ORACLE

Oracle's Converged Database: How to Make Developers And Data More Productive

As enterprises digitize more business processes and decision points, they face a seemingly impossible choice—improve developer productivity now or data productivity later. But a radically new approach, Oracle's converged database, breaks this impasse.

December 2020 | Version 1.2

Purpose Statement

This document is intended to help CTOs, enterprise architects, and development managers understand the benefits of converged databases compared to single-purpose databases.

Intended Audience

The intended audience of this paper is I.T. leaders making decisions about the future of enterprise computing architecture, including CTOs, enterprise architects, and development managers.

Disclaimer

TABLE OF CONTENTS

THE DILEMMA: DEVELOPER PRODUCTIVITY OR DATA PRODUCTIVITY

Enterprises have to create unique data assets and make the most of them to remain competitive. But as companies create more applications, analytics, and AI to digitize more processes and decision points, they're faced with a difficult choice: to optimize for either fast application development now or easier value creation from data later. In other words, developer productivity or data productivity.

To optimize for developer productivity, teams spin up single-purpose databases for specific projects. Each database offers a convenient data model for that purpose and a simple set of APIs, making it easier to start developing against them. However, as a project grows and additional single-purpose databases or cloud services are required, data fragments across these services. Each has its own tooling, security methods, and operational characteristics, risking inconsistent data, security gaps, and increased difficulty in using that data in critical reporting and analytical work.

To optimize for data productivity, teams build on instances of a corporate standard database, usually a relational database or a relational-based multi-model database. The corporate standard database enforces official policies and simplifies anticipated data reuse, like reporting, but limited functionality may slow or even prevent innovation. This risks putting the entire business at a disadvantage to the competition.

Neither one of these choices is acceptable. How did businesses get stuck with this dilemma? More importantly, how do they escape it?

INTERNET CONSUMER SERVICES: WHERE DEVELOPERS REIGN

When internet consumer services like search, ecommerce, and social media took off, they faced a new set of requirements from enterprise applications. They typically used relatively simple application logic because they supported piecemeal customer interactions rather than detailed business processes. They didn't connect into existing enterprise systems, and they preferred lightweight data models like documents, objects, and graphs that more closely fit their web-based app development methods. These services faced unprecedented volatility with peaks of tens to hundreds of millions of users. And those users cared far more about the overall experience of the service than the integrity of the data it might capture.

To deal with these new requirements, internet pioneers invented their own data tier from individual services, each supporting a specific data model, access method and workload needs. These data tier cloud services embodied features that made sense for internet consumer services and embraced a specific set of trade-offs:

1. *Availability over consistency at extreme scale*. A down service is a useless service. Making sure the service remained available, even at the expense of applications having to manage inconsistent data, is paramount. Therefore, transaction support and data consistency were relaxed to improve performance at extreme scale.

2. *Bitbuckets over database management systems (DBMSs).* Bare-bones, single-purpose APIs that offered basic requests consistent with internet protocols, supporting a specific data model or workload such as JSON, graph, or analytics, were preferred. Although these minimal databases were too primitive for complex enterprise processes, they could still support the development of individual features of consumer internet apps.

3. *Microservices over monoliths.* If the application required more features and complexity, then multiple "bare-bones", single-purpose services could be leveraged together to build richer applications while maintaining tolerance to individual service failures. This approach, based on microservices, allowed development teams to operate more independently of each other, potentially increasing agility and innovation.

GREATER FLEXIBILITY, BUT INCREASED OPERATIONAL COMPLEXITY AND RISK

By building on a new cloud data tier consisting of single-purpose databases intended for specific data types and workloads, early adopters gained faster development time for simple cloud consumer applications.

But integrating multiple, single-purpose databases to create a complete, highly available, and secure enterprise solution quickly becomes complicated. It may require a lot of custom code and hard trade-offs to manage data fragmented across multiple services. The enterprise itself bears the burden of the continuous integration work required to make apps built on an array of single-purpose databases feasible on a large scale, leading to endless re-integration, re-testing, re-tuning, and troubleshooting.

Personnel need to be knowledgeable about the operational aspects of each single-purpose database. Security policies need to be re-implemented in every database, and apps become more complex as they propagate data from one database to another. Ironically, combining "best-of-breed" single-purpose databases often results in the "worst-of weaknesses" for the shared capabilities necessary to run an enterprise application such as security, scalability and replication. The weakest link will, unfortunately, define the strength for all.

ENTERPRISE SAAS: WHERE DATA IS KING

On a parallel track from consumer internet apps but at roughly the same time, enterprise cloud applications, more commonly known as SaaS (Software-as-a-Service), reimagined enterprise apps for delivery over the internet. Unlike consumer internet services, enterprise SaaS apps used relatively complex application logic to support critical business processes. They often had to connect into existing enterprise systems. SaaS apps required extreme scale. And the business using them cared just as much about the data created through these apps as the apps themselves. Strict transactional consistency, as well as data integrity constraints, were non-negotiable.

As a result, enterprise SaaS vendors stuck with relational databases but pushed them, and the vendors like Oracle who built them, to new levels, demanding:

1. *High availability and consistency at very large scale.* ACID transactions were required for applications, data had to be correct, and systems available.
2. *Database management systems (DBMSs) with bitbucket behaviors.* Relational databases supported complex queries and application logic plus reporting. It was essential for enterprise data to be available for analytics and reporting. But enterprise SaaS vendors needed support for new data models and workloads too.
3. *Monoliths and microservices as needed*. Instead of using a purely microservices-based approach, enterprise SaaS vendors used designs where most of their application code remained in a monolith, but microservices might be utilized to add functionality, such as mobile interfaces, or to improve performance in an isolated but critical section of code.

THE KEY TO HAVING BOTH: ORACLE'S CONVERGED DATABASE

Enterprises pursuing digital transformation face the same challenges as internet consumer services and enterprise cloud apps, but with the additional burden of making sure these new systems interact with existing ones.

Companies need to create consumer-facing mobile and web apps with the same rapid iteration and flexibility as the internet giants. They also have to provide IT services to multiple departments with the same agility as commercial SaaS providers. Plus, they have to make their existing enterprise systems use data from those new environments and contribute data back to them.

However, this is extremely difficult when each environment is built on different single-purpose databases with different operational, security, and performance profiles. Instead, firms need a unified data tier supporting all of these apps, analytics, and AI algorithms. This requires an innovation in data management—a converged database.

A converged database is a *multi-model, multitenant, multi-workload* database (Figure 1). It supports the data model and access method each development team wants, without unneeded functionality getting in the way. It provides both the consolidation and isolation these different teams want but don't want to think about. And it excels in all the workloads (like OLTP, analytics, and IoT) these teams require. Oracle Database 19c is the world's first converged database.

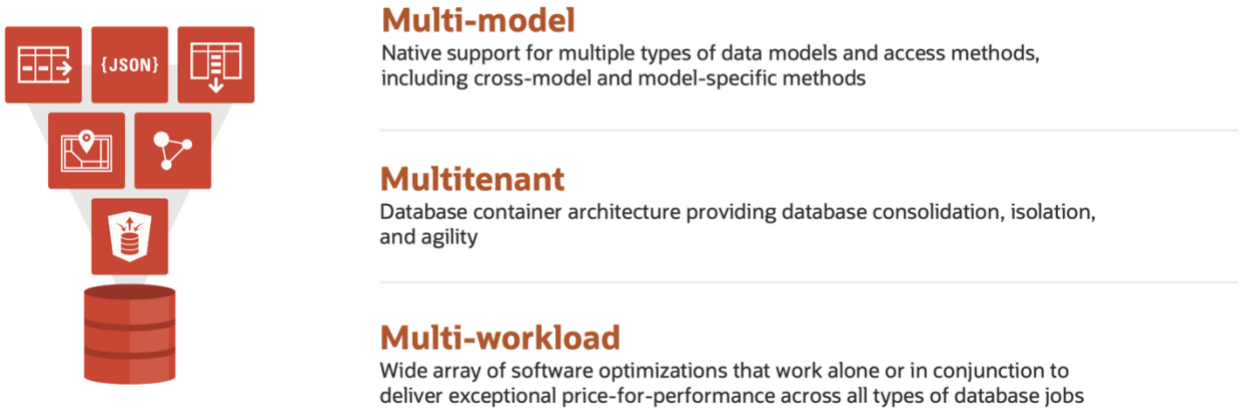

Figure 1. Oracle Database 19c is the world's first converged database

POWERFUL SYNERGIES

A good analogy for a converged database is a smartphone. Consider how smartphones integrated phone calls, messaging, a camera, calendar, music and other features into a single product when each initially required separate products. Now, these point products are mere features of smartphones. Smartphones rely on constantly improving custom integrated circuits produced at very high volume to

achieve their small size, low power consumption, and unique features such as augmented reality, virtual reality, and 3D. With smartphones, synergy across features makes the whole better than the sum of parts, because of the tight integration and the new workflows this integration makes possible.

For example, the camera in your smartphone integrates with a lot of other applications. It automates the photo storage and backup process, allows pictures to be sent in emails and texts and easily posted on popular social media sites, and can even provide automated editing and color correction in real-time. The calendar is continuously updated since it uses the phone's internet connectivity to sync with the cloud. The music app can stream music continuously from an extensive music library in the cloud. Each of these separate features is more capable and powerful in some ways compared to its standalone, single-purpose counterparts. Expensive, single-purpose, high-end cameras and stereo systems can still deliver higher quality pictures and sound than your smartphone camera and music app, but the economies of scale smartphones achieve versus these high end systems (with volumes in the millions versus only thousands for high end cameras) helps them narrow the gap over time.

The same ease of use, convenience and synergy you get from a smartphone also holds for a converged database. A converged database makes it much simpler to develop applications because standard SQL can be used to run very sophisticated machine learning, spatial, and graph algorithms instead of implementing these in separate databases and APIs. Instead of writing complex messaging and event code to weave data together, you can use standard SQL functionality like JOINs. If a developer prefers to program directly to the native data type without using SQL, Oracle provides rich APIs for the most popular data models, including JSON and graph, that support this.

Beyond the synergies and dynamic new workflows, built-in capabilities of Oracle's converged database often surpass those of single-purpose databases. Oracle's converged database delivers the most complete and highest quality feature sets for spatial, graph, JSON, machine learning capabilities, and more. Industry analysts have noticed this as well; ranking Oracle's converged databases number one in all areas that are purportedly the focus of some niche, single-purpose database.[1] Choosing to use Oracle's converged database does not mandate acceptance of lesser capabilities for the specific requirements of each data model, workload, or development paradigm.

MULTI-MODEL: ONE ENGINE, MANY PERSONALITIES

Let's consider multi-model first: how does supporting different data models and their access methods in a single engine improve both developer productivity and data productivity?

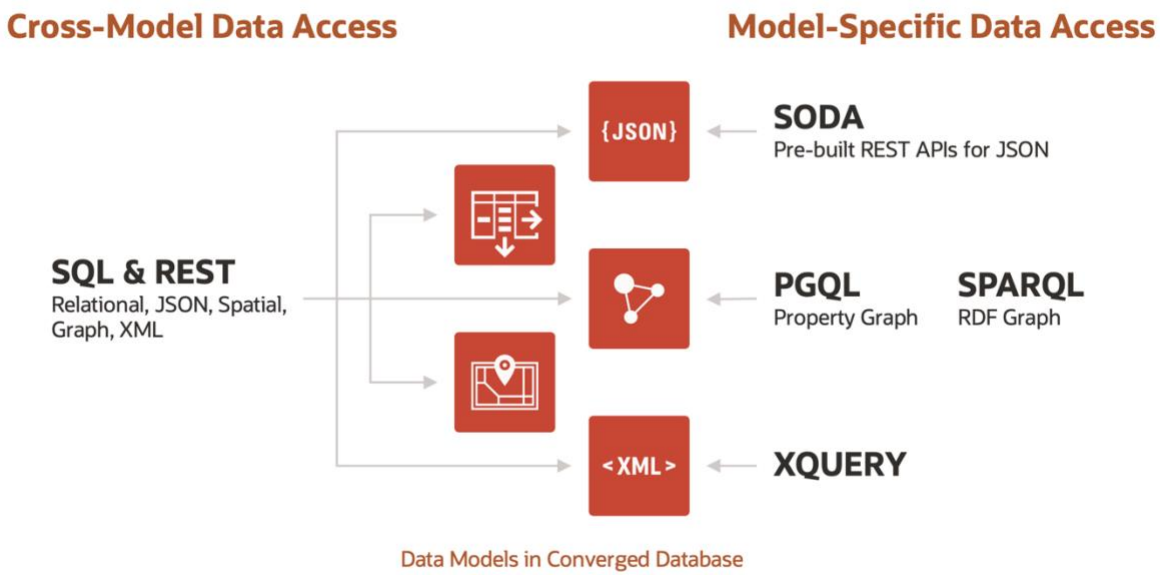

Figure 2. Oracle's converged database supports multiple data models and access methods

[1] Gartner, Critical Capabilities for Operational Database Management Systems https://www.gartner.com/document/3975496

Unlike single-purpose databases, Oracle's converged database supports JSON, XML, relational, spatial, graph, IoT, text and blockchain data with full joins, transactions, and other critical SQL features enterprises rely on (Figure 2). In addition, Oracle's converged database also supports model-specific access methods for graph and spatial queries, as well as hundreds of common machine-learning algorithms. These abilities are accessible through RESTful APIs as well as stateful connections, leaving the choice in developers' hands.

Oracle's converged database goes even further in its support of JSON data commonly used in web and mobile applications. Simple Oracle Document Access (SODA) is a set of pre-built RESTful APIs that allow developers to create and query JSON collections without having to use SQL. However, unlike single-purpose document databases, Oracle Databases can generate a schema and indices from JSON objects to enable parallel SQL analytics, transactions, and joins of JSON data with spatial, graph, and relational data. In fact, one of the largest consumer electronics companies in the world uses the Oracle Database JSON features to support world-wide point-of-sale transactions.

These multi-model capabilities allow enterprises to have both developer productivity now and data productivity later. They give developers simple API-driven access and model-specific languages, while still having recourse to powerful SQL capabilities whenever they want. Meanwhile, IT enjoys a common approach to security, upgrades, patching, and maintenance across all deployments of Oracle's converged database.

MULTITENANT: CONSOLIDATION, ISOLATION, AND AGILITY

Containers, one of the most powerful ideas in apps today, got its start in 1979 as a way to isolate individual processes on a shared Unix machine. Containerization not only provides better isolation, but when coupled with orchestration of the containers, virtualizes the underlying resources, leading to more efficiency, agility, and portability.

But just as OS containerization means something different from app containerization, so too does containerization of the database.

Unlike containerized apps which can disappear when no longer needed, databases need to persist data. Storing data durably and making it available for later usage is a core part of a database's job. So how do you get container capabilities in the data tier? The Oracle converged database, with its unique multitenant features, incorporates containerization and orchestration within the data tier itself.

The multitenant architecture of Oracle's converged database allows a single container database to support multiple pluggable databases (Figure 3). The Oracle container database is analogous to an app container engine in that it supports multiple pluggable databases (or, data containers), just as the app container engine supports multiple application containers.

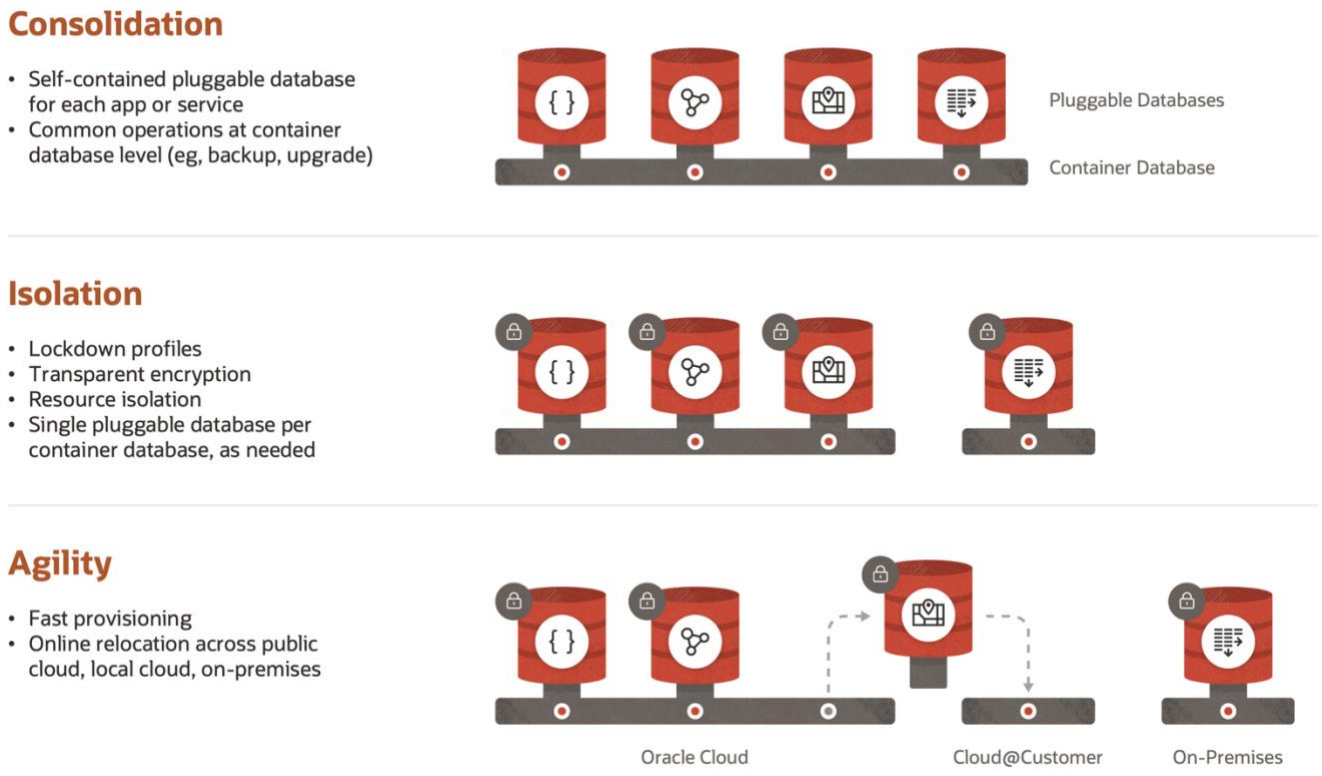

Figure 3. Oracle's converged database provides containerization in the data tier

Like application tier container orchestration frameworks, such as Kubernetes, Oracle's multitenant cloud architecture leverages container databases to orchestrate pluggable databases in a unified data tier. With this multitenant architecture, Oracle's converged database delivers:

- *Technical efficiency*: Supports multiple databases in isolated workspaces while sharing common infrastructure. This eliminates redundant replication of overheads, enabling more databases per server.
- *Operating efficiency*: Administrative costs for the common environment are divided between multiple databases, effectively allowing you to manage many as one.
- *Agility*: It's simple to provision new databases, clone existing ones, or redeploy them on other platforms as needed for consolidation, isolation, or performance.
- *Ease-of-use*: Databases (and their schemas) run unchanged. Scalability is transparent. Centralized management reduces complexity. Simple SQL extensions control new multitenant capabilities, like rapid provisioning and cloning.

The multitenant architecture of Oracle's converged database also makes it easier to develop and deploy microservices by providing a pluggable database for each microservice. Each pluggable database is isolated, secured, and uses the exact data type or workload a particular microservice requires—even as the Ops team manages many databases as one through the container database.

The increased efficiency and simplified management of a unified data tier leveraging multitenant architecture increases developer productivity. Developers can focus on application development rather than data management tasks. In addition, the pluggable databases in the unified tier are easily accessed by data scientists and business analysts using their preferred tools and access methods to generate reports, perform analytics and build AI models, increasing data productivity.

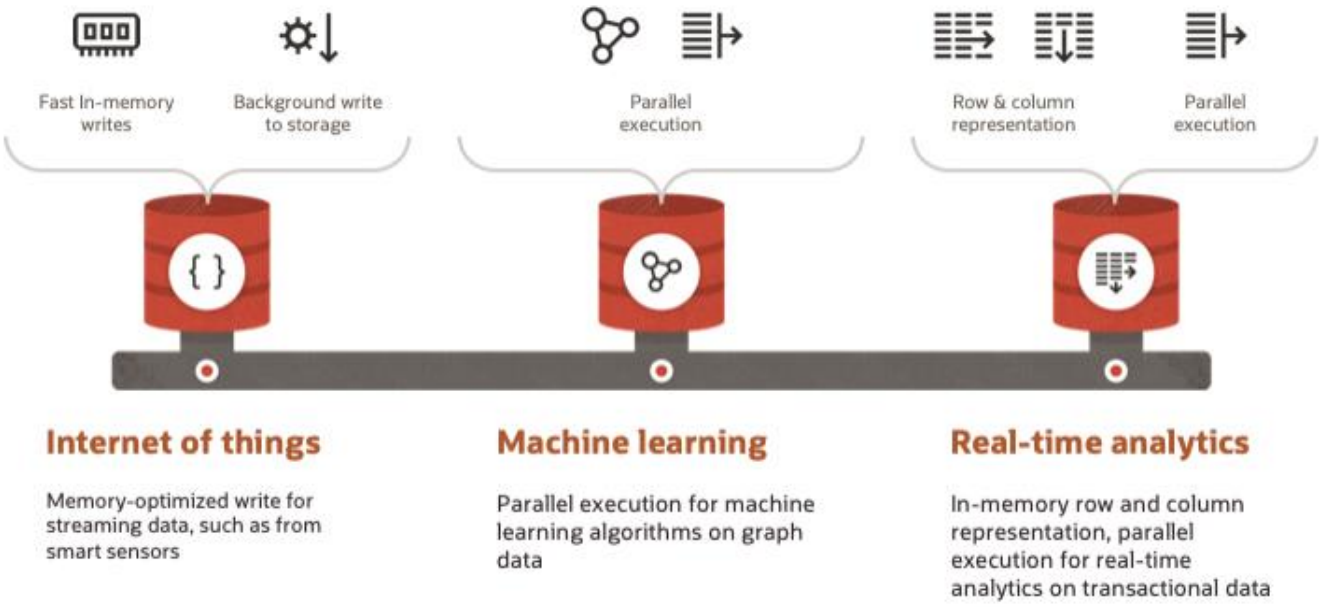

Figure 4. Oracle's converged database provides a wide array of optimizations for different data workloads

MULTI-WORKLOAD: DOING MANY JOBS AT ONCE

Different kinds of database workloads require different kinds of software optimizations to remain performant at scale (Figure 4). For example, smart sensors taking frequent measurements need a database that can ingest a large number of new records extremely quickly. Oracle's converged database supports this by letting an IoT app write its data directly into a memory buffer while, in a separate background process, the database commits those records to disk. This ability enables Oracle's converged database to handle 25 million inserts per second on a two-socket server by eliminating the wait for data to be persisted to disk, which is a slower process.

But training machine-learning models is a very different kind of job. This involves extremely large numbers of relatively simple calculations which means a lot of CPU cycles. Oracle's converged database includes hundreds of common machine-learning algorithms which have been modified to allow parallelization across many CPU cores, dramatically speeding up these kinds of calculations.

In some cases, a development team needs a single converged database to support multiple diverse workloads simultaneously. For example, most modern transactional applications need to run at least some analytics on operational data in real-time. Oracle's converged database optimizes both of these workloads behind the scenes by representing the same data in both a row format (for fast transaction processing) and columnar format (for fast analytics) and keeping the two perfectly in sync as the data changes.

Developer productivity improves when a core database engine fully supports different kinds of application workloads without requiring excessive customization or app tuning. Likewise, data productivity improves when that same core engine also fully supports different analytical workloads critical to data scientists and business analysts.

Oracle's converged database lets DevOps teams turn these different optimizations on and off without changes to the application. This enables performance tuning as demand on the database grows and even as the kind of demand changes.

All of these software optimizations have implications for the physical computing resources required, including processors, memory, disk, and input/output between these different components. Many enterprises have built sophisticated combinations of blade servers and network attached storage to provide Oracle's converged database with an operational environment to support its extraordinary abilities. But there's a better way—Oracle Exadata.

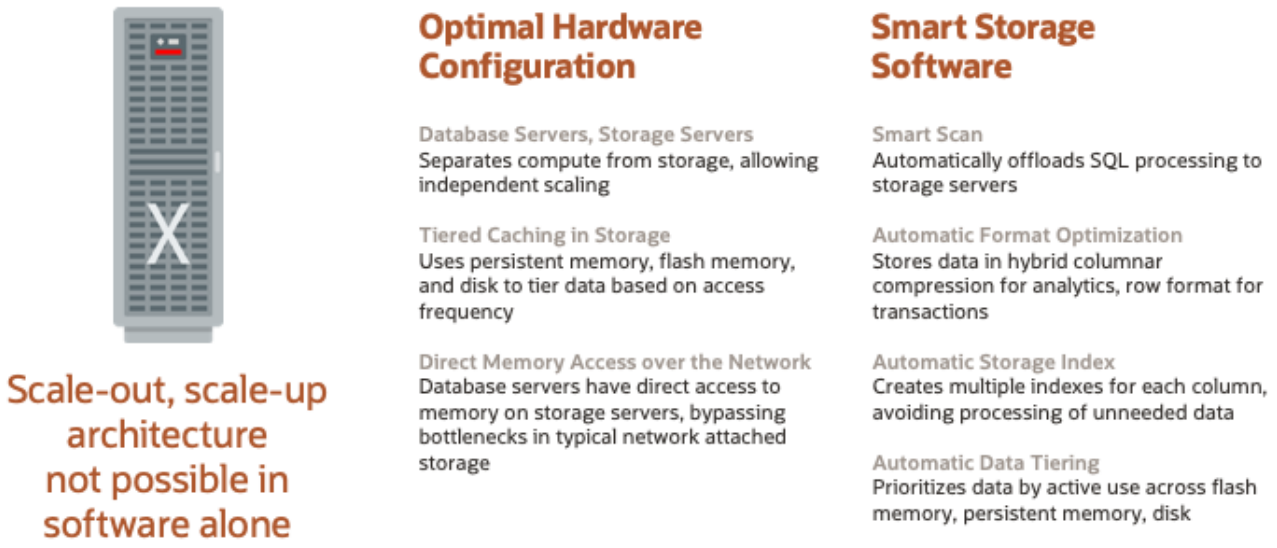

Figure 5. Oracle Exadata combines hardware and software innovations to provide unbeatable price-for-performance for all data workloads

ORACLE EXADATA

Oracle Exadata is a combination of hardware and software specifically designed for database workloads (Figure 5). Using an architecture that separates compute from storage, Oracle Exadata provides a modular scale-out/scale-up solution with the best price-for-performance in the industry for database workloads. The reason Oracle Exadata outperforms all other database architectures is Oracle's co-engineering process which optimizes database software to squeeze every drop of performance from the latest hardware innovations incorporated into a known architecture.

Oracle Exadata X8M, which uses Intel® Optane™ Persistent Memory in its storage servers is a prime example of this process. Persistent memory is extremely fast, like normal dynamic memory, but persists data in the event of power loss, like disk. This creates the opportunity for radical increases in performance if only the database software can be modified to take advantage of it.

The average network latency of round trips between a database server and storage would negate the speed persistent memory can offer. Instead, Oracle Exadata X8M uses Random Direct Memory Access over Converged Ethernet (RoCE) to take advantage of data cached in memory on the storage servers. This is just one of thousands of hardware-software collaborations that make Oracle Exadata the most cost-effective platform for all database workloads.

Oracle's converged database, running on Oracle Exadata infrastructure is available as a public cloud service in Oracle cloud, as Oracle Exadata Cloud Service, via Oracle Exadata Cloud@Customer and Oracle Dedicated Region, and on-premises as Oracle Exadata.

ORACLE AUTONOMOUS DATABASE: CONVERGED DATABASE AS A SERVICE

Oracle's converged database achieves its ultimate expression in the Oracle Autonomous Database, which consists of Oracle's converged database running on Oracle Exadata infrastructure, delivered as a fully managed cloud service. There's no administration required because the Autonomous Database is:

- **Self-driving.** You tell the Autonomous Database the service level to achieve, and it handles the rest. The Autonomous Database automates the provisioning, securing, monitoring, backup, recover, troubleshooting, and tuning of databases. This dramatically reduces mundane database maintenance tasks, reducing costs and freeing scarce administrator resources to work on higher value tasks.

- **Self-securing.** The Autonomous Database is more secure than a traditional database deployment because it protects itself. This applies to defenses against both external and internal attacks.

- **Self-repairing.** The Autonomous Database is more reliable than a traditional database deployment. At startup, it automatically establishes a triple-mirrored scale-out configuration in one regional cloud datacenter, with an optional full

standby copy in another region. The Autonomous Database automatically recovers from any physical failures, whether at the server or datacenter level. It has the ability to rewind data to a point in time in the past to back out user errors. By applying software updates in a rolling fashion across nodes of the cluster, it keeps the application online during updates of the database, clusterware, OS, VM, hypervisor, or firmware.

The Oracle Autonomous Database is available in several popular personalities to save time for developers, data scientists, and analysts who want to get to work on the applications, AI, and analytics they really care about. The family of Autonomous Database services includes:

- **Autonomous Transaction Processing (ATP)** simplifies database operations for OLTP and mixed workloads requiring real-time or batch analytics. ATP reduces runtime costs by up to 90% and provides unparalleled scale, performance, and security with embedded machine learning-based automation.

- **Autonomous Data Warehousing (ADW)** is optimized for analytical processing. It automatically scales compute and storage, delivers fast query performance, and allows querying data beyond the data warehouse in other cloud services like object storage and Kafka streams.

- **Autonomous JSON Database (AJD)** is tailored to support JSON-centric app development. It provides native JSON management with advanced indexing and transparent scale-out, as well as full ACID transactions over JSON documents. All these capabilities are available through REST APIs.

ORACLE'S CONVERGED DATABASE DELIVERS THE UNIFIED DATA TIER

Oracle's converged database is available in a wide array of deployment scenarios across public cloud, local cloud, and on-premises. By relying on the same core engine delivered through different mechanisms in different operating environments, companies maintain their power of choice while creating a unified data tier (Figure 6). With this unified data tier, enterprises can:

- **Build both microservices and monolithic apps.** Oracle's converged database is a new way to support microservices applications. Each service has its own database with the model and access method it requires. Each of these databases can scale up or out as needed, using sophisticated methods like Real Application Clusters (RAC) or relatively simple tactics like sharding. And yet, because these databases are tenants of a supervising container database, they have one security model, patching, and upgrade path. Meanwhile, other instances of this same engine can do what they've always done—support the building and running of traditional enterprise applications.

- **Run workloads in the public cloud, local cloud, or on-premises.** Oracle's converged database, running on Oracle Exadata infrastructure is the only way to have the exact same data tier architecture in all three computing environments. In the Oracle Cloud, this architecture is available as the Oracle Autonomous Database and Oracle Exadata Cloud Service. In local cloud, it's available as Oracle Exadata Cloud@Customer and part of the new Dedicated Cloud Region offering. And on-premises, it's available as Oracle Exadata.

- **Do analytics and data science on operational data as well as data outside the database.** The multi-workload capabilities of Oracle's converged database make it possible to run analytical queries on data in operational databases supporting production applications. This provides real-time analytics on production data. In addition, instances of Autonomous Data Warehouse let analysts ask questions of data residing in object storage on the Oracle Cloud or other clouds.

- **Manage many databases as one—or not at all.** When every database instance your ops team has to manage is a different configuration of the same core engine, fleet management becomes much simpler. The entire fleet follows the same security protocols, patching methods, and upgrade path. With the Oracle Autonomous Database, not only does the entire fleet follow the same management methods, it does so on its own.

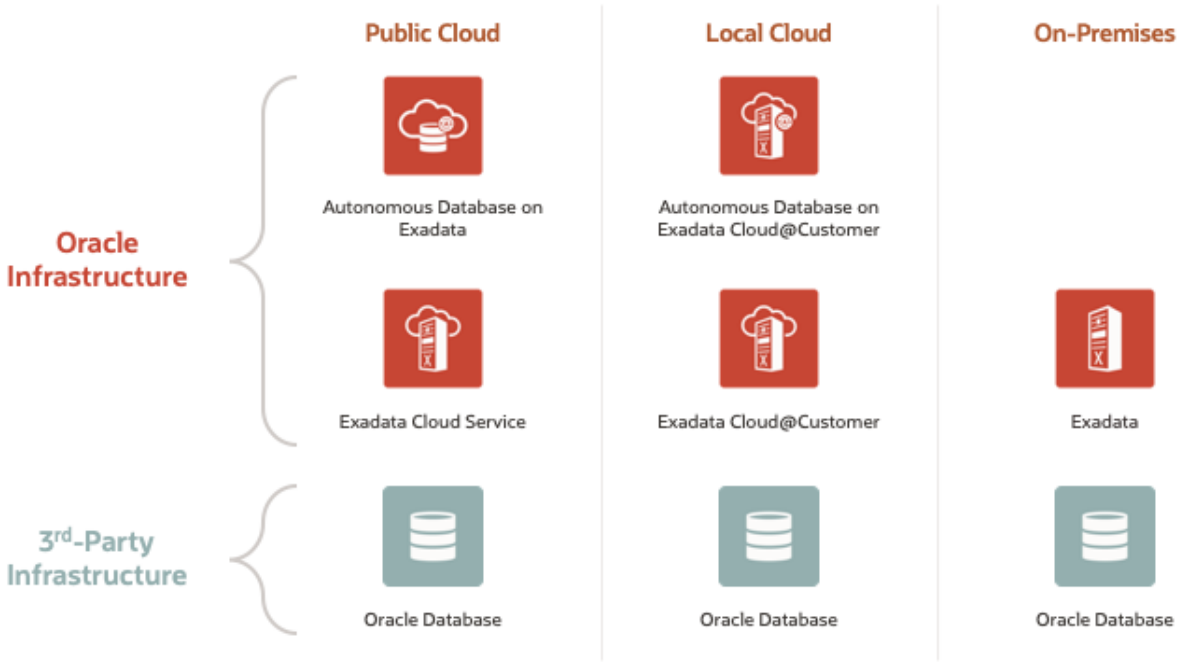

Figure 6. Companies can use Oracle's converged database in a variety of scenarios, creating a unified data tier across deployment environments

CONCLUSION

Oracle's converged database is a radical re-thinking of the Oracle Database. This multi-model, multitenant, multi-workload database allows enterprises to satisfy two seemingly opposite goals: give developers the right tools for each job to make them more productive, and IT a unified data tier that makes reuse of data and overall management simpler and easier. What this means for enterprises is a simplified foundation for digital transformation. With Oracle's converged database, enterprises can both digitize more business processes more quickly and inject more data into more decision points more confidently.

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com.
Outside North America, find your local office at oracle.com/contact.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Oracle's Converged Database: How To Make Developers And Data More Productive

By Matthew O'Keefe, Maria Colgan, Paul Sonderegger

December, 2020