db insight

Oracle’s Autonomous Database can now be globally distributed

Meeting the new reality for data sovereignty

Published March 2024

Executive Summary

Trigger

The distributed, scale-out topology of the cloud has enabled databases to literally expand their footprints across multiple physical locations. Oracle has become the latest provider to take advantage of the ability to distribute workloads with the release of Oracle Globally Distributed Autonomous Database. No stranger to distributing its database through use of sharding, how does the new globally distributed edition of Autonomous Database depart from Oracle's traditional Real Application Clusters (RAC) architecture, not to mention other distributed cloud database options from the hyperscalers and specialized providers such as Yugabyte and CockroachDB?

Our Take

Global, distributed databases comprise a large umbrella; there is no single recipe for distributing data that will satisfy every use case. So, while adoption of cloud computing has both enabled and driven the need for global distributed databases, they are not all alike. There is no single recipe that will satisfy requirements that may vary from the need for quick reads to quick reads *and* writes, and depending on the application, different degrees of ACID transaction consistency.

Oracle was one of the pioneers of distributed databases when it introduced Oracle Real Application Clusters (RAC) over 20 years ago. RAC was conceived before the rise of the cloud where the need was for greater scale, performance, and availability within the data center. Since then, the cloud has made distributed databases more commonplace. But significantly, they came later to relational databases because RDBMSs were associated with systems of record that required strong ACID transaction consistency. In the cloud, Google Cloud Spanner pioneered the fully distributed, global multimaster relational database. Since then, others have hopped on the bandwagon, but with a few exceptions, most have focused primarily on improving read response with read-only local replicas.

Sharding, which horizontally partitions the data, emerged to accelerate local reads *and* writes, but with most implementations, it required customers to build the routing logic into the application tier. When Oracle introduced sharding with database 12i in 2017, it was the first to incorporate sharding into the database tier, which eliminates the need for customers to rewrite their applications. It supports multiple independent sharded database instances that present as a single logical database to the application with features such as automated policy-driven data placement, automated application routing, elasticity, and manageability. Oracle has now taken sharding to the logical next step by adding a globally distributed edition to the Autonomous Database that enables customers who want to take advantage of OCI's growing global footprint. The key significance is that, by taking advantages of OCI's global presence, they can implement an autonomous database that satisfies the emerging breed of data sovereignty laws, which are increasingly becoming facts of life.

The synergy: the cloud and distributed databases

The ability to scale both compute and storage in the cloud tore the limits off databases and their use cases. Emergence of distributed databases is a classic chicken-and-egg question. Did they emerge because of the extreme scaling demands of Internet data applications or vice versa? Isolated sectors, such as the financial services industry, had such requirements before the advent of the Internet or the cloud. In those cases, much of the logic for routing and managing transactions was handled in the application rather than the database tier, which made routine maintenance and upgrade processes especially fragile. And the complexity placed the technology off limits to mainstream enterprises.

With the rise of the cloud, distributed databases first emerged on the NoSQL side, with Amazon DynamoDB and Google Bigtable prime early examples. For use cases such as counters, leaderboards, shopping carts, global scale, performance, and availability were more important requirements than strict transaction consistency. Distributed NoSQL databases with *eventual consistency* delivered the high performance and availability by enabling local data centers across the globe to perform fast reads and updates, getting around the speed of light limitation.

Why was relational late to the distributed cloud database party? The short answer is that relational databases are, more often than not, considered systems of record, and therefore have critical need for ACID transactions. The long answer is that the classic CAP Theorem, which states that distributed databases can only achieve two of three goals (consistency; availability; and tolerance of network outages), have forced designers to develop sophisticated approaches for delivering the best tradeoffs between consistency and availability for the demands of the specific type of use case.

The need for strict ACID support also explains why approaches for distributed relational databases departed from their NoSQL/non-relational counterparts. A case in point of Azure Cosmos DB, Microsoft's distributed database that has relational (a revamped and rebranded version of the former Azure PostgreSQL Hyperscale offering) and non-relational (NoSQL key-value store, MongoDB, and Apache Cassandra) editions. The Cosmos DB SQL edition retained the original transaction engine from the PostgreSQL Hyperscale service, which was based on the Citus DB extension of the PostgreSQL open source project; conversely, the non-relational editions supported the original Cosmos DB menu of five levels of transaction support.

Oracle's path to distributed databases

As we'll note below, there are many different approaches to distributed databases. Oracle's journey evolved with several architectures. To understand the new globally distributed edition of the Autonomous Database, it helps to understand what came before it. They are each suited for different use cases or scenarios, and therefore, the "new" product does not necessarily replace the old.

Oracle RAC: Clusters for high-performance OLTP and analytics

Oracle RAC debuted with Database 9i back in 2001. Dating before the advent of the modern cloud, Oracle RAC was originally designed for on-premises deployment. Today, it is available as an edition of Oracle Database, part of Oracle Exadata, and as a customer-managed service on OCI. *Oracle RAC is typically used for demanding use cases requiring high performance transaction and analytic processing.*

Oracle RAC places one physical database on a cluster that allows any data to be accessed from any server within it, with frequently accessed data cached locally. Within Oracle RAC, clusters can be scaled both horizontally (adding more nodes) or vertically (using bigger machines), and can be configured to shard data for load balancing However. clustered databases are not designed to work across data centers or regions.

Sharding comes to Oracle Database on-prem

Sharding addresses the horizontal scaling limitations of clustered approaches. Introduced with Oracle Database 12i Enterprise Edition in 2017, data is partitioned and assigned to a specific storage node (or a specific group of multiple nodes for local high availability). That is a major departure from the any-to-any access patterns of clustered architectures. While clustered databases are treated as physical instances, sharded databases can be treated as a single logical instance than spans multiple physical instances. While clustered databases share data and compute, sharded databases are a shared-nothing architecture. *Sharded databases are best suited for transaction processing across data centers or regions where access patterns tend to hit distinct subsets of data – which is common with OLTP systems.*

While sharding and partitioning can be complex, from the standpoint of network topology, it is a simpler and more scalable alternative to clustering because there is no need to manage communications and contention. If partitioned properly, all transactions run against a local shard. The drawbacks of sharding occur when data must be accessed from remote partitions; latency then creeps in; that is why sharding is not well suited for analytic queries that typically cover all of the data and often involve complex joins.

To address analytical queries, Oracle sharding has a query coordinator which enables massively parallel processing by pushing SQL statements in parallel to all shards. Data can be accessed in row or columnar format. Oracle also supports Oracle RAC as a shard which enables high performance analytical queries.

Oracle is hardly the only database supporting sharding. For instance, you can create shards on Amazon Relational Database Service (RDS) and Aurora, but they are not part of the managed service; the necessary routing must be built into the application tier. The same holds true for Google Cloud SQL. By contrast, Oracle builds sharding into the database tier and supports a range of sharding options including:

- **Value-based** data distribution based on a specific value or range of values;
- **System-managed** data distribution where the system automatically manages partitioning through consistent hashing to evenly distribute data; with this mode, the system can add new shards with automated rebalancing of data.
- **User-defined data partitioning,** where the data set is highly skewed (e.g., a predominance of data with a specific value within the range, a feature unique to Oracle Database).
- **Composite data distribution**, which shards by value or range, and then evenly partitions data with that value or range across multiple shards using consistent hashing (a feature unique to Oracle Database) when specific shards are prone to load balancing issues.
- **Duplicated data distribution,** where frequently accessed small tables are replicated as read-only replicas across all shards.

Bringing sharding to the Autonomous Database

With Oracle having supported sharding since 2017, the question was when it would add support to the Autonomous Database. As noted above, Oracle's prime differentiator with sharding is that it built it into the database layer, avoiding the need to write routing logic into the application. However, until now, Oracle customers using sharding had to handle the lion's share of data lifecycle management and operational tasks.

Consequently, with the Autonomous Database, customers opting for sharding no longer have to handle the operational tasks, while much of the data lifecycle management gets automated; for instance, the task of load balancing is taken off the customer's shoulders.

With Oracle Globally Distributed Autonomous Database, the operation is automated while data layout is policy-driven. And it can leverage other features available only through the autonomous database such as Select AI for natural language query.

How Oracle Globally Autonomous Database compares

No silver bullet

There is no single cookie-cutter approach to distributing a database because variations in database use cases necessitate different methods. As noted above, Oracle offers two options: clustering and sharding. In broader perspective, here is a more complete list of variants and where Oracle plays:

- **Geographically sharded databases** for horizontal scale, that deliver a single *logical* database across multiple physical instances. Here, each region has its own primary for data residing in that region. This is the route that Oracle chose for the globally distributed edition of the Autonomous Database (although as we explain below, there are nuances on how reads and writes are scaled inside each local region). A fun fact is that the sharded approach is not restricted to the relational world; for instance, sharding is quite popular among MongoDB customers, which like Oracle, builds routing logic into the database engine. *This is suited for scenarios where the bulk of updates are likely to be confined to specific regions and/or for serving countries that have strong data sovereignty laws.*

- **Clustered architectures** that deploy a single *physical* database across multiple compute and storage nodes, with the goal of avoiding single points of failure (SPOF), with prime examples being Oracle Real Application Clusters (Oracle RAC), Azure Cosmos DB for PostgreSQL (formerly Azure Database for PostgreSQL Hyperscale), and the now-discontinued MariaDB Xpand (formerly Clustrix). *This is suited for use cases requiring high availability and performance for both transactions and analytics.*

- **Primary/secondary architectures** that establish a single logical database, with a single designated write node supplemented with read-only replicas. This is for high availability and lower-latency local reads. Also known as active-passive designs, a prominent example is Amazon Aurora Global Database (for MySQL or PostgreSQL); Oracle also supports such an architecture using replication technologies like Active Data Guard and Oracle GoldenGate. In such cases, writes can be committed quickly; however, there may be some latency for achieving consistency across the distributed read-only local replicas, depending on whether the replicas are within the same data center or in a remote center or region. *This is suited for use cases where the primary need for global distribution is fast, local reads.*

- **Multimaster architectures**, involving a single *logical* database where the primary node for committing a write is chosen on the fly. Also known as "active-active" designs, primaries are typically selected through a consensus protocol involving a polling process of active candidate nodes, and often requires special measures such as atomic clocks or network time protocols for synchronizing transaction sequence across all those masters. Google Cloud Spanner, Cockroach DB, and Yugabyte are prime examples of relational multimaster databases. *This is suited for use cases where changes to all data are likely to happen anywhere.* Oracle also supports this architecture with bi-directional active/active replication with Oracle GoldenGate, which allows data to be replicated to many regions globally.

A fun fact is that many distributed database implementations may utilize a blend of approaches. For instance, they may mix and match vertical (larger compute and/or storage node) and horizontal (adding compute and/or storage nodes) scaling based on the requirements of the node or region. Or a clustered database could also support sharding within specific cluster nodes, or a sharded database that within a shard supports multimaster configurations. As we'll note below, Oracle's new distributed edition of the Autonomous Database mixes and matches several approaches under the hood.

Replication plays pivotal role

Replication, which is typically used for fault tolerance with traditional, single-node databases, grows even more critical for distributed databases. When data is replicated in a distributed database, it enables the local read-only copies of data that improves performance, and also allows more flexibility with load balancing for directing queries to less-used nodes. And of course, replication provides those added data nodes for high availability/fault tolerance.

For multimaster, primary/secondary, or clustered databases, where all nodes are shared, it is about replicating the entire database across all nodes for high availability. Conversely, for partitioned (sharded) databases, which have a shared nothing architecture, replication is about backing up each partition or shard individually for higher availability. Consequently, because there is no single silver bullet approach to distributed database topology and their associated replication approach, comparisons are akin to looking at apples and oranges.

For instance, Google Cloud Spanner, which is a multimaster database, takes advantage of the byte-level replication of the underlying distributed file system and uses the Paxos consensus protocol to determine which replica is the leader. Amazon Aurora, which is based on a primary with read replicas, relies on change stream logs for replicating across a data center, or with the global edition, across regions.

Both Spanner and Aurora differ from Oracle in that they are purpose-built databases, whereas Oracle positions its platform as a converged database. And not surprisingly, Oracle has always offered a range of replication options for its databases, including synchronous replication or ensuring zero data loss or asynchronously if the priority is the design goal is reducing latency. Or replication can be a blend of both modes, such as a network outage or bottleneck triggering a temporary toggle to asynchronous mode to keep a local node operational. The tools offered by Oracle include:

- **Data Guard**, which is bundled with the enterprise edition of the database that is used for physically replicating the database from the primary to read-only standbys for high availability;
- **Active Data Guard**, which adds the ability to keep the read-only standby available for reads during the replication process (and is licensed separately); and
- **GoldenGate**, which is a more flexible logical approach that can be run in a variety of modes including data movement and data transformation, with sources and targets configurable (e.g., one-to-many; many-to-one; or many-to-many). The process can be configured to involve replicating the entire data set or a subset (that is where sharding support comes in). While Data Guard and Active Data Guard are designed principally for disaster recovery stand-bys. GoldenGate is more of a Swiss Army Knife that can be used for DR and/or for synchronizing data across departments and databases, or for replicating subsets of data for populating distributed database shards. And it could also be used in conjunction with Oracle Autonomous Database to make the database globally replicated.

For Globally Distributed Autonomous Database, Oracle will soon introduce a third option based on the emerging RAFT protocol. RAFT is a consensus algorithm that is used for multimaster, active-active replication scenarios. Often compared to Paxos, RAFT is considered a simpler alternative with a relatively straightforward process for electing leaders or masters.

Here's the fine print: there are a couple ways that Oracle could implement RAFT. In one scenario, RAFT will operate *inside* a specific region or partition, implementing multimaster within a shard. This option will be useful for global enterprises with a higher concentration of business within a specific region. Another scenario would implement RAFT across multiple availability zones within a region or across regions that won't breach data sovereignty requirements. Oracle is currently testing this mode with Database 23c beta customers. RAFT support will come when Oracle updates the Autonomous Database (including the globally distributed version) to Database 23c, expected later this year.

Takeaways

Distributed processing came later to relational databases because of their robust requirements for strong transaction consistency. The case for distributed databases is all about unifying the management of data for applications that are global and ensuring access to current data. That case has become more urgent as the cloud has provided the scale and the connectivity to make distributed databases feasible.

When distributed processing came to relational, it took several forms because there was no single silver bullet approach that would fit for all use case scenarios. The old truism that "all politics is local" could well apply to most transaction processing. The market for full multimaster databases has remained narrow; no other startups beyond CockroachDB or Yugabyte have emerged to challenge Google, which created the category with Cloud Spanner. Other forms, such as maintaining a single global primary with multiple, distributed read-only replicas, are a simple alternative when the key requirement is for fast, local reads.

But for writes, the old aphorism of "all politics is local" could well apply to the global bank, financial services, or e-commerce firm that services customers across the world. In most cases, the writes will be against data for specific customer accounts, which are likely (and ideally) to reside locally. Increasingly, in many regions, such as across the EU or in China, that is becoming a legal necessity owing to data sovereignty laws.

Oracle's extension of sharding support in the Autonomous Database is the logical outgrowth. Besides the self-driving nature of autonomous database, Oracle's key advantage is that, unlike many rivals, it builds sharding into the database tier, which makes applications far less breakable and requiring minimal changes. This may be useful, not simply when accounts move across state, provincial, or national boundaries, but also when the volume of business changes markedly within a region. Implementing sharding within the Autonomous database also ensures that scaling of shards up or down is also automated. When business spikes up or down within a region, enterprises can divide or consolidate the local partitioning scheme with the confidence that they will not break the application.

Author

Tony Baer, Principal, dbInsight

tony@dbinsight.io

Linked In   https://www.linkedin.com/in/dbinsight/

About dbInsight

dbInsight LLC® provides an independent view on the database and analytics technology ecosystem. dbInsight publishes independent research, and from our research, distills insights to help data and analytics technology providers understand their competitive positioning and sharpen their message.

Tony Baer, the founder and principal of dbInsight, is a recognized industry expert on data-driven transformation. *Onalytica* named him as a Top Cloud Influencer for 2022 for the fourth straight year. *Analytics Insight* named him one of the 2019 Top 100 Artificial Intelligence and Big Data Influencers. His combined expertise in both legacy database technologies and emerging cloud and analytics technologies shapes how technology providers go to market in an industry undergoing significant transformation. A founding member of The Data Gang, Baer is a frequent guest on *theCUBE* and other video and podcast channels.