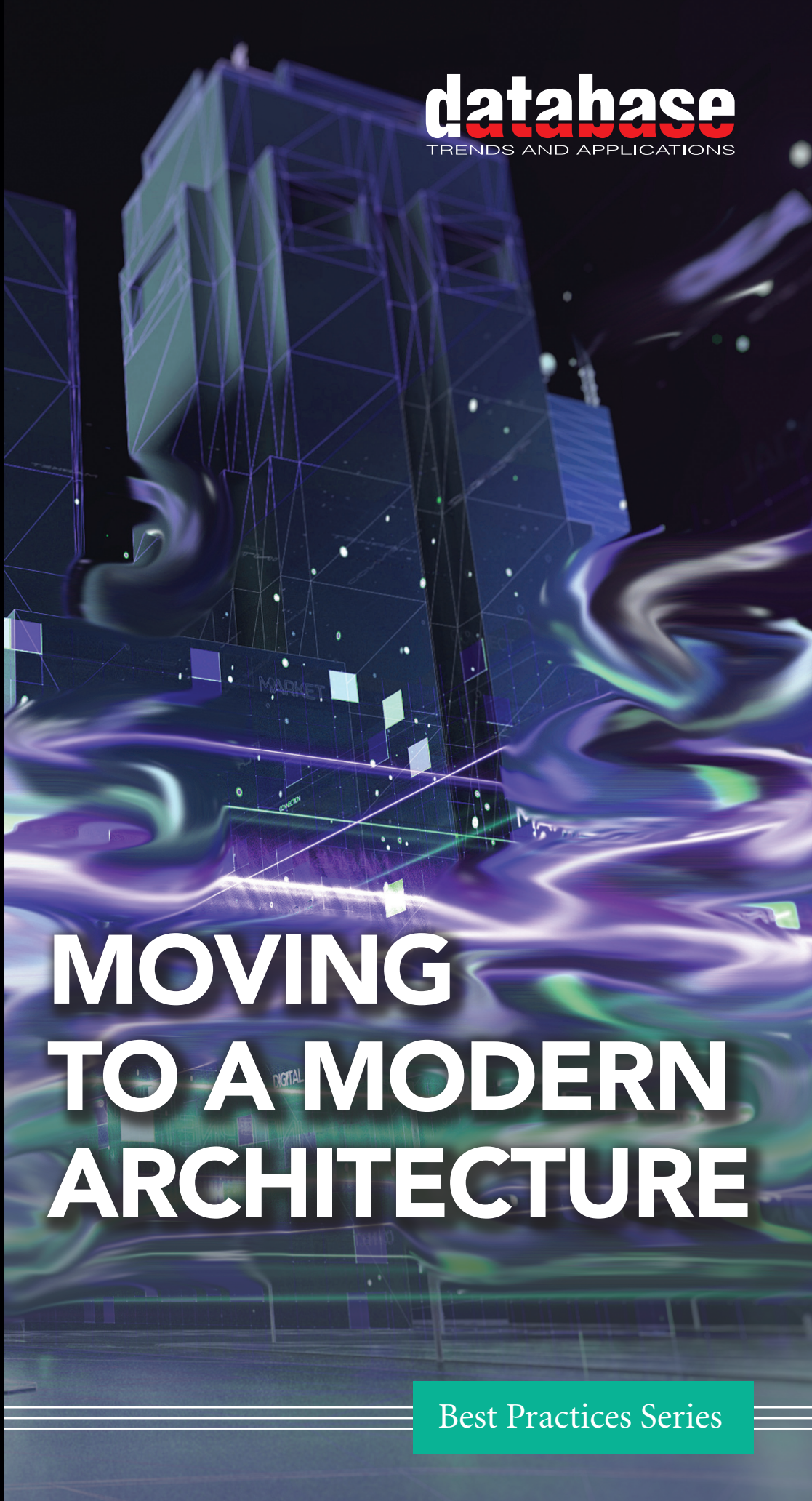
DataStax
Page 10
DESIGNING A FUTURE-PROOF DATA ARCHITECTURE

database
TRENDS AND APPLICATIONS



MOVING TO A MODERN ARCHITECTURE

Best Practices Series

database
TRENDS AND APPLICATIONS

EIGHT TRAITS OF A SUCCESSFUL MODERN DATA ARCHITECTURE

Best Practices Series

The database world is changing fast, and the coming decade—the 2020s—will mean even more change. Open source solutions and frameworks have become the norm. More services and data now reside in the cloud. Today's enterprises are looking to data managers to be able to respond to business challenges with scalable and responsive systems that deliver both structured and unstructured data—and accompanying insights—at a moment's notice, with the ability to respond to any and all queries. What's needed is a modern data architecture that is built on flexible, modular technology, either from open source frameworks and software or through cloud services. Here are eight key ways to prepare for and manage a modern data architecture:

FOCUS FIRST AND FOREMOST ON CUSTOMER EXPERIENCE.

All data management initiatives should ultimately focus on one thing: serving the customer or end user. In today's digital economy, competitive advantage comes from delivering a superior customer experience, and a superior experience is only possible with enough of the right data at the right time. Data architects need to work with business owners to determine customer requirements, and work backward from there.

ENSURE DATA PORTABILITY.

The goal of a modern data architecture is to abstract all underlying infrastructure—servers, storage, and networks—from the applications and data that are being used. Thus, data should be capable of being moved expediently between platforms with little-to-no disruption to the end user. This may be from an on-premise system to the cloud, from cloud to cloud, or from the cloud back to an on-premise environment. While cloud platforms offer many compelling advantages to data managers and developers, they also pose the risk of vendor lock-in by potentially making it too costly and messy to attempt to switch providers. It may not even be possible to extract databases from cloud providers. That's why data portability needs to be an element of a modern data architecture—data and applications will inevitably be moved to the platform that makes the most sense to the business. Just

as important is enabling data analytics and storage anywhere it is requested. Gone are the days when data was stored in a central data center in storage arrays, to be made accessible to end users within the walls of the enterprise. Today's users may be accessing data and applications from anywhere in the world.

BUILD A DATA SERVICE LAYER.

Fundamental to data architecture success is a data infrastructure that is agile and flexible. Data and insights should be available to end users at a moment's notice. A modern data architecture should support a highly networked business, marked by a continuous flow of information across all boundaries and organizational walls. Establishing a standardized interface that can access all applications and data brought into the infrastructure is the essence of a modern data architecture. This helps to meet the goals of data portability, mentioned earlier, while enabling end users to quickly gain access to information from all parts of their enterprises. Data will reside in many places in the years ahead—in on-premise legacy systems, in private clouds, and in public clouds. The experience in using this data needs to be the same, with the underlying platform virtually invisible to the end user. In addition, IT departments should be able to deliver services through an interface that is as easy to use as public a cloud service.

Today's users may be accessing data and applications from anywhere around the world.

REACH OUT TO THE EDGES.

A concept that is gaining traction in enterprises is "fog computing," in which access, storage, and processing can take place anywhere between a centralized server and edge device. This is critical, as much of the information that will drive businesses—operational issues, product usage, and customer trends—will come from devices and sensors situated at the edges of enterprises. Sensors in production systems may be streaming time series data on the health of the system, for example. A modern data architecture needs to support edge computing, as well as streaming data.

BE HIGHLY AVAILABLE.

A modern data architecture needs to ensure that data and applications are available to end users at all times, even if there are disruptions or outages to the system. This is accomplished through a well-defined resiliency strategy that, in the past, typically required a secondary data center, but today may involve cloud resources. Disaster recovery, failover, load balancing, backup, and data restoration processes need to be built into architectural planning. This requires greater planning than was necessary in the simpler days of data mirroring or replication between primary and hot standby systems. Today's data environments may be dependent on various cloud services, in conjunction with on-premise environments. While cloud providers are very adept at maintaining highly resilient systems, data architects need to look at the whole picture of how data is stored and used across their enterprises.

Fundamental to data architecture success is a data infrastructure that is agile and flexible.

BE SELF-DRIVING.

There are many database tasks and operations that take up inordinate amounts of staff time that could be applied to higher-level tasks, such as backups, patching, security, and anomaly detection. A modern data architecture incorporates automation of such tasks. AI and machine learning can play important roles in increasing the ability of databases to manage themselves.

PROMOTE COLLABORATION IN DESIGN AND IMPLEMENTATION.

Importantly, a modern data architecture is not solely the domain of the IT department. It is a set of services and capabilities that are built and used by just about every department across the organization. A modern data architecture needs to be closely tied to the needs of the business, and needs to have the flexibility to adapt to and meet business changes. While data managers and IT departments will continue to oversee infrastructural and security matters, their business partners in their organizations will help set the agenda for the types of analytics delivered, the speed of delivery, and the capabilities offered.

ENSURE A HIGHLY SECURE AND COMPLIANT INFRASTRUCTURE.

Cybersecurity needs to be baked into a modern data architecture right from the beginning. The goal of a modern data architecture should be to reduce the potential attack surface as much as possible. In addition, as data is subject to many regulations and mandates, the data architecture needs to build in checks and balances, as well as transparency to ensure that data is well-maintained and secured.

A modern data architecture is a living, breathing entity that grows and adapts as the enterprise itself evolves. Today's enterprises run on data, and their success depends on how well they leverage their data assets. ■

—Joe McKendrick

DATASTAX

Designing a Future-Proof Data Architecture

What does a data architecture that can withstand nearly anything thrown at it—both now and in the future—look like?

Without a doubt, that's a rather foggy crystal ball to gaze into, but there is a way to bring things more clearly into focus. By reviewing the most often cited modern application data needs and issues that typically cause IT architectures to buckle and collapse, you can design a data blueprint that can hold up even under the nastiest of weather.

Let's start with some of the most common requirements and then go into others that have more recently come about with the current evolution of database systems, digital applications, and their radically distributed deployments.

VOLUME AND TRAFFIC

More data + more users = database headaches.

That's a pretty well-understood and proven formula where databases are concerned. And today's modern applications, with their keep-all-data-online requirement and concurrent user traffic that can spike 1,000x in an hour, will easily make a wreck of even the best laid-out data architectures.

TIP: The keys to future-proofing a data design where these two issues are concerned are to (1) rather than legacy scale-up approaches, utilize a divide-and-conquer method that scales out the data layer, which means (2) utilizing a masterless database architecture that is able to remove both data volume and user-traffic bottlenecks through an elastic design. The architecture should work in a uniform way whether it's on-premises, in the cloud, and in hybrid deployments.

CONSTANT UPTIME

Trust us when I tell you that you have *way* more downtime risk than you think you do.

Studies done by the Uptime Institute humble even the cockiest IT professional who thinks their systems won't go down—especially those who trust the cloud to save them. Even with all the advances we have in technology, Uptime's studies show that ***outages are actually increasing*** and—take a deep breath—cloud providers are now the second most commonly cited reason for IT service failure (No. 1 is still on-premises data center issues).

TIP: If you've put a scale-out masterless data foundation in place, you need to make sure that it's one that has built-in redundancy in compute, storage, and data. If this three-legged data architecture stool is in place, then you have a good shot at *continuous* availability versus simply *high* availability.

LOCATION INDEPENDENCE

If you want a future-proof data architecture, it needs to be one that is location *independent* in nature versus location *dependent*.

Location independence means that your data can live anywhere and not only be read but written everywhere as well. There are three reasons you need location independence. First, as just described, with properly used data replication, it allows for constant uptime versus almost assured downtime at some point because you have multiple copies of your data in different locations.

Second, it provides for uniform customer response times. Being able to put your data where your customer is means they'll be able to access their data just as fast in any location, which is especially important since many application architectures are multi-home in nature.

Last, it protects against vendor lock-in and vendor location limitations. Having your data held hostage by a particular platform vendor who is limited in location support, with the only way out being a costly and heavy-lifting migration initiative, is not wise.

TIP: Be sure to pay attention to the fine print of your database vendor's architecture diagrams and descriptions.

Many can distribute data to multiple locations for read operations, but they can't do the same for write activity. You need both, plus a transparent and easy way to put your data where it's needed and sync it up with other copies of it around the globe.

TRANSACTIONAL CONTEXT

User and application database interactions have evolved far past the standard transactions offered in legacy relational engines.

For example, take your typical credit card transaction. The authorization process used by the credit card vendor contains many different contexts in order to avoid a fraudulent event. Not only does it contain the standard database transactional characteristics, but it also involves search operations that review historical purchase activity, followed by analytics that are run on that data, which ensures it's in line with the current purchase, and then it goes through the approval process and returns its response back to the application and user. One transaction, done in split-second fashion, so the user doesn't grow impatient and move on to something else.

A multi-model database approach represents the next phase of maturity for our database industry and addresses these hurdles, especially as it relates to mainstream developers and administrators within the enterprise.

There are various industry terms used for contextual transactions—hybrid transactional analytical processing (HTAP), translytical processing, and so on. Whichever name you use, the idea is that your typical ACID transactions of legacy database yore are long gone

Instead, a data architecture that forgoes the typical separation of database workloads is required today. This is why Gartner, in their "There is Only One DBMS Market" report, says: "The separation of the DBMS market into operational and analytic tasks has outlived its usefulness. Now each type of use case can be addressed by a single vendor or product. Data and analytics leaders responsible for data management strategies must view DBMS as one market to drive new capabilities."

TIP: Your future-proof data architecture needs DBMS systems that handle contextual transaction management, with full support for transactional, analytical, and search workloads.

MULTI-MODEL

Microservices architectures are the *soup du jour* today where developing modern applications are concerned.

The idea is to stitch together components that roll up to the big picture application in a way that allows development teams to more easily work in parallel and thus get things to market more quickly. Key to the success of microservices is strong flexibility at the data layer, which accommodates the concept of polyglot persistence in a more modern way than previous implementations.

In the past, polyglot persistence implied that customers would use either a limited set of data models (key-value, tabular, JSON, relational, graph) or perform extract-transform-load (ETL) operations across data stores. Use cases such as master data management, customer-360-view, and others, mandated the latter approach, which increased complexity and total cost of ownership (TCO) because multiple vendors and cost models were involved.

Development was difficult because each vendor's mechanism for interacting with the data store was different, both with respect to its dialect and where they lay on the logical / physical design divide. This forced application developers to write abstraction layers if they needed more than a single model in their application. Further, these abstraction layers had to work at very different levels across the physical / logical spectrum to keep application development aligned.

A multi-model database approach represents the next phase of maturity for our database industry and addresses these hurdles, especially as it relates to mainstream developers and administrators within the enterprise. This is accomplished with a single, integrated backend that:

- Supports multiple data models (e.g., tabular, JSON, graph) at a logical layer for ease of development for application developers

DATASTAX

- Ensures all models are exposed via cohesive mechanisms, thereby avoiding cognitive context switching for developers
- Provides a unified persistence layer that delivers geo-redundant, always-on characteristics and a common framework for operational aspects such as security, provisioning, disaster recovery, etc.
- Empowers a variety of use cases across OLTP and OLAP workloads for lines of businesses within an enterprise to innovate with agility

All relational database management system (RDBMS) vendors offer a version of their database in the cloud, however, that doesn't make them cloud databases. The saying, "There is no cloud; it's just someone else's computer," applies to databases as well.

- Delivers best-in-class TCO efficiency for the long haul to enable wider adoption within centralized IT teams of an organization

TIP: Ensure your data platform supports true multi-model capabilities versus a surface-level implementation of data store support. As an example, a number of relational and NoSQL vendors offer graph functions in their core model, which fall light-years short of a true graph database.

CLOUD NATIVE

You can park a car in a lot full of boats, but doing so doesn't make the car a boat.

All relational database management system (RDBMS) vendors offer a version of their database in the cloud, however, that doesn't make them cloud databases. The saying, "There is no cloud; it's just someone else's computer," applies to databases as well. There is no cloud pixie dust that auto-magically takes legacy relational databases or certain NoSQL databases and transforms them into a native cloud database that exploits all the benefits that cloud offers.

Why care about this? Because your future-proof data architecture will almost certainly use cloud in some way. In their "The Future of the DBMS Market Is Cloud" report, Gartner states, "Database management system deployments and innovations are increasingly cloud-first or cloud-only. Data and analytics leaders selecting DBMS solutions must accept that cloud DBMS is the future and must plan for shifting spending, staffing and development accordingly."

So how does a native cloud database behave versus the pretenders? A short list of the characteristics include:

- Transparent elasticity—being able to easily expand and contract resources given usage and resource demands.
- Unbounded scalability—one TB or PB, one thousand users or ten million, the cloud database elasticity should include seamless scalability (and make no mistake, the two are not synonymous).
- Built-in redundancy—the database should be able to fully exploit different regions, availability zones, and (yes!) different clouds to guarantee zero downtime and no loss of data access.
- Simplified data distribution—same as location independence described above.
- Autonomous manageability—the typical administrator tasks of backup, provisioning, tuning, etc., should be handled in a hands-free way.
- Uniform security—data protection should be applied in a consistent manner across workloads, data models, other clouds, and on-premises participants in the deployments.

TIP: The underlying data architecture's foundation is absolutely key here—a masterless, shared-nothing design will allow for all of the above requirements to be fulfilled either within the database itself or via simple integration with the cloud platform vendors.

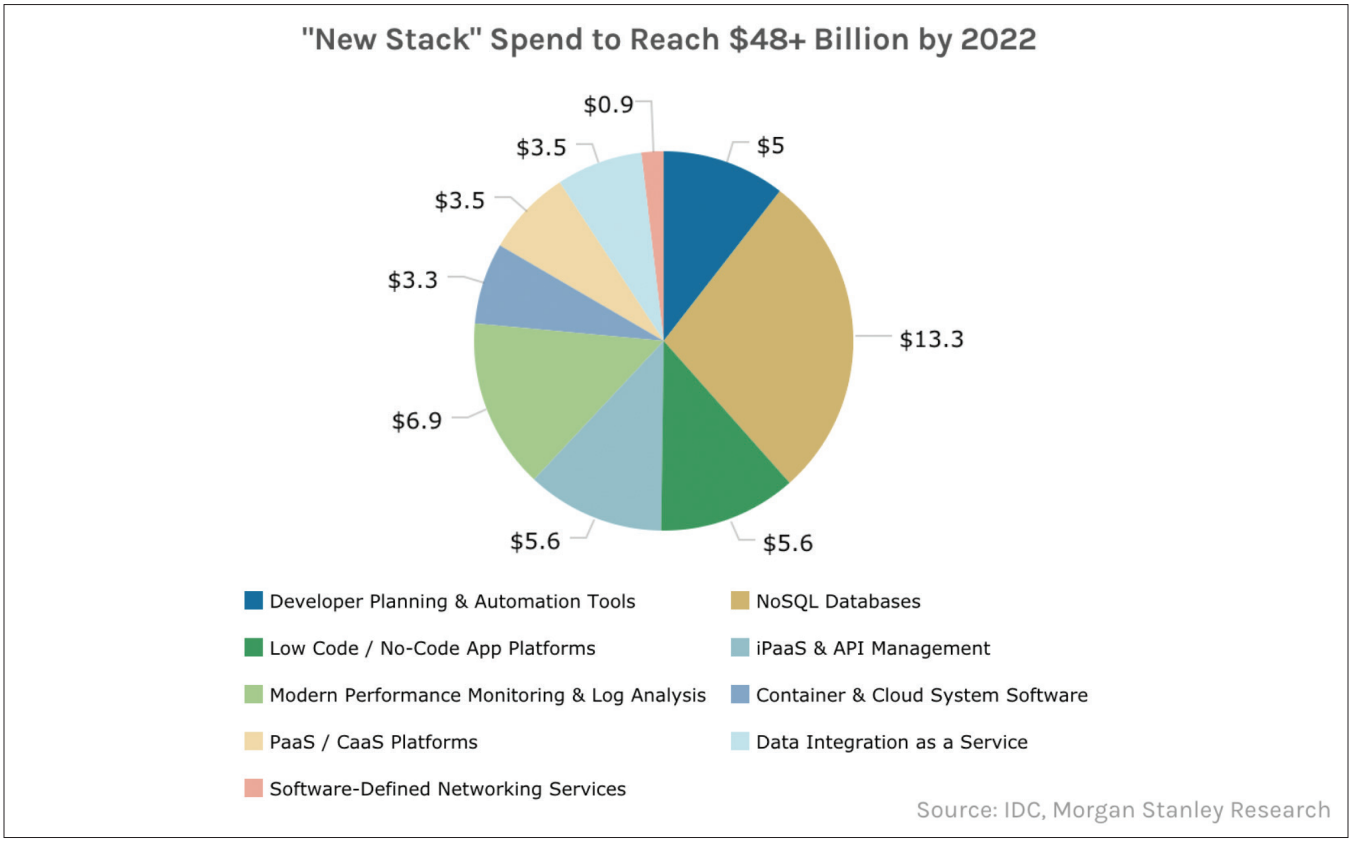
WINNOWING THE FIELD

There's no doubt that the requirements we've covered for a future-proof data architecture are tough, but c'mon, you didn't think this would be easy did you?

So what will step up to the challenge? Traditional relational databases are good for centralized and legacy applications (e.g., ERP) but aren't really a fit for widely distributed digital applications. Instead, you need something that's a little different than your grandfather's database.

In a recent report, analysts at Morgan Stanley assert that the "new stack" needed for modern applications consists of:

- Developer planning and automation tools
- NoSQL databases



- Low code / no-code app platforms
- iPaaS and API management
- Modern performance monitoring and log analysis
- Container and cloud system software
- PaaS / CaaS platforms
- Data integration as a service
- Software-defined networking services

Their prediction for the modern data architecture layer is NoSQL, with their forecast being that it will attract the most new dollars spent:

However, just because a database platform is non-relational in nature, doesn't mean that it will be able to support the requirements needed for a true future-proof data architecture. As an example, most NoSQL databases mimic the master-slave design of relational databases.

SUMMING THINGS UP

So what does a data architecture that can withstand nearly anything thrown at it, both now and in the future, look like? It's one that:

- Doesn't hit the wall when the **data volumes and user traffic** swell to extremely high levels.
- **Stays online** come hell or high water.
- Offers **location independence**, allowing you to read and write data anywhere in the world.
- Supports **contextual transaction processing** so that various workloads—standard transactions, analytics, search—can all be supported within the context of a single interaction.
- Has not only **multi-workload functionality**, but **multi-model support** as well, which helps streamline microservices development.
- Is **cloud native** in the sense that it seamlessly takes advantage of all the cloud's benefits, while sporting security uniformity across the platform and allowing for the ultimate in deployment flexibility (on-premises, single cloud, multi-cloud, hybrid cloud).

If you're looking for more details on how to actually implement a future-proof data architecture, please give our experts at DataStax a shout (datastax.com/contactus) and they'll be happy to show you how it's done. ■