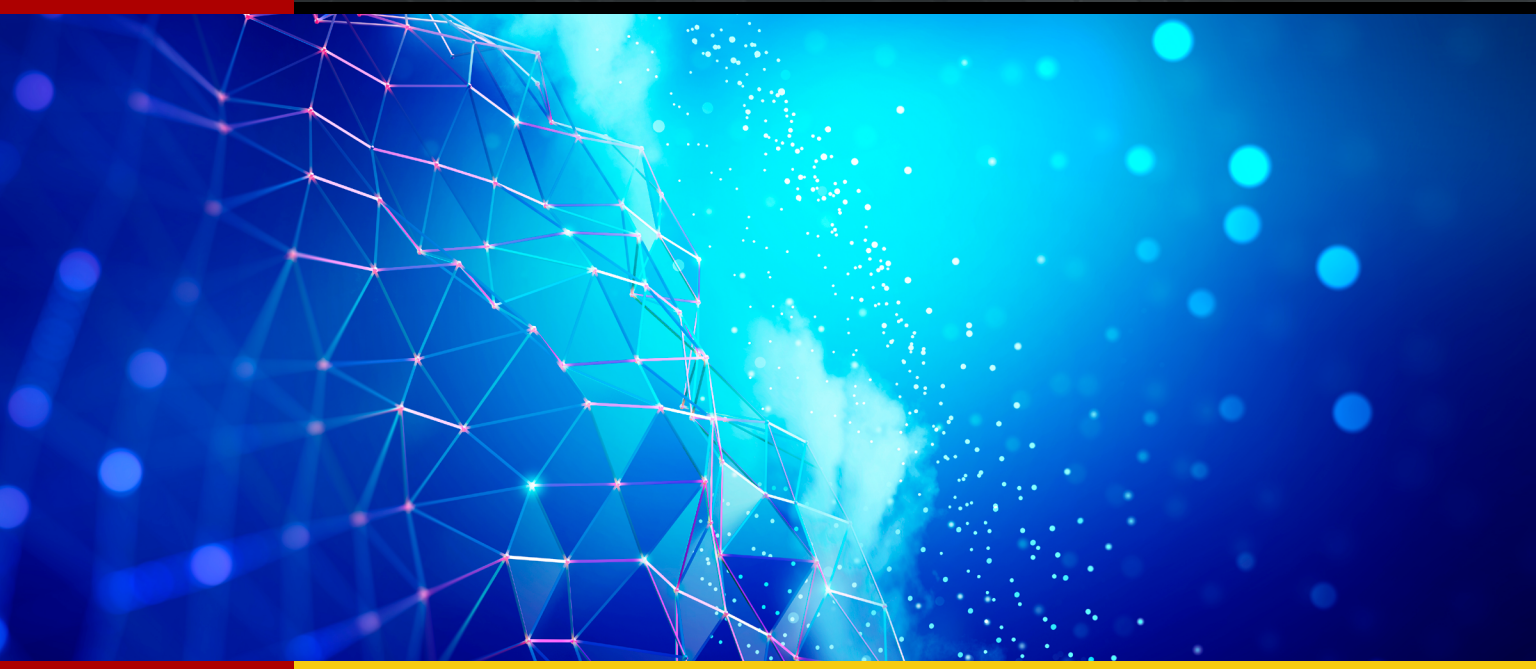
Q3 2021



Ten Mistakes to Avoid

In NoSQL

By James Kobielus

Sponsored by redis

tdwi.org

tdwi
Transforming Data With Intelligence™

Ten Mistakes to Avoid

In NoSQL

By James Kobielus

FOREWORD

NoSQL is a key pillar in many organizations' data architectures. Over the past 20 years, this segment of the data platform market has emerged to support a wide range of requirements for which traditional relational database management systems are not optimal.

NoSQL has gained traction in enterprises to support diverse emerging requirements, including big data analytics, unstructured data sources, low-latency web interactivity, application interconnectivity, data streaming, caching, time-series analysis, and behavioral graph analysis. Increasingly, NoSQL platforms support enterprise DataOps functions such as content ETL. Contrary to popular belief, many NoSQL platforms support query through SQL or SQL-like languages.

NoSQL refers to a wide range of data platform architectures optimized for specific use cases rather than an all-purpose platform for all enterprise requirements. Under the NoSQL umbrella, many industry observers group such approaches as document, wide-column, key-value, and graph databases. What all NoSQL data platforms have in common is support for nonrelational data models, horizontal scaling, and eventual consistency.

This TDWI report identifies the chief mistakes enterprise IT and data managers often make with deployment and operation of these platforms. This report recommends ways DevOps, developers, enterprise IT architects, and data management practitioners can avoid these mistakes in their own NoSQL initiatives.

1 MISTAKE ONE:
APPLYING NOSQL PLATFORMS TO USE CASES FOR WHICH OTHER DATA PLATFORMS ARE BETTER SUITED

NoSQL platforms are application stores for specific use cases, not enterprise data repositories for every application.

Most NoSQL feature sets are designed to meet the demands of online applications, such as personalization, online catalogs, mobile-first applications, and fraud detection. Likewise, NoSQL data platforms are not a full substitute for relational databases in handling strongly guaranteed atomic, consistent, independent, durable (ACID) transactions. Instead, NoSQL platforms tend to compromise consistency in favor of scalability, availability, partition tolerance, and speed.

Nevertheless, NoSQL data platforms are well suited to a wide range of use cases. To avoid choosing the wrong data platform for a specific use case, organizations should assess their requirements for:

- **Interactivity.** The steady rise in user demands for on-demand, real-time, and instantaneous experiences and closed-loop process optimization in all online applications has spurred demand for key-value stores and other data platforms optimized for interactivity. For example, e-commerce and gaming applications depend intimately on interactivity.
- **Latency.** The shift of most use cases toward near-real-time, real-time, and stream processing has raised the importance of in-memory, wide-column (a NoSQL segment), key-value store (another NoSQL segment), and other low-latency data architectures optimized for speed. In-memory data architectures are accelerating queries, interactive exploration, event management, stream computing, and caching.
- **Transactionality.** The intensifying pervasiveness of OLTP, ERP, CRM, and e-commerce applications has highlighted the value of RDBMSs, NewSQL databases, and other platforms that can support ACID transactions. Most NoSQL databases support eventually consistent transactions, under which updates to various replicas will be observed eventually, though some may support strongly eventual consistency (which guarantees that any two replicas that have received the same unordered set of updates will be in the same state). To the extent that a NoSQL platform—such as a document database or wide-column database—supports strong, guaranteed-

Continues

consistent ACID transactions, it is often through a converged architecture, such as a lakehouse, that overlaps with these other segments.

- **Scalability.** The never-ending growth in data volumes, processing throughputs, query concurrency, mixed workloads, and platform decentralization has made scalability a prime criterion in every data platform category. Horizontal scalability—through sharding and eventual consistency—is a prime advantage of NoSQL platforms. The specific scalability profile depends on the particulars of how a database was engineered and deployed, of course.
- **Heterogeneity.** The adoption of unstructured and semistructured data sources, types, and formats has placed a fresh emphasis on document databases (a type of NoSQL platform), distributed file stores, and other data platforms that can flexibly ingest, transform, cleanse, index, search, and manage it all.
- **Temporality.** The ongoing convergence of historical, real-time, and predictive analytics has put the focus on wide-column databases (a type of NoSQL platform), time-series databases, event stores, and other platforms optimized for these workloads.
- **Contextualization.** The need for rich data contextualization has spurred demand for graph databases that are optimized for geographic, social, semantic, behavioral, influence, and other analytical contexts. This is a NoSQL segment.

2 MISTAKE TWO:
IMPLEMENTING A NOSQL PLATFORM WITHOUT THE FORESIGHT TO KEEP IT FROM DEGENERATING INTO A DATA SWAMP

Many NoSQL platforms scale horizontally into petabytes (and beyond) and are optimized for querying, managing, and processing unstructured, semistructured, and other data types.

These features alone might tempt enterprise data professionals to use NoSQL clusters to persist great amounts of random data without a clear application in mind. Rather than let their NoSQL clusters become the proverbial data swamp—which can become prohibitively expensive to maintain—organizations must be clear about what types of data can or should be persisted in their NoSQL clusters, for how long, and with what downstream applications.

To avoid deploying a NoSQL platform that ends up as a costly, underutilized data swamp, enterprises should:

- **Ensure that NoSQL data remains trusted.** NoSQL platforms can serve as powerful collection points for a vast and growing range of data sources. However, business analysts, data scientists, and other users may choose not to use the data in a NoSQL store if they do not trust that data. The best way to keep NoSQL platforms from becoming data swamps is to ensure that new data can be loaded only if it is from a pre-approved source. In addition, all new data can be loaded into a NoSQL platform only if it has been cleansed at the source or completed a rigorous ETL process to confirm it is fit for its downstream purpose.
- **Align NoSQL deployments with data monetization strategies.** Data held in NoSQL stores is a core component of many business decisions, outcomes, and products. NoSQL platforms can easily become costly overhead unless enterprises keep in mind how they figure into a larger data monetization strategy. If every piece of NoSQL-persisted data has a potentially monetizable use, the dreaded swamp scenario is less likely to materialize, and the more useless data will be purged to make way for data that can directly or indirectly generate fresh revenues. Chief among monetizable uses of NoSQL data are driving better operational decision support, improving the efficacy of new AI-based applications, and bundling into new revenue-bearing products. For the same reason, more businesses are providing fresh data for licensing through cloud data

Continues

marketplaces, thereby monetizing these resources beyond any bottom-line payoff from operational uses.

- **Move, archive, and purge less-frequently accessed data from in-production NoSQL platforms.** NoSQL data should be managed like any data resource so it is moved out of the data lake to archives and even purged when it is no longer needed for production applications. If the retention of every piece of NoSQL data depends on its relevance to users, the platforms are less likely to become unmanageable swamps. Enterprise data professionals should apply multitemperature storage management to data maintained in NoSQL platforms just as consistently as they do with structured, system-of-record data kept in RDBMSs. Doing so may require acquiring new integrated life cycle management tools that work with an enterprise's NoSQL platforms. It will also require clarifying which, if any, NoSQL data domains are "ephemeral" (in the sense that there may be no need to retain the bulk of the data in permanent systems of record or in data lakes).

3 MISTAKE THREE:
PUTTING AN UNSUPPORTED NOSQL OPEN-SOURCE DISTRIBUTION INTO PRODUCTION

Most NoSQL platforms are available as open-source distributions; most are also available with commercial licenses that provide enterprise-grade reliability and support as well as subscription-based, on-demand offerings from cloud providers.

Deploying a purely open-source NoSQL distribution is problematic because although it is "free" from licensing fees, the requisite tools administrators need to configure, scale, monitor, manage, and ensure high availability of these platforms are primarily available through commercial licenses. Putting an unsupported NoSQL open-source distribution into production is a one-way ticket to failure if you have not factored ongoing support costs and staffing requirements into your budget.

To avoid betting your business on an unsupported NoSQL open-source platform, data and IT professionals should follow these steps.

- Subscribe to a fully managed NoSQL cloud service.
- Acquire NoSQL open-source support as part of an enterprise software subscription offering.
- Sign up for NoSQL open-source enterprise licenses that include phone and email contacts, guaranteed response times, 24/7 support, and guaranteed resolution times.
- Put your organization's exact NoSQL support requirements out for tender rather than choose a packaged offering from a company that supplies enterprise support.
- Certify your own data and IT technical staff in the NoSQL platform and support it in-house, exploiting free support resources such as mailing lists, developer forums, live support chat, and extensive documentation libraries.
- Get support from a systems integrator or IT consultant, if they are also responsible for the part of your infrastructure running open-source software.

4 MISTAKE FOUR:
APPLYING A NOSQL DATA MODEL UNSUITED TO A PARTICULAR APPLICATION

NoSQL's versatile embrace of diverse data models is also a data management mistake waiting to happen.

Savvy professionals choose the right data model for the application domain of interest. For example, key-value stores are well suited for keeping track of simultaneous web, mobile, and other interactions that involve thousands of users, such as in online gaming and collaboration applications. They are well suited for personalizing and optimizing user experiences. However, they are *not* particularly suited to storing and retrieving large, unstructured data files, a use case for which document databases are a better choice.

To avoid applying a NoSQL data model that is not optimal—in other words, too slow, inefficient, and complex for a particular application—enterprise data professionals should adopt a NoSQL platform that supports multiple data models. Fortunately, multimodel data platforms have become mainstream. More NoSQL platforms are architected with dedicated engines that support various models—such as key-value, search, time-series, and graph—rather than simply through evolution of their APIs.

When the underlying engines are evolved to support various models, the NoSQL platform can achieve superior performance and low latency when processing requests regardless of which model an application speaks. This approach enables each engine and its associated data structures to be optimized for various use cases. The engines can be selectively loaded into memory according to use case. All engines can access the same data, eliminating the need to store multiple copies of the same data or to incur the overhead of transferring data between engines. This approach also facilitates NoSQL processing in a microservices environment so the engines can efficiently communicate state, events, and data with each other.

5 MISTAKE FIVE: INTRODUCING NOSQL SILOS INTO ENTERPRISE DATA ARCHITECTURES

NoSQL databases are purpose-built and optimized for various use cases. Chief among these are real-time applications; edge, mobile, and embedded applications; customer experience optimization; content management; caching; and archiving.

NoSQL platforms persist diverse data types that are essential to these and other applications. What they generally do not do is manage the tabular "golden record" data that is the heart of transactions and other core business applications. Generally, NoSQL applications are unsuited to such business applications as online transaction processing, online analytical processing, data warehousing, and structured data analysis.

To avoid the mistake of deploying purpose-built NoSQL platforms as silos within enterprise data architectures, IT professionals should:

- **Adopt a multimodel NoSQL platform that supports a wide range of use cases.** Enterprise data professionals should explore the growing range of multimodel data platforms that converge the data engines associated with previously distinct NoSQL segments (key-value store, document, wide-column, graph), relational, file, object, and other data platforms. Chief among these emerging segments are lakehouse platforms that converge relational and NoSQL to support unification of data warehousing and data lake workloads; multimodel platforms that converge relational, document, graph, key-value store, and even hyperledger/blockchain; and NewSQL databases that converge NoSQL with relational/ACID.
- **Integrate NoSQL platforms as purpose-built zones within multiplatform data architecture.** Enterprise IT professionals should incorporate their NoSQL silos into an enterprise data architecture alongside relational, file-based, and other databases. This involves implementing a common data catalog, data engineering pipeline, data governance backbone, data virtualization middleware, and other platform capabilities as unifying service layers. Bridging NoSQL and other data silos among on-premises and public clouds often involves deploying hybrid and multicloud integration fabrics.
- **Migrate NoSQL instances to fully managed public clouds.** Subscription-based NoSQL database-as-a-service offerings are gaining traction. Public cloud

Continues

providers offer SaaS-based access to their own NoSQL and other databases, as well as hosted instances of many of the leading NoSQL enterprise products.

One silo-busting strategy is for enterprises to migrate NoSQL data and workloads to these cloud-based environments, relying on cloud providers to bridge them with other databases (on-premises and public cloud) through their own on-demand data integration, governance, and other services. Alternatively, enterprises should consider migrating their on-premises NoSQL silos to the multimodel, on-demand databases offered by more cloud providers, many of which offer multimaster global replication (which allows data to be stored by a group of computers and updated by any member of the group).

6 MISTAKE SIX:
FAILING TO RECOGNIZE THE SCALABILITY CONSTRAINTS OF A PARTICULAR NOSQL PLATFORM

NoSQL platforms boast their ability to horizontally scale data. However, given the variety of NoSQL architectures on the market, it is prudent to assume they do not all have the same scalability profiles.

Scaling of NoSQL databases, as with any software, is constrained by the capacity of the compute, memory, storage, bandwidth, and other hardware resources. It's also constrained by the architecture of the software itself, especially by its ability to distribute, parallelize, and dynamically provision resources to the various functional components of the database software stack.

To avoid inadvertently hitting one's head on the scalability constraints of a NoSQL database platform, IT professionals should:

- **Perform NoSQL database benchmarking.** Consider the scaling and performance constraints of various NoSQL platforms. Benchmarking a NoSQL platform requires identifying the specific NoSQL software distribution, its deployment platform and configuration, the use cases to be supported, the expected normal and peak workloads, and the performance metrics (e.g., throughput, latency, query response times, concurrent writes and reads, join performance) to be measured. Also factor into the benchmark the extent to which the platforms being evaluated use such performance-impacting features as indexing, ACID transactions, object-level caching, asynchronous replication, data persistence, and dynamic auto-scaling of resources depending on workload. Given the fact that every one of those factors is a variable, enterprise IT professionals will need to either benchmark the alternatives themselves or rely on credible third-party benchmarks when assessing whether a given NoSQL platform deployment can scale sufficiently for their intended purposes.
- **Consider how a NoSQL platform trades off scalability against other database operational metrics.** When assessing the scaling limits of any NoSQL database, IT professionals should keep the CAP Theorem uppermost in mind. The acronym refers to three core database engineering operational metrics: consistency, availability, and partition tolerance. The theorem states that one can only guarantee at most two but not all three of these metrics in any

Continues

deployment of a database. Considering that scaling a NoSQL database depends on horizontal partitioning of data sets, its operational integrity must always be engineered for partition tolerance, which guarantees that the partitioned database continues to operate despite arbitrary message loss or failure of part of the system. Consequently, the process of horizontally scaling a NoSQL database across separate servers or other nodes must always introduce a corresponding hit to the consistency of database writes (i.e., the guarantee that all database nodes see the same updates at the same time), the availability of database nodes (i.e., the guarantee that every database request receives a response about whether it was successful or failed), or both.

- **Assess the scalability impacts of different NoSQL replication, sharding, and consistency schemes.** As previously mentioned, NoSQL databases scale horizontally by distributing data sets across multiple nodes, servers, and clusters. This involves—at the very least—robust replication of partitioned database instances to multiple nodes. When benchmarking the scalability of different NoSQL platforms, it's important to consider the number of nodes, the replication and partitioning schemes employed, the manner in which applications access and manipulate data in the partitions, and the level of consistency (strong, strongly eventual, or eventual) guaranteed on distributed database writes. It also depends on whether and how the NoSQL platform employs sharding, which splits large, partitionable tables across the servers, thereby enabling database distribution over a large number of machines for improved performance.

7

MISTAKE SEVEN:
INTRODUCING POTENTIAL NOSQL BOTTLENECKS INTO DATA-PROCESSING PIPELINES

NoSQL platforms are often built to ingest, load, and persist a huge volume and variety of data types. Consequently, the configuration of NoSQL clusters may overprovision storage capacity while skimping on the compute, memory, and bandwidth resources necessary to support acceptable performance on downstream access, query, and analysis of the stored data.

Within a multiplatform data engineering pipeline, NoSQL platforms may have insufficient resources to handle ETL workloads on the volume of data being ingested or to handle batch and real-time workloads. These resource imbalances can introduce bottlenecks that impede delivery of aggregated, conformed, cleansed data to data warehouses, analytics applications, and other consumers.

To avoid creating resource imbalances in their NoSQL deployments, enterprises should adopt database platforms that do the following:

- Support elastic scaling of compute, storage, memory, and bandwidth resources as data workloads and requirements change so bottlenecks can be avoided through on-demand resource provisioning.
- Provide the flexibility to run batch and real-time workloads anywhere—including on premises, in one or more public clouds, or in a hybrid public/private cloud—so jobs can be moved away from potential bottlenecks to nodes with the requisite availability, performance, and scale.
- Leverage data replication with sharding, thereby enabling geographic distribution, horizontal scaling, high availability, disaster recovery, and load balancing across multiple, separate database server instances so no instance can become a bottleneck or single point of failure.

8 MISTAKE EIGHT:
CONFIGURING RESOURCE IMBALANCES INTO NOSQL DATABASE CLUSTERS AND SERVERS

NoSQL's powerful processing comes from being deployed on scale-out clusters and relying on sharding to horizontally partition large data sets out to diverse servers and geographies.

As NoSQL clusters are scaled out over the years, the individual servers added to these clusters may vary widely in specifications, capacity, and capabilities. This may result in resource imbalances if, for example, NoSQL clusters are processing compute-bound workloads on older nodes that have single-core CPUs while multicore CPUs on newer nodes sit idle. Likewise, NoSQL clusters that mix nodes of various specifications may not process I/O-bound workloads efficiently if they are executed on older nodes with rotating disk rather than being sent to newer nodes configured with solid-state drives.

To avoid configuring resource imbalances into NoSQL database clusters and nodes, enterprises should adopt platforms that:

- Scale to thousands of database instances in support of high-volume concurrency across disparate workloads
- Provide on-demand scaling of compute, storage, memory, and bandwidth resources within and between every database cluster and node
- Automatically rebalance and re-shard data workloads horizontally across clusters and nodes to guarantee high-throughput, low-latency, real-time performance
- Leverage sharding so database operations can tap into the full computational power of multicore processing platforms
- Offer dynamic workload management to support traffic peaks without the need to scale database infrastructure
- Use multitenancy to allow multiple database endpoints to run in a single cluster, thereby maximizing infrastructure utilization, enabling endpoint-level database optimization, avoiding performance degradation, and safeguarding database security
- Use intelligent storage tiering so that frequently used "hot" data stays in memory while less-frequently used "cold" data goes on flash, SSD, rotating storage, or tape, with the system automatically balancing and otherwise managing where data is physically persisted

Continues

- Run test, development, and production databases on separate tenants, nodes, or clusters, providing workload isolation and enabling separate provisioning, management, and optimization to meet their various scaling, performance, and availability requirements

9

MISTAKE NINE:
USING A NOSQL PLATFORM FOR MISSION-CRITICAL APPLICATIONS WITHOUT PROVISIONING STRONG DATA GOVERNANCE

NoSQL platforms persist data that may need to be governed for quality, compliance, or other reasons.

Enterprises' NoSQL platforms are ingesting a dizzying range of new data types from mobile devices, edge applications, intelligent robots, and other endpoints connected to the cloud. As NoSQL platforms ingest a wider range of structured, semistructured, and unstructured data that is then delivered into enterprise applications, the legal, regulatory, and other governance-related risks grow.

NoSQL databases' broad adoption is also a potential risk factor. Enterprises' shift toward self-service business analytics has heightened the risk that NoSQL-persisted data may fall into the wrong hands or, even when accessed by authorized users, may be employed for inappropriate ends and be exposed to misuse by unauthorized third parties. Likewise, the advent of citizen data scientists leveraging NoSQL platforms as shared data lakes has increased the risk that users will assemble rogue data sets that are inaccurate, inconsistent, and entirely outside the oversight of enterprise IT professionals.

To avoid creating data governance issues when deploying NoSQL for these and other use cases, organizations should:

- Update their data governance and curation practices to ensure that new unstructured, semistructured, and other data persisted in NoSQL platforms is kept correct, current, conformed, and compliant
- Migrate legacy data governance infrastructure so it integrates tightly with the NoSQL platforms to which enterprises have migrated core business data
- Deploy cloud-based data governance infrastructure that includes scalable data cataloging, data lineage, metadata management, master data management, and data quality management tooling
- Leverage cloud-native platforms to orchestrate data governance workflows across disparate source systems, processing pipelines, stewardship tasks, and production applications
- Automate the discovery, matching, merging, and correction of disparate customer, product, and other data sets persisted in NoSQL platforms

Continues

- Set access and permission controls on which NoSQL data sets may be used by data scientists when building and training their machine learning models or by developers when building their own applications

Until enterprises fully migrate their data governance investments to the cloud, they risk introducing inaccuracies, inconsistencies, and noncompliant changes across data assets scattered across silos associated with various data stores. Just as important, they risk hitting their heads on the scaling of data governance workloads until they move these investments to NoSQL platforms that enable flexible scaling of compute and storage resources as those requirements grow.

10

MISTAKE TEN:
MAKING NOSQL A PILLAR OF ENTERPRISE DATA ARCHITECTURE WITHOUT FIRST TRAINING IT AND DATA MANAGEMENT STAFF

NoSQL platforms may be unfamiliar to most of the people who deploy, manage, and build relational database applications.

It's a big mistake to introduce any NoSQL platform into an IT or data management organization without first upgrading the skills of existing staff and recruiting people with the right mix of competencies. Likewise, it would be a blunder to accept someone's claim to be an expert in NoSQL when, in fact, NoSQL is a vague term that applies to such disparate data platform architectures as document, key-value, wide-column, and graph databases.

To avoid the mistake of rolling out NoSQL platforms in advance of upgrading technical staff expertise, organizations should:

- Explore both commercial and free online courses that can familiarize your data staff with NoSQL concepts, architectures, practices, platforms, and solutions
- Immerse technical staff in the fundamentals of NoSQL databases, starting with the fact that they are nonrelational database management systems that support flexible schemas, provide eventual consistency, and use sharding for horizontal scalability
- Train data architects to recognize both the advantages and limitations—in scalability, performance, concurrency, availability, transactionality, security, optimization, and fault tolerance—of various NoSQL data architectures
- Certify staff in both the designated NoSQL data architectures (i.e., document, wide-column, key-value, graph) to be deployed and the specific commercial or open-source NoSQL platform to be adopted
- Leverage open-source communities that use NoSQL databases to augment the talent in your organization

SUMMARY

NoSQL platforms are a fundamental component of enterprise data infrastructures. Given the business stakes of succeeding with NoSQL platforms, and the risks of failing to use best practices, enterprise IT and data professionals should heed the guidance TDWI has presented in this report.

The chief takeaways for enterprise data professionals presented here include:

- Keep NoSQL deployments aligned with business imperatives
- Apply NoSQL platforms to use cases for which they are best suited
- Only put supported NoSQL open-source distributions into production that have enterprise-grade features such as scalability and high availability
- Apply NoSQL data models that are best suited to a particular application
- Make sure that NoSQL platforms are fully integrated into enterprise data architectures
- Recognize the scalability constraints of each NoSQL platform
- Remove potential NoSQL bottlenecks in data-processing pipelines
- Provision balanced NoSQL database clusters and servers
- Implement strong data governance in NoSQL platforms
- Upgrade IT and data management staff's expertise in NoSQL

Finally, although many NoSQL data platforms are purpose-built, many are well suited to a wide range of use cases and should not be excluded from consideration without evaluating their full merits. For example, key-value stores—an important NoSQL platform architecture—are suitable for requirements such as these:

- **Interactivity** is a sweet spot for key-value store databases, which are optimal for any use case for which reducing interactive response times or closed-loop process latencies is the key to boosting customer experience and business outcomes, such as retail mobile applications.
- **Latency** is another advantage of key-value store databases, which are optimized for cloud-to-edge use cases that require real-time caching, robust replication, message

Continues

brokering, and customizable on-disk persistence. They use simple operational commands such as GET, PUT, and DELETE, which makes them efficient at processing constant streams of read/write operations when path requests used are short and direct.

- **Scalability** enables key-value stores to process a constant stream of read/write operations. They scale up by maintaining the database in memory; scale out through partitions, sharding, and replication; and minimize the scaling constraints of ACID guarantees by avoiding locks, latches, and low-overhead server calls.
- **Heterogeneity** is intrinsic to key-value store databases, many of which support multimodel data sets and process heterogeneous data as associative arrays in which an arbitrary string (such as a hash or filename) represents each key and the values can be any data that can be stored as a binary large object.
- **Temporality** is built into a key-value store through its ability to store time-stamped data as a hash table or dictionary.
- **Contextualization** is within the wheelhouse of key-value store databases, which treat graph data and rich metadata as a single opaque collection and which may have different fields for every record.

ABOUT THE AUTHOR

James Kobielus is senior director of research for data management at TDWI. He is a veteran industry analyst, consultant, author, speaker, and blogger in analytics and data management. He focuses on advanced analytics, artificial intelligence, and cloud computing. Kobielus has held positions at Futurum Research, SiliconANGLE Wikibon, Forrester Research, Current Analysis, and the Burton Group, and also served as senior program director, product marketing for big data analytics, for IBM, where he was both a subject matter expert and a strategist on thought leadership and content marketing programs targeted at the data science community. You can reach him by email (jkobielus@tdwi.org) on Twitter (@jameskobielus) and on LinkedIn (https://www.linkedin.com/in/jameskobielus/).

ABOUT REDIS

Businesses are more digital today than ever before. They need to build, deploy, and run real-time services in order to stay ahead of the curve. The notion of real time is not just a nice-to-have feature anymore. It's an expectation. It is what sets a merely good user experience apart from a great one. A real-time data layer is a critical enabler in creating those real-time experiences.

Redis makes apps faster by creating a data layer for a real-time world. We are the driving force behind Open-Source Redis, a popular in-memory database, and commercial provider of Redis Enterprise, a real-time data platform.

Redis Enterprise powers real-time services for over 8,000 organizations globally. It builds upon the simplicity and speed of Open-Source Redis along with an enterprise-grade data platform that offers robustness of modern data models, management, automation, performance, and resiliency to deploy and run modern applications at any scale from anywhere on the planet.

ABOUT **TDWI**

TDWI is your source for in-depth education and research on all things data. For 20 years, TDWI has been helping data professionals get smarter so the companies they work for can innovate and grow faster. TDWI provides individuals and teams with a comprehensive portfolio of business and technical education and research to acquire the knowledge and skills they need, when and where they need them. The in-depth, best-practices-based information TDWI offers can be quickly applied to develop world-class talent across your organization's business and IT functions to enhance analytical, data-driven decision making and performance. TDWI advances the art and science of realizing business value from data by providing an objective forum where industry experts, solution providers, and practitioners can explore and enhance data competencies, practices, and technologies. TDWI offers major conferences, topical seminars, onsite education, a worldwide membership program, business intelligence certification, live webinars, resourceful publications, industry news, an in-depth research program, and a comprehensive website: tdwi.org.

Transforming Data With Intelligence™

A Division of 1105 Media
6300 Canoga Avenue, Suite 1150
Woodland Hills, CA 91367

E info@tdwi.org