programs-have-files approach
    - Conceptual, logical, and physical model independence, which can result in information workers and application developers being counterproductively burdened with lower-level implementation considerations
- Lower probability of appreciation for the sustainable and complementary fit between relational and XML DBMSs, which is central to current database market dynamics

Database Market Dynamics

This section includes a high-level review of current database market dynamics.

RDBMS Reconsidered

The last decade has been challenging for the leading commercial RDBMS vendors. They appear to be under siege in several respects, with widespread IT frustration with RDBMS business-as-usual norms, due in part to sometimes counterproductive RDBMS vendor policies (e.g., unpopular pricing and licensing policies) and customer service. The database administrator (DBA) profession has also been challenged, with many application developers believing DBAs are excessively conservative or control-oriented.

There have also been several challenges to long-held assumptions about RDBMS capabilities, including:

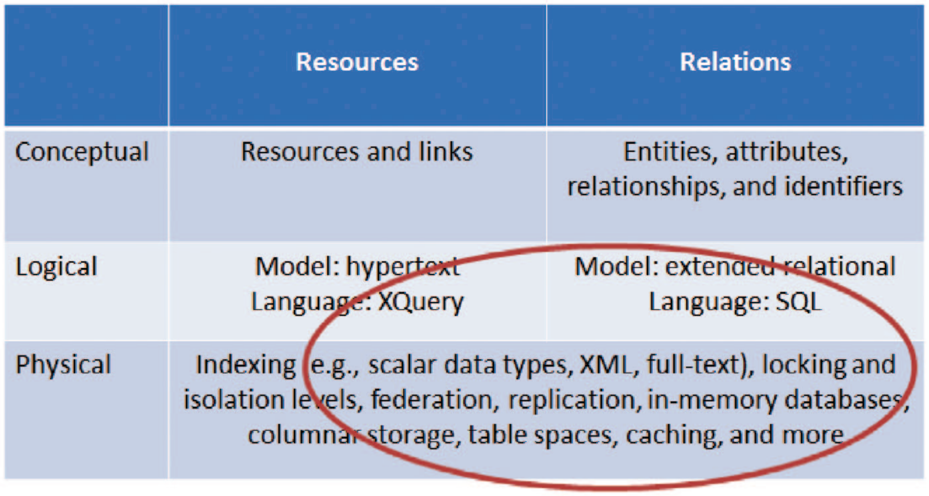
- The suitability of RDBMSs for "Web scale" computing(e.g., as required by Facebook, Google, Twitter, and other planetary-scale services)
- The flexibility or "agility" required to address rapidly-changing application domains
- The application/data "impedance mismatches," for both the long-standing frustration with programming framework/SQL boundaries and, increasingly, with programming framework and XML boundaries as well

These dynamics, along with the advent of open source DBMSs and special-purpose DBMSs (with, e.g., distinct DBMS types being promoted for "Web-scale" analytics and information stream processing), have led some to assert it's "the end of the one-size-fits-all database," i.e., the end of the traditional RDBMS approach.

While there is still ample room for information management innovation, we believe that leading RDBMS commercial products and open source initiatives are very powerful and flexible, and that they will continue to rapidly evolve, e.g., with the mainstream adoption of massive-memory servers and solid state disk (SSD) storage.

However, RDBMS incumbents and advocates nonetheless currently face unprecedented challenges (both technical and political), and those challenges sometimes resonate with frustrated architects and developers because of negative experiences that often have more to do with how RDBMSs have been used rather than what RDBMSs can effectively address.

Figure 4 is a high-level depiction of our perspective on where RDBMSs fit into the framework introduced earlier in this document.



	Resources	Relations
Conceptual	Resources and links	Entities, attributes, relationships, and identifiers
Logical	Model: hypertext Language: XQuery	Model: extended relational Language: SQL
Physical	Indexing (e.g., scalar data types, XML, full-text), locking and isolation levels, federation, replication, in-memory databases, columnar storage, table spaces, caching, and more	

Figure 4: RDBMS in the Unified Framework

XDBMS for XML Information Management

The use of DBMSs for XML information management is another important database market dynamic. XML information management presents several significant opportunities, building on related industry standards such as XML Schema, XPath, XSLT, XQuery, and SQL/XML.

XML-focused information management systems can be useful for addressing both XML hypertext documents (resources) and XML data (relations). For resources, XML systems are primarily useful for displacing earlier, proprietary content/document management systems. For XML data, the systems are useful primarily for scenarios in which XML is to serialize structured data interchange between applications.

Considerable confusion can result from insufficient appreciation for the distinctions between XML resource and relation concerns. Treating all XML information management domains the same way is as unhelpful as the "everything is an object" approach associated with the ill-fated "object database" product wave of the late 1980s, especially when that type of over-generalization is driven by programmer preferences without full consideration of related information management implications.

There are two general approaches to XDBMS (XML DBMS) architectures today:

- Hybrid, multi-model DBMSs, with XML model management added to leading commercial RDBMSs. This is the approach pursued by the leading commercial RDBMS vendors, IBM, Microsoft, and Oracle (indeed, it's a misnomer to categorize the latest releases of their products as RDBMSs, as all three vendors now offer multi-model management systems, but the RDBMS category association won't be easy to change). The vendors have invested very significant research and development resources in their XML capabilities, including seamless SQL/XML and XQuery support.
- Specialized XML DBMSs, exemplified by the eXist open source initiative and the MarkLogic Server commercial product. Modern XML DBMSs are architected for massive scalability on commodity server hardware.

Figure 5 is an example of the hybrid approach (IBM DB2 9.x).

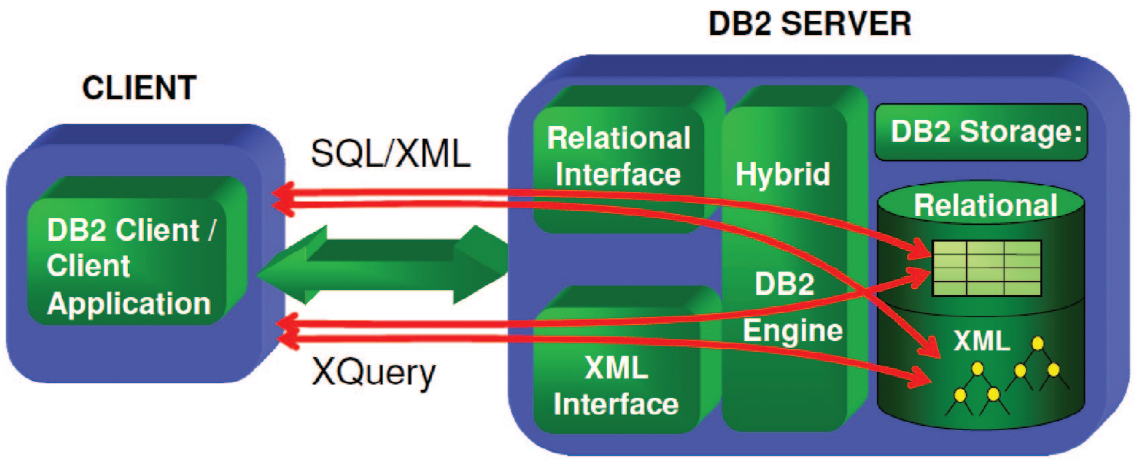
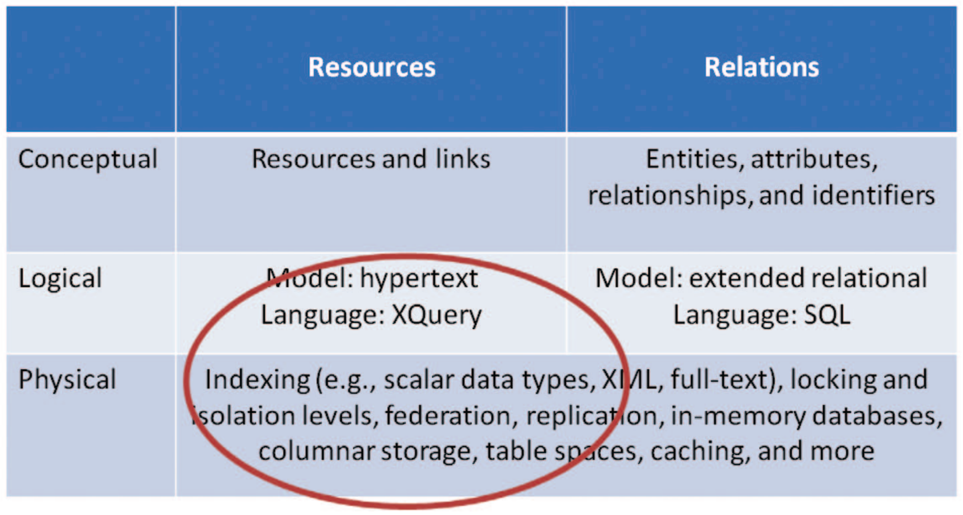


Figure 5: A Hybrid DBMS (source: IBM)

As suggested in the figure, the hybrid approach makes it possible for resource-oriented developers to use XQuery while working with both relational and XML information, while relation-oriented developers can use SQL with XML extensions to work with the same information.

Figure 6highlights where XDBMS fits within the unified framework (hybrid X/RDBMS offerings address the combined areas highlighted in Figures 4 and 6).



	Resources	Relations
Conceptual	Resources and links	Entities, attributes, relationships, and identifiers
Logical	Model: hypertext Language: XQuery	Model: extended relational Language: SQL
Physical	Indexing (e.g., scalar data types, XML, full-text), locking and isolation levels, federation, replication, in-memory databases, columnar storage, table spaces, caching, and more	

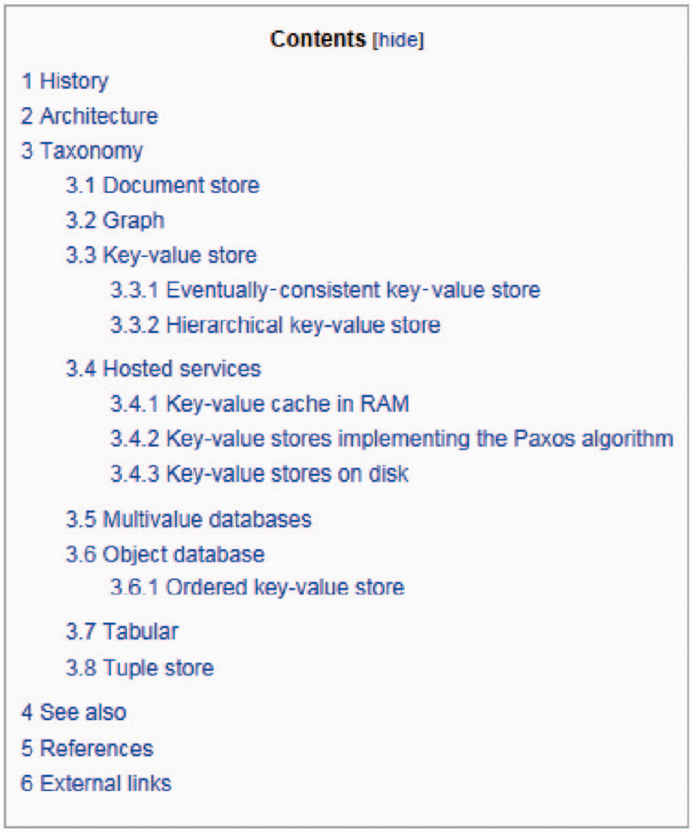
Figure 6: XDBMS in the Unified Framework

NoSQL

NoSQL is a fascinating database market phenomenon, in part because there is no clear consensus on what "NoSQL" means. As an industry initiative, NoSQL starts by defining what it isn't (SQL-based) rather than what it is. Nonetheless, NoSQL advocates can find receptive audiences due to frustration with RDBMS business-as-usual experiences.

It's important to note that NoSQL is also rapidly evolving. It was first received as a movement suggesting "Just say no to SQL!" but has, over the last couple years, been quietly and retroactively redefined as "Not Only SQL," emphasizing more of a complementary than competitive stance relative to RDBMSs.

Figure 7 is a snapshot of the NoSQL system type taxonomy from the NoSQL Wikipedia article.



Contents [hide]

- 1 History
- 2 Architecture
- 3 Taxonomy
  - 3.1 Document store
  - 3.2 Graph
  - 3.3 Key-value store
    - 3.3.1 Eventually-consistent key-value store
    - 3.3.2 Hierarchical key-value store
  - 3.4 Hosted services
    - 3.4.1 Key-value cache in RAM
    - 3.4.2 Key-value stores implementing the Paxos algorithm
    - 3.4.3 Key-value stores on disk
  - 3.5 Multivalue databases
  - 3.6 Object database
    - 3.6.1 Ordered key-value store
  - 3.7 Tabular
  - 3.8 Tuple store
- 4 See also
- 5 References
- 6 External links

Figure 7: The Wikipedia NoSQL Taxonomy

While a review of the eight types of systems identified in the article is beyond the scope of this document, Figure 8 captures our perspective on how the various types map to topics previously discussed in this document.

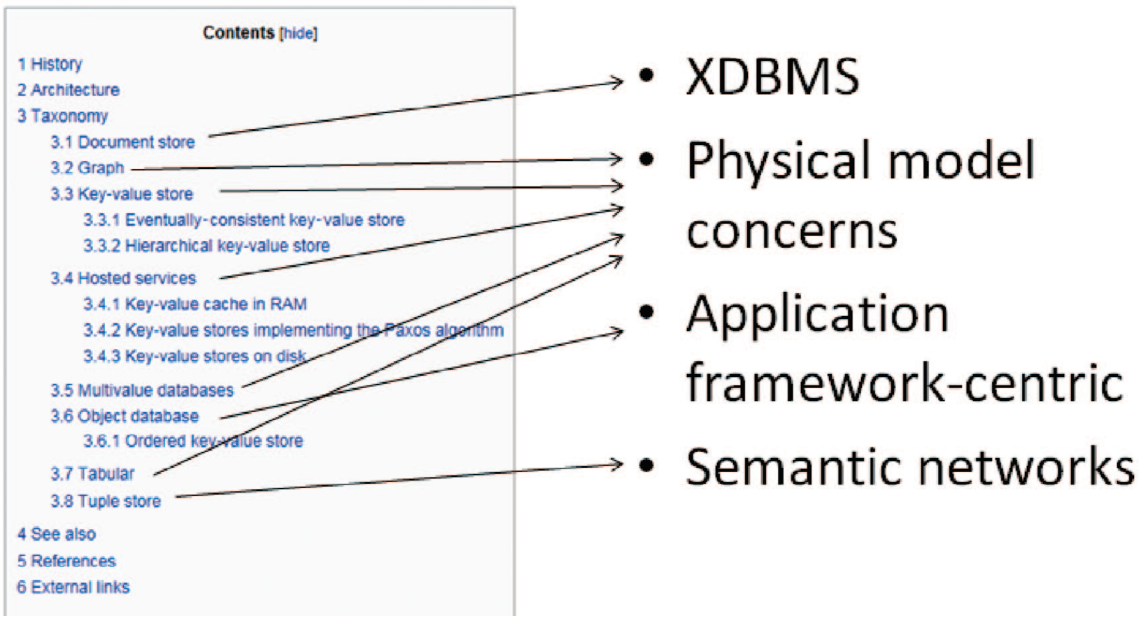


Figure 8: NoSQL Concept Mapping

As suggested in Figure 8, the NoSQL meme confusingly conflates:

- Document store requirements best served by the XDBMS approach (to add more confusion, XDBMS are considered a subclass of NoSQL systems)
- Physical model concerns on which DBAs and systems architects should focus (and which are more complementary with than competitive to RDBMS and XDBMS)
- Object databases, which have been struggling to gain traction for several decades, but for which some application developers are nonetheless enamored, due to a perception of minimized "impedance mismatch" and insufficient focus on the inherent information management compromises
- Semantic models, which are beyond the scope of this document but are, generally, also more complementary with than competitive to RDBMS and XDBMS alternatives

Big Data

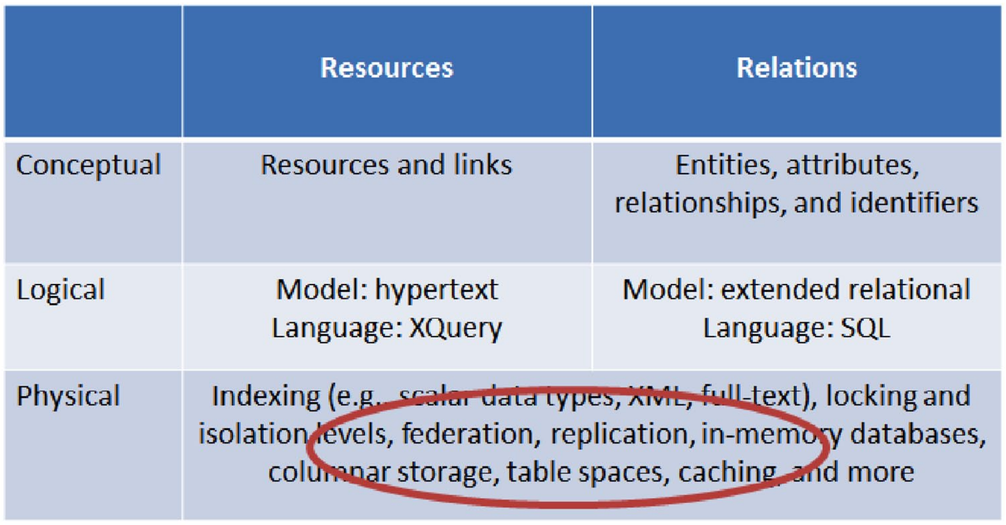
"Big Data" is another imprecisely defined market dynamic, and one that is considered by many people to be a subclass of NoSQL. Big Data is generally associated with application domains including:

- Information stream processing scenarios, e.g., cell phone usage, sensor network, and Web site log analysis
- Massive-scale data analysis, e.g., as Google is applying to refine its visual search and speech recognition techniques

Hadoop is also regularly associated with Big Data. Originating with Google initiatives including its MapReduce and Google File System, Hadoop is an Apache architecture for distributed processing, usually associated with the use of networks of commodity hardware.

Many of the leading vendors focused on data warehousing and business intelligence opportunities are embracing Hadoop as a means to offer more value to their customers. Overall, however, Forrester Research, in a November, 2010 report titled "Stay Alert to Database Technology Innovation," noted that "For 90% of BI use cases, which are often less than 50 terabytes in size, relational databases are still good enough."

Figure 9 captures our view of Big Data relative to the framework presented earlier in this document.



	Resources	Relations
Conceptual	Resources and links	Entities, attributes, relationships, and identifiers
Logical	Model: hypertext Language: XQuery	Model: extended relational Language: SQL
Physical	Indexing (e.g., scalar data types, XML, full-text), locking and isolation levels, federation, replication, in-memory databases, columnar storage, table spaces, caching, and more	

Figure 9: Big Data in the Unified Framework

In other words, we consider Big Data (along with much of the rest of the NoSQL type taxonomy) to be primarily focused on physical model concerns, and more complementary with than competitive to RDBMS and XDBMS approaches.

Cloud

The final database market dynamic within the scope of this document is the advent of cloud database platforms. Composed of a wide variety of RDBMS, XDBMS, and more file system-oriented alternatives, leading cloud database options include:

- Amazon SimpleDB and RDS
- Google BigTable and MegaStore
- Microsoft SQL Azure

Platform-as-a-service offerings generally include multiple database service options.

The creation of cloud database services has produced important database architecture innovation, and much of the innovation is now being applied to future releases of traditional RDBMSs as well. Overall, however, we believe cloud database represents more of a deployment option than a distinct database model alternative. Cloud is also in many cases a financial accounting-driven decision, since cloud computing can be used to shift capital equipment-intensive capabilities into operating expenses.

Database Market Dynamics: Recap

To recap, before shifting to a review of related data modeling considerations, we believe:

- There are substantive, sustainable, and synergistic relationships among:
    - RDBMS
    - XDBMS
    - Hadoop
    - The cloud as an information management platform
- The NoSQL and Big Data market memes are, relatively speaking, ill-defined, transitory, and over-hyped

Revisiting the Role of Information Modeling

Current database market dynamics deserve a response: new realities should induce new behaviors. However, tried-and-true old behaviors should not be discarded haphazardly. Many of the long-standing merits of information modeling will remain valid.

Old, Unchanged Realities

Although the recent and upcoming changes in the information management market are significant and exciting, most of the fundamental realities about information modeling remain unchanged. In some cases, honoring these fundamental realities becomes more important than ever.

Separation of Concerns

Separation of concerns is the effort to compartmentalize design efforts and the resulting artifacts. Separation of concerns distinguishes conceptual, logical, and physical modeling; designers and requirements analysts should distinguish these three forms of modeling because they have different goals, involve different people, and require different skill sets.

Likewise, separation of concerns distinguishes data from process.

Distinguishing Data from Process

It remains crucial for information modelers and requirements analysts to distinguish data requirements from process requirements. Failure to do so has far-reaching negative consequences: compromised application/data independence, premature obsolescence of data models and their underlying databases, and poor information quality caused by low-fidelity data models.

This best practice deserves renewed attention because new realities in information management have introduced new misperceptions. Most notably, some designers have replaced data modeling with the design of message models (e.g., for SOA systems). A message model is first and foremost a processing artifact; it is a manifestation of a portion of a data-flow diagram (which is, after all, a process model). Designing message models is important, but it does not replace data modeling and does not provide the same benefits.

Contextual Consensus

It remains crucial for modelers to recognize the primary value of conceptual data modeling, which is the process of establishing contextual consensus on the meaning of information: what categories exist, what traits apply to those categories, how those categories can interrelate, how the members of any category are distinguished from each other, and what business vocabulary is used to name those categories, traits, and relationships.

This best practice deserves renewed attention because misperceptions about the new realities in information management can compromise (or heedlessly eliminate altogether) conceptual data modeling. Many of the techniques lumped under the NoSQL meme fundamentally alter the nature of physical data modeling. However, changes to the nature of physical data modeling do not necessarily translate to equivalent changes to conceptual data modeling. Conceptual consensus is not a "nice to have," it is essential:

- It is a linchpin of application/data independence—Developers and users of different applications that share data must agree on the meaning of that data
- It provides a foundation for the expression of business rules—The nouns defined on a conceptual data model will serve as the operands of the business rules that formalize processing and policy requirements

User-Recognized Metamodels

Modelers, IT strategists, and software architects must recognize that users are familiar with various conceptual meta-models. Users can even switch from one metamodel to another on an empty stomach; the menu at a typical rural diner includes four metamodels:

- Narrative data – e.g., a quaint history of the diner as founded three generations ago
- Structured data – e.g., the food items, with names, numbers, descriptions, prices, and perhaps even times of day when they are offered
- Image data – e.g., a picturesque etching of the diner as it appeared years ago
- Geospatial data – e.g., a map showing the local roads leading to the diner

System designers do not have the privilege of choosing a convenient metamodel if that metamodel will confound user expectations.

This best practice deserves renewed attention because new realities in information management will break down the artificial walls that have traditionally separated data from content. As those walls come down, data and content will be commingled, largely to the benefit of enterprises. However, such commingling will reward those developers and architects who think carefully about the affinity between user-recognized conceptual metamodels and the developer-chosen implementation metamodels. Likewise, developers who choose metamodels based on implementation convenience, rather than on user needs, will sow the seeds of poor data quality and user confusion.

Note that the user-preferred metamodel will become clear during the requirements analysis phase. In other words, conceptual data modeling cannot work if the modeling notation presumes a particular meta-model.

Noteworthy New Realities

Although most fundamentals about data modeling remain unchanged, the new realities of the database marketplace will have some effect.

Data Modeling For Cloud Computing

Cloud applications might diminish the need for physical data modeling. If an enterprise shifts the burden for database design and operations to an external cloud, that enterprise may no longer needs to make physical design decisions about tablespaces, indexes, and the like.

Note that this does not relieve the enterprise from its conceptual-modeling responsibilities. The use of "database-in-the-cloud" technologies does not obviate the need for contextual consensus. In fact, those organizations that have considered conceptual modeling to be a luxury, deprioritized by the seemingly more immediate need to keep existing databases up and running, might now find that they have time to perform conceptual modeling properly.

Distinguishing Between Conceptual and Logical Modeling

New realities in information management will bring the differences between conceptual and logical data modeling into stark relief.

For better or worse (worse, typically) some enterprises have been using logical-model techniques to collect and articulate user data requirements. This can work, albeit marginally and awkwardly, provided the data requirements align with the logical meta-model. The most commonly used logical metamodels have been close cousins of the extended relational model: Entity-Relationship modeling, the Barker Notation, IDEF1X, etc. Not surprisingly, conceptual data modeling is most typically applied to relations (rather than resources).

New realities in information management, however, allow architects and systems designers to choose logical notations that are optimized for resources rather than relations: e.g., HTML, XML Schema, and their cousins. For non-hierarchical phenomena, such notations will yield low-fidelity data models and incomplete user requirements.

The proper course of action is for requirements analysts to renew their commitment to conceptual modeling techniques and notations that can express requirements for both relations and resources.

Paradox: Underinvestment in Data Modeling

The shift in market dynamics should encourage IT strategists to contemplate more than just the old and new realities of data modeling; it is also useful to think about the old (and new) delusions. Historically, such delusions have resulted in the under-appreciation of and under-investment in data modeling.

In the history of civilization, the bad ideas outnumber the good ones by a wide margin. Data modeling is no different; accordingly, no attempt will be made here to exhaustively list the delusions about it. However, some delusions are especially noteworthy:

- The value of data modeling is frequently underestimated

The consequences of poor or inadequate data modeling are not widely recognized, often because those consequences might not reveal themselves immediately. They become obvious later when, for example, data-quality issues result in erroneous reporting to regulatory agencies and steep fines.

- Data modeling confounds the IT reward structures that focus more on code/process artifacts than on data artifacts

IT strategists and architects will pay sufficient attention to data modeling when IT reward structures encourage them to do so. This includes making data-quality metrics as important as code-quality metrics.

To these longstanding, tried-and-false delusions, we can add some new ones that also discourage the widespread use of data modeling:

- Some activities are inaccurately called data modeling, leaving IT strategists to believe they are doing data modeling, when in fact they are not
- For example, designing an XML schema to accommodate message models is process modeling, not data modeling.
- Some IT strategists erroneously believe that data modeling is unimportant for XML-based information management

Contextual consensus is always important, whether the data is expressed in a relational database, an XML file, or via hammer and chisel on stone tablets.

Recap and Recommendations

It is an exciting time to be involved in information management. New market realities offer solutions to certain problems that have bedeviled IT for decades. Nevertheless, IT experts can continue to leverage many of their tried-and-true best practices.

Amazing improvements in software capabilities, hardware capabilities, and the attendant price/performance ratios will enlarge the size and scope of problems that can be solved by traditional RDBMS's. These improvements include solid-state disk technology and massive-memory commodity servers.

As a deployment option, database-in the-cloud can simplify information architectures by removing some of the moving parts, and by off-loading the operational burden from internal IT organizations. This can free up physical modelers to perform more conceptual modeling (although they will need to enlarge their skill sets).

Another opportunity for simplification: New approaches to coordinating the management of relations and resources can replace many moving parts (web content management, enterprise content management, and file management) with an XML-based data management (XDBMS or a hybrid, multi-model DBMS). In addition, the combination of SQL and XQuery can simplify the application stack, maximize declarative development, and minimize low-level scripting and programming.

Implementations and approximations of Google's Map-Reduce paradigm (most notably, Hadoop) will offer viable and useful frameworks for distributed processing on networks of commodity hardware.

The market meme "NoSQL" will likely disappear, to be replaced by: a) a more disciplined contemplation and deployment of the individual technologies and techniques that NoSQL purports to include, and b) better-informed, calmer acknowledgement that traditional RDBMS technology will remain a foundational workhorse of modern information management.

Recommendation: Establish strategic consensus

In response to the new market realities, enterprises should establish a strategic vision and long-term plan.
This vision should span the entire enterprise, not just IT, because it will affect data governance, information quality, information lifecycle management, security and risk management, and other business concerns.

The plan should establish goals for the effective use and selection of technology. These goals should include the effective decommissioning of some technologies, as when judicious use of XDBMS or hybrid multi-model DBMSs can supplant Web content management, enterprise content management, and file management tools in one fell swoop.

The plan should rearticulate best practices, revising those that no longer apply to the new market realities. The plan should also clarify/improve IT reward structures to ensure that best practices will be honored by programmers, modelers, information architects, and other IT practitioners.

Recommendation: Refocus on Data Modeling

Best practices must include data modeling, and modeling-specific practices must be rearticulated starting from first principles of language, culture, categorization, and technology. Modeling best practices must honor the separation of concerns. IT reward systems need to encourage/require modeling, and IT metrics must expand to reward information quality as well as code quality.

Recommendation: Appreciate the Interplay between Relations and Resources

The new market dynamics will dismantle the artificial wall separating data management from content management. The resulting commingling of relations and resources will offer new efficiencies, but also introduce new opportunities for error. To avoid these errors, IT strategists and practitioners must develop expertise on the interplay between resources and relations. Such expertise begins with scrutiny in the beyond-the-basics metamodels:

- The extended relational model (not merely the model as implemented by your favorite DBMS vendor)
- The hypertext information model (not merely HTML or XML)

Recommendation: Learn and Leverage XQuery

In many cases, software-engineering best practices for managing relations will apply equally well to managing resources. One example: declarative programming will be preferable to procedural code. XQuery is the language of choice for writing declarative queries against XML data. IT practitioners who want to capitalize on the new realities of information management must learn XQuery.

For More Information

For more detailed information about key database market dynamics and their implications for data modeling, information management, and collaboration, including information about workshops and consulting services that can help enterprises fully leverage new opportunities, please visitwww.okellyassociates.com.

For more information about foundational best practices in data modeling starting from first principles of language, culture, categorization and technology, refer to Mastering Data Modeling: A User-Driven Approach, by John Carlis and Joseph Maguire, Addison-Wesley, 2000.

embarcadero®

Embarcadero Technologies, Inc. is the leading provider of software tools that empower application developers and data management professionals to design, build, and run applications and databases more efficiently in heterogeneous IT environments. Over 90 of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero's award-winning products to optimize costs, streamline compliance, and accelerate development and innovation. Founded in 1993, Embarcadero is headquartered in San Francisco with offices located around the world. Embarcadero is online at www.embarcadero.com.