Cockroach Labs

CockroachDB vs Amazon Aurora Global Database & Aurora DSQL

A Comprehensive Enterprise Distributed Database Analysis

Published November, 2025

Table of contents

Executive Summary 3–6

Introduction 7–9

Database Architecture Determines Distributed SQL Capabilities 10–14

Resilience, Availability & Disaster Recovery 15–22

Scalability 23–27

Flexible Deployment 28–32

Customer Experience 33–40

Conclusion: CockroachDB: Distributed Without Compromise 41–42

Executive Summary

What enterprise companies *do* varies widely. When it comes to their data, though, what enterprises *need* is remarkably similar:

- **Resilience to maintain uptime**
- **Scalability to manage growth**
- **Deployment that matches their operational parameters**
- **A straightforward and well-supported database experience**

An enterprise database must have few resource constraints to make it capable of delivering strong consistency, low latency, and rock-solid resilience in a manner that matches today's distributed application architectures. The choice between CockroachDB and AWS Amazon Aurora Global Database or Aurora DSQL ultimately comes down to which solution best delivers on these universal enterprise priorities: resilience, scalability, flexible deployment, and customer experience.

Resilience: Performance under adversity

Organizations experience an average of 86 outages each year. CockroachDB's "Performance under Adversity" methodology tests six escalating phases of real-world stress – from internal operational stress and disk stalls to network failures, node restarts, zone outages, and regional outages – measuring throughput, latency, and recovery time throughout. **Resilience is the new performance metric.**

Aurora limitations

Amazon Aurora Global Database's single-writer architecture creates write scalability bottlenecks and requires maintenance windows. Regional failovers can take minutes, and the architecture cannot survive multi-region write scenarios without complex custom orchestration. Aurora DSQL, while offering active-active architecture, is a relatively new solution with significant feature gaps, limited region availability, and no proven track record under production adversity. Both solutions have a significant cloud survivability limitation, that in the event of a full AWS outage they (like any other AWS database service) would experience downtime with no failover across cloud providers.

CockroachDB solution

Distributed SQL architecture with multi-region active-active writes, automatic failover without data loss, and proven resilience through continuous operation during node failures, zone outages, and regional disasters. CockroachDB offers all these features without maintenance windows, thanks to zero-downtime online schema changes and database upgrades, and real-world benchmarks to back it all up.

Scalability: Growth without limits

Enterprises need strongly consistent databases that scale horizontally as demand grows, without complex re-architecting or application downtime.

Aurora constraints

Aurora Global Database requires manual sharding at scale and its single-writer architecture creates bottlenecks for write-intensive workloads. Aurora DSQL's distributed architecture confers scalability, but their use of optimistic concurrency control (OCC) and snapshot isolation can cause performance issues for some workloads with hotspots and concurrent writes. DSQL also requires rearchitecting applications for application retries.

CockroachDB solution

True horizontal scalability with strong consistency, distributed writes across regions, automatic data rebalancing, and the ability to scale up or down by adding or removing nodes without downtime or application changes. Scales from startup to enterprise without refactoring.

Flexible deployment: Needed now more than ever

A database's core design affects every aspect of application infrastructure. A database built around a single primary writer cannot suddenly become multi-region active-active; a system designed exclusively for one cloud provider cannot magically support multi-cloud or hybrid deployments. **These architectural decisions are permanent, meaning your database choice today determines your deployment options tomorrow.** And deployment flexibility matters now more than ever, as demonstrated by the October 2025 AWS outage that took down more than 70 AWS cloud services, including Aurora DSQL.

Aurora deployment limitations

Amazon Aurora Global and DSQL databases can only run on AWS platforms and are not compatible with hybrid, self-hosted, or multi-cloud implementations.

CockroachDB solution

Runs anywhere – public cloud, private cloud, hybrid environments, and on-premises. Cloud customers can run on our cloud, or their own. Its cloud-agnostic architecture supports single-region, multi-region, and even multi-cloud configurations for global scalability and data sovereignty, with cloud-native deployment on Kubernetes simplifying operations.

Customer Experience: Database migrations, operations, and user support

Enterprises can't accept limitations – they need databases that operate with minimal friction and vendors who provide genuine partnership when challenges arise. They also need high-quality dedicated tooling, along with knowledgeable engineering support, for migrating and operating their database.

⊗ Aurora challenges

For Aurora Global Database, single-writer architecture requires planned maintenance windows with potential downtime, necessitates manual sharding at scale, and offers limited multi-region write capabilities.

For Aurora DSQL, severe PostgreSQL compatibility gaps (no foreign keys, triggers, sequences, extensions, or stored procedures), 3,000-row transaction limit, lack of point-in-time recovery (PITR) for backups, no user control over upgrade timing, and potentially extensive application refactoring make it suitable only for simple greenfield applications.

For migrations to either database, AWS requires using the paid Amazon Data Migration Service (DMS), often necessitating expensive consulting engagements for complex implementations. Both lock organizations into technical support that some AWS customers have described as difficult to access and favoring infrastructure over database expertise.

✓ CockroachDB advantages

Operationally CockroachDB has zero-downtime online schema changes, rolling upgrades, and rolling repaves, true multi-region write capabilities with horizontal scaling, incremental locality-aware backups with granular table/database/cluster-level restore options, and point-in-time recovery.

PostgreSQL compatibility covers the majority of real-world use cases while avoiding vendor lock-in, and the free MOLT toolkit and our experienced services organization support migrations at every step.

CockroachDB customer support emphasizes direct engineering access with short escalation paths, proactive architecture guidance, and achieved 97% customer recommendation ratings in 2024 Gartner reports, earning Gartner Customers' Choice recognition in both 2023 and 2024.

Summary of our comparative analysis

The detailed and in-depth multi-vector comparisons in this report definitively show that, despite what its name suggests, Amazon Aurora Global Database is limited by its own single-region write architecture. It best serves organizations that choose to prioritize absolute PostgreSQL compatibility and can accept eventual consistency, data latency, scalability ceilings, and the potential risk of data loss in failover scenarios.

The findings further show that, while Aurora DSQL is indeed a step forward for Amazon databases there are still serious shortcomings. When it comes to supporting modern distributed application architecture, DSQL's geographical constraints and far-from-complete Postgres compatibility keep it from being a serious contender for implementing truly global, enterprise-scale workloads.

For enterprises requiring mission-critical reliability, global scalability, deployment flexibility, and operational simplicity, CockroachDB's natively distributed architecture is the optimal match for modern distributed applications. As the database that established the distributed SQL category, CockroachDB makes sure companies are never again forced to choose between global scale, consistency, and high availability in their database.

Introduction: Why Enterprises Choose Distributed SQL

Enterprise organizations with a global footprint – or even those in a single country but spanning multiple cloud provider availability zones (AZs) – need data architectures that deliver always fast, always correct transactional data, and that also scale organically and deploy flexibly. But traditional RDBMSs were never designed to support modern distributed applications, forcing companies to make difficult tradeoffs between performance, availability, cost, and data integrity.

Distributed SQL eliminates these tradeoffs. This mature technology combines enterprise-grade relational database capabilities with horizontal scalability and cloud-native resilience, providing:

- **Always-on operations in a 24/7 world:** Distributed SQL databases are designed to be resilient and highly available. They continue to operate even if one or more nodes fail, surviving availability zone and even regional data center outages.
- **Global scale without compromise:** Distributed SQL databases scale natively and horizontally simply by adding more nodes to handle increased load, or removing them when traffic is light. Truly distributed SQL databases also eliminate the traditional PostgreSQL bottleneck of routing all writes to a single primary server. And, they should be capable of deployment on any infrastructure, whether in the cloud, on-premise, or a hybrid implementation.
- **Declarative data placement and survival goals:** Native data placement capabilities help ensure data compliance and the ability to place data near to users for improved performance.

Distributed SQL databases are not all the same

This paper compares and analyzes differences between three major enterprise distributed SQL databases. First, let's meet the candidates.

CockroachDB

A truly global, natively distributed SQL database, CockroachDB's active-active multi-region architecture enables writes and strongly consistent reads from any region, without need for a single primary.

- **PostgreSQL compatible database** – designed from the ground up for cloud-native resilience, horizontal scalability, and automated operations.
- **Enterprise-grade with minimal operational overhead** – automatic sharding, rebalancing, and recovery with no manual intervention required for scaling or failover. No downtime for schema changes or software upgrades.
- **Strong, global consistency** – CockroachDB's serializable isolation means all nodes and regions can serve consistent reads and writes.

Amazon Aurora Global Database

Aurora Global Database is an advanced Aurora feature extending a single Aurora database to span multiple AWS regions, achieved by bolting a distribution layer on top of Aurora RDS to add read-only clusters in other regions.

- **Multi-region PostgreSQL-compatible database** – active-passive architecture spanning up to 10 secondary regions, optimized for disaster recovery and global read access.
- **Single-writer design** – all writes must go to the primary region, with secondary regions providing eventually consistent (~<1s lag) read replicas and managed cross-region failover.
- **Eventually-consistent database** – built on Aurora's decoupled storage and compute architecture with 6x storage replication, offering familiar PostgreSQL compatibility and RDS tooling.

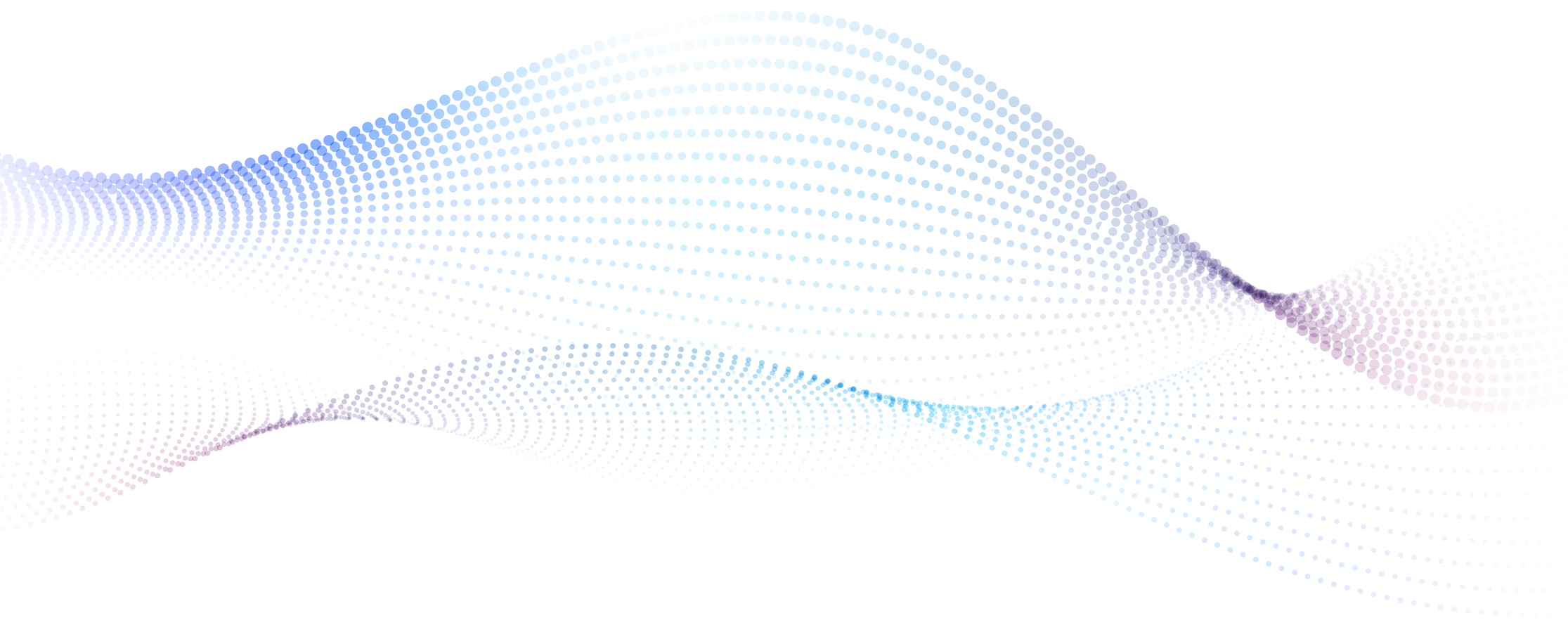
Despite its name, Amazon Aurora Global Database is mostly an option for single-region deployments. It's optimized for read-heavy workloads when write scalability can be limited to a single master node in a single region. No matter how it's deployed, Aurora Global Database will suffer latency issues with global writes as its single write node is always tied to a single region.

Amazon Aurora DSQL

Aurora DSQL is not just Amazon's recently-launched distributed SQL database offering (GA as of May 2025). It's also an acknowledgement from AWS that distributed SQL is a long-term solution that solves many business-critical problems.

- **Serverless distributed SQL service** – multi-region active-active architecture, introduces AWS's newest approach to always-on, globally distributed online transaction processing (OLTP) databases.
- **Fully managed serverless platform** – automatic scaling across separated architectural layers and requiring zero infrastructure management.
- **Limited availability and significant restrictions** – currently restricted to specific region pairs (not truly global), with notable feature limitations including no foreign keys, triggers, or extensions, and a 10TB default storage cap.

Now let's put the three solutions to the test. First we will examine the architectural differences between CockroachDB, Amazon Aurora Global Database, and Aurora DSQL. Then we'll analyze and compare how each of these databases delivers on reliability, scalability, deployment, and customer experience.



PART 1:

Database Architecture Determines Distributed SQL Capabilities

The fundamental architecture of a distributed SQL database determines not just whether it can survive failures, but how quickly it recovers, how much data might be lost, and whether your business continues operating seamlessly during outages.

CockroachDB architecture

CockroachDB's architecture is a layered, distributed system designed for scalability, strong consistency, and high availability (HA), inspired by Google's Spanner and F1. CockroachDB has just three components, an efficient design that separates storage and compute, with a SQL layer on top.

1. **SQL layer:** This is the highest level, providing a familiar Postgres-compatible SQL API to developers. It handles parsing SQL statements, optimizing queries and translating them into low-level key-value operations.

2. **Transaction layer:** This compute layer ensures ACID (Atomicity, Consistency, Isolation, Durability) properties for all transactions. It manages distributed transactions across multiple nodes and uses a concurrency control mechanism to maintain strong consistency.

3. **Key-value store:** A distributed, replicated key-value (KV) store is the foundation of CockroachDB. This layer is responsible for:

   - **Data distribution:** Data is automatically sharded into "ranges," which are small, contiguous chunks of the key-space. These ranges are distributed across the cluster's nodes.

   - **Replication and consensus:** Each range is replicated across multiple nodes (typically 3 or 5) for fault tolerance. CockroachDB utilizes the Raft consensus protocol to ensure that all replicas of a range are consistent and to handle leader election and failure recovery.

   - **Data storage:** The KV store uses a persistent storage engine to store data on disk within each node.

CockroachDB key principles

- **Shared-nothing architecture:** Each node is independent and can handle requests. Adding more nodes to the cluster automatically distributes data and workload, allowing for seamless horizontal scaling.

- **Strong consistency:** Raft consensus and a distributed transaction manager ensure that all nodes always see the same, up-to-date and transactionally consistent data.

- **High availability and fault tolerance:** Data replication and the Raft consensus protocol enable the system to survive node, rack, or even data center failures with minimal disruption.

- **Active-active multi-region architecture:** Data is synchronously replicated and transactionally consistent across *all* regions.

Amazon Aurora Global Database architecture

Aurora Global Database uses a primary-secondary replication model: a key primary region handles all write operations, with one or more secondary regions as read-only replicas. Aurora Global Database's design rests on four pillars:

1. **Primary region:** This is the only region where write operations can be performed. The writer database cluster for the global database resides here.
2. **Secondary region(s):** Data is replicated to secondary read-only clusters with typical latency of ~1 second, providing only eventual consistency for read replicas.
3. **Global database cluster:** This is a logical container that links the primary and secondary clusters together across different regions.
4. **Replication layer:** Dedicated network infrastructure replicates data between regions as an asynchronous process.

Aurora Global Database key principles

- **Single-writer architecture:** The database has a single primary Amazon Aurora database cluster in one AWS region that processes all write operations.
- **Up to ten read regions:** The primary cluster's data can be asynchronously replicated to as many as ten secondary, read-only database clusters in different AWS regions.
- **Storage redundancy:** Within each region, the shared storage volume is automatically replicated six ways across three availability zones. With a recovery point objective (RPO) of typically one second, **most** data is preserved during an outage.
- **Localized data access:** Each secondary region can support up to 16 read-only database instances that can provide low-latency access to read-only data for local users.

Aurora DSQL architecture

Aurora DSQL's architectural design distributes functions into six functional components:

1. **Transaction and session router:** This frontend component handles incoming client connections and the PostgreSQL protocol, routing queries to the appropriate processors.
2. **Query processor:** Responsible for parsing and planning SQL queries, it uses the Amazon Time Sync Service for snapshots and executes queries directly on the storage layer.
3. **Adjudicator:** This component implements optimistic concurrency control and ensures global consistency, validating transaction commits and managing conflict resolution.
4. **Journal:** A distributed log that maintains a record of all transaction changes to provide durability and enable point-in-time recovery.
5. **Storage layer:** A shared-nothing, distributed system that manages data persistence and partitioning, automatically rebalances data across nodes, and handles data replication.
6. **Control plane:** Coordinates the various components, maintains redundancy across multiple AZs, and manages automatic scaling and self-healing in case of failures.

Aurora DSQL key principles

- **Serverless and fully managed:** Aurora DSQL is designed to operate with zero infrastructure management for users and usage-based billing with no charges when idle.
- **Optimistic concurrency control:** Instead of traditional locking mechanisms, Aurora DSQL uses an OCC model to handle concurrent transactions.
- **Single database, multiple regions:** Aurora DSQL presents as one logical database spanning multiple regions *within a single continent* (North America, Europe, etc.), with writes propagating instantly via journal log replication.
- **Independent scaling:** DSQL's component architecture allows the compute, read/write, and storage layers to scale separately, providing flexibility for varying workload demands.

Architecture is destiny

Amazon Aurora DSQL's disaster recovery capabilities approach CockroachDB's architectural sophistication with synchronous replication and no single point of failure. However, its deliberate geographic limitations prevent truly global deployment – a constraint that may never change, given Amazon's design choice to limit cross-continent clusters.

Amazon Aurora Global Database's slower failover, potential data loss, and single-writer bottleneck are all factors that could represent material business risks in disaster scenarios.

Capability	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Architecture	True active-active multi-region architecture; serverless or dedicated hardware	Active-passive with single-writer bottleneck	Active-active within region sets, serverless with separated layers
Replication model	Synchronous with consensus-based quorum	Asynchronous global replication with ~1s typical lag; in-region read replicas have a replication delay of 10s of milliseconds.	Synchronous with 3-region quorum (journal-based)
Read consistency	Strong consistency everywhere globally	Strong consistency in primary region only; eventually consistent reads in secondary regions (read replicas read data up to 100ms stale)	Strong consistency only within region sets
Geographic flexibility	Deploy anywhere globally without restrictions; can deploy multi-cloud or hybrid options for regulatory compliance or better survivability	Up to 10 secondary regions, but all write traffic routes to single primary region	Strong within region pairs (e.g., all North America OR all Europe); cross-continent deployment not supported
Data loss risk	No data loss on failover	Potential data loss on unplanned cross-region failover due to async replication	No data loss due to synchronous replication

PART 2:

Resilience, Availability & Disaster Recovery

Traditional SQL databases once forced enterprises to treat downtime as an acceptable business risk, until distributed SQL arose to virtually eliminate that risk. But **a distributed SQL database's resilience varies according to how its architecture dictates availability and disaster recovery.**

Disaster recovery

Disaster recovery focuses on restoring database operations after major incidents. This includes data loss prevention and restoration, backup strategies, and recovery time objectives.

CockroachDB: Self-healing architecture

CockroachDB's self-healing capabilities leverage automatic rebalancing and data replication, continuously monitoring node health and redistributing data across surviving nodes when failures occur. Beyond built-in Raft replication and fault tolerance, additional capabilities minimize downtime and data loss:

- **Multi-active, multi-region design:** Architected for resilience with deployment across multiple availability zones, regions, cloud platforms, or hybrid environments to eliminate single points of failure.
- **Point-in-time data recovery:** Incremental, locality-aware backups and precise point-in-time recovery (PITR) with granular restore at the table, database, or cluster level deliver faster recovery times and lower RPO than traditional snapshot-based approaches.
- **Physical cluster replication (PCR):** Continuously replicates all cluster-level data from primary to independent standby cluster asynchronously for two data center deployments. Failover stops replication, resets the standby to a consistent point in time, and makes it ready to accept application traffic.
- **Automatic recovery without intervention:** CockroachDB's self-healing architecture automatically rebalances data and maintains availability when nodes fail – no manual intervention or complex failover procedures required.
- **Strong consistency across all regions:** CockroachDB's default SERIALIZABLE transaction isolation ensures all reads and writes are consistent across all nodes. This eliminates the data reconciliation challenges that plague eventually-consistent systems during disaster recovery.

The upshot: CockroachDB provides continuous, automatic protection through synchronous replication and fast, granular restore capabilities. No external disaster recovery (DR) planning or manual failover (and failback) is necessary.

Amazon Aurora Global Database: Snapshot-based recovery with manual intervention

Because it requires explicit planning for failover, failback, and reconciliation, Aurora does *not* self-heal across regions.

- **Daily Snapshots with manual PITR setup:** Aurora Global Database relies on daily snapshots for backup, with manual intervention needed for point-in-time recovery setup. This traditional approach requires planning and additional operational overhead.

- **Data loss risk during unplanned failover:** Aurora achieves less than one minute RTO with zero data loss during *planned* switchovers when promoting a secondary region to primary. However, unplanned failover can cause data loss due to asynchronous replication, requiring application teams to implement reconciliation processes to verify data was copied between regions – a time-consuming and error-prone task.

- **Operational complexity:** Regional failover with Aurora requires ensuring both application and database can failover *and* failback successfully. Failback is a manual process and teams must maintain reconciliation processes to verify data consistency.

The upshot: Amazon Aurora Global Database provides solid intra-region durability but limited cross-region DR. Failovers may cause short downtime and potential data loss; manual coordination and post-outage data consistency is required.

Aurora DSQL: Physical synchronous replication with limited recovery options

- **Synchronous replication without data loss:** Aurora DSQL provides physical, synchronous replication with strong consistency across regions. All writes are replicated via consensus using a three-region quorum, eliminating the data loss concerns of Aurora Global Database's asynchronous model.

- **Backup features recently added:** DSQL now includes centralized backup management, full cluster backups, automated schedules, cross-region and cross-account support, and WORM (write-once, read-many) protection for compliance scenarios.

- **Full snapshot limitations:** Despite these additions, Aurora DSQL relies on full snapshot backup/restore via AWS Backup with no point-in-time recovery (PITR) capability, potentially impacting both RPO and RTO during disaster

The upshot: Aurora DSQL introduces automated, synchronous DR – an improvement over Aurora Global Database – but remains geographically constrained, has limited PostgreSQL compatibility, and lacks a track record of proven real-world resilience.

High availability data comparison

High availability (HA) ensures continuous database operations despite component failures, minimizing downtime and maintaining service continuity.

CockroachDB can achieve up to five nines (99.999%) availability, which is equivalent to approximately 5 minutes of downtime per year.

This high availability is a core feature, especially for multi-region deployments, and is achieved through CockroachDB's multi-active replication with automatic rebalancing, creating a system that continuously monitors node health and automatically redistributes data across surviving nodes if one goes down. CockroachDB's natively distributed architecture and the use of the Raft consensus protocol ensure data consistency and fault tolerance even during an availability zone – or even full region – outage.

- **Sub-10 second failover:** If you lose a node in CockroachDB, automatic failover occurs in less than 10 seconds with zero data loss.
- **Advanced fault tolerance:** CockroachDB can withstand multiple simultaneous node failures while maintaining operations, as long as quorum is maintained for each replica and data range across the cluster.
- **Logical data replication (LDR):** For more complex availability scenarios, CockroachDB supports logical data replication. This enables sophisticated data distribution patterns and additional resilience options for enterprise architectures in self-hosted environments.

> **The upshot:** CockroachDB's always-on, multi-active design avoids data loss and provides near-instant continuity. Availability is architected into every node, not bolted on.

Aurora Global Database' single-write-master replication model

This means that only one node in the cluster can receive writes. While on the surface it may appear that Aurora offers high availability because it can run across data centers, the reality is very different.

- **Choice between manual failover or data loss:** Teams using Aurora might choose manual failover because Aurora's automatic cutover is vulnerable to potential data loss.
- **Failover latency:** Within a region, Aurora auto-failovers in approximately 30 seconds. Cross-region failovers take 1-2 minutes – up to *ten times* longer as compared to CockroachDB.
- **Multi-AZ failover with service disruption:** Aurora Global Database provides multi-AZ failover capability, but failover takes time and can cause downtime.
- **Single-region constraint:** As David Phillips, Starburst co-CTO, noted: "Amazon Aurora is tied to a single region. While it provides a standard SQL Database and nice features, if the region goes down the application goes down and it also cannot provide the latency guarantees like CockroachDB can."

> **The upshot:** Amazon Aurora Global Database provides solid read availability but limited write availability; automatic failovers incur latency, delays, and potential data loss.

Aurora DSQL features an active-active, no-manual-failover architecture

This stateless design gives no possible single point of failure. If one region or AZ goes down, transactions can continue in others with no data loss due to quorum replication at the journal level.

- **Availability claims:** DSQL claims to be designed for 99.99% availability in single-region deployments and 99.999% availability in multi-region deployments, matching CockroachDB claims. The failover process entails simply switching to an available region endpoint with all data current.
- **Geographic constraints:** DSQL cluster regions are confined to a single region set (e.g., all in North America OR all in Europe). This is a significant limitation for enterprise applications with users all around the world.
- **Lack of production validation:** Critically, DSQL is an immature product with no proven track record. While the architectural design appears sound, the lack of extensive production validation in diverse real-world scenarios means its high availability and enterprise capabilities remain largely unproven.

> **The upshot:** Aurora DSQL's HA model closely resembles CockroachDB in design goals (active-active, zero-loss). However it's new, geographically limited, and lacks real-world validation.

Resilience: Summary findings

In summary, **CockroachDB** delivers both disaster recovery and high availability as native, always-on capabilities: Every node is active for reads and writes, data is synchronously replicated across regions, and recovery from node or region failure occurs automatically with no data loss or downtime.

Amazon Aurora Global Database, by contrast, relies on asynchronous replication, manual failover, and snapshot-based backups. This means outages or region failures could result in service disruption and potential data loss.

Aurora DSQL narrows this gap with active-active architecture and synchronous replication, but its limited regional scope and lack of proven production resilience could make it less reliable for enterprises requiring global continuity and deployment.

Table: Disaster recovery comparison (November 2025)

Capability	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Backup type	Incremental, locality-aware	Daily snapshots	Full snapshots
Recovery complexity	Automated	Manual intervention required	Simplified but limited options
PITR support	Yes, native	Manual setup required	No
Granular restore	Table/database/cluster level	Cluster level	Cluster level
Data loss risk	Zero	Possible during unplanned failover	Zero (for replication)
RPO	Minimal (PITR + incremental backups)	~1 second (replication lag)	Zero (replication), higher (backup-only)
RTO	Fast (targeted recovery)	Slower (full snapshot restore)	Slower (full snapshot restore)

Table: High availability comparison

Capability	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Architecture	Multi-active replication	Single-primary with read replicas	Active-active
Failover type	Automatic, self-healing	Manual (to prevent data loss) or automatic (with data loss)	Automatic
Failover time	<10 seconds	30 seconds (intra-region), tens of seconds to minutes (inter-region)	Near-instant (endpoint switch)
Data loss on failover	Never	Possible with automatic failover	No
Node failure response	Automatic rebalancing	Replica promotion	Endpoint switching
Regional failure	Survives with no interruption	Requires manual intervention	Survives within region set

What is Performance under Adversity?

Current industry benchmarks like the TPC-C (initially established in 1992) evaluate maximum performance under controlled laboratory conditions, isolated from real-world unpredictability.

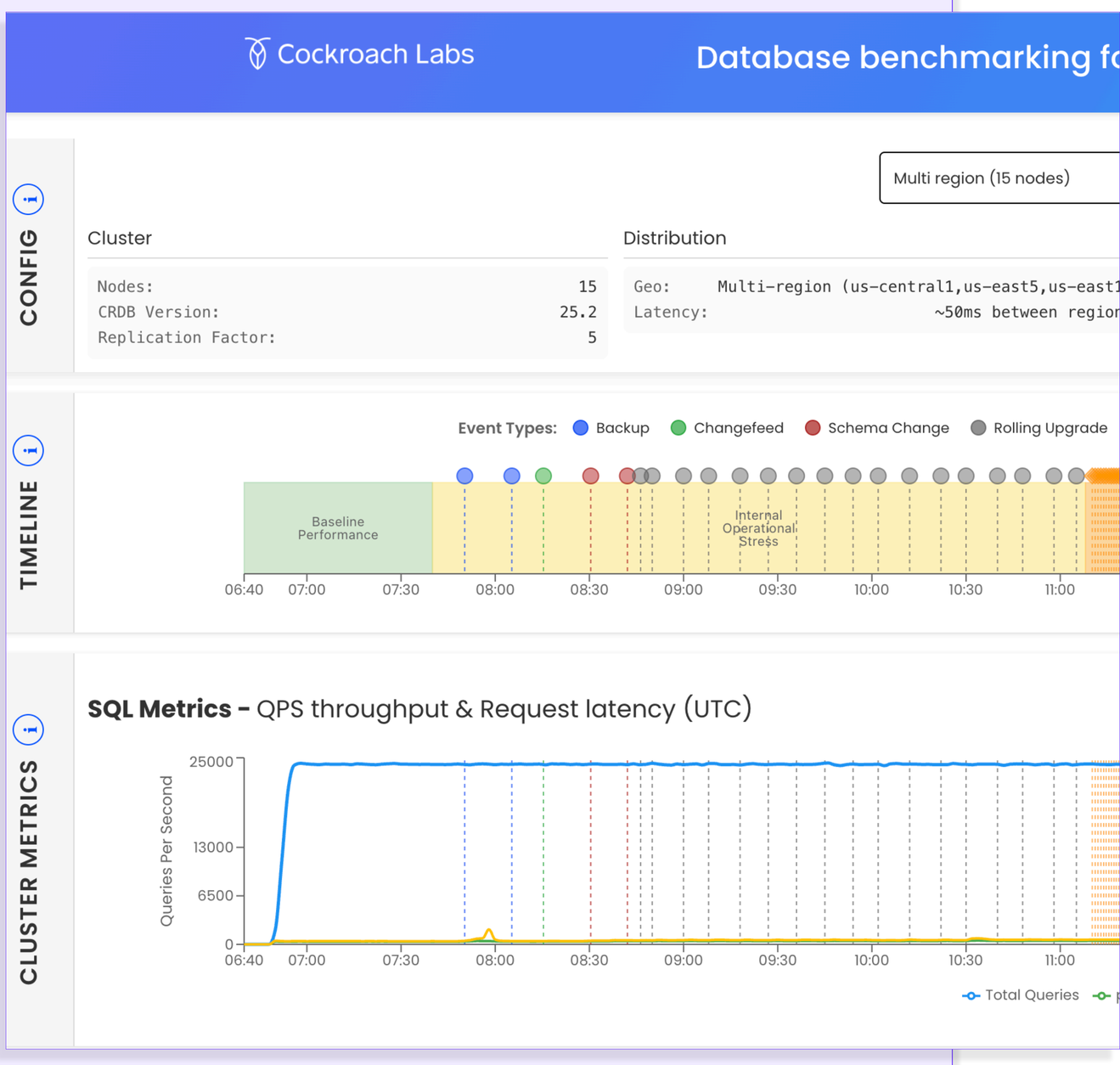
These benchmarks presume single-data center setups, overlooking the complexities of globally distributed applications and data. They may incorporate durability testing but typically don't directly assess performance during outages or planned maintenance, nor do they gauge how quickly systems recover following failures. When the stable operational phase is interrupted by something like a simulated power outage, the benchmark is deemed complete. That's not how production works.

Resilience is the new performance metric

CockroachDB has released a novel "Performance under Adversity" pressure-testing methodology that measures the impact of real-world conditions faced by application and infrastructure teams every day.

The benchmark includes six escalating phases of adversity beyond baseline level in a single continuous run, each introducing progressively harsher failure conditions and operational stress:

1. **Baseline performance:** Measure steady-state throughput under normal conditions
2. **Internal operational stress:** Simulate resource-intensive operations such as change data capture, full backups, schema changes, and rolling upgrades
3. **Disk stalls:** Randomly inject I/O freezes to evaluate storage resilience
4. **Network failures:** Simulate partial and full network failures preventing one partition from communicating with nodes in another partition
5. **Node restarts:** Unpredictably reboot database nodes (one at a time) to test recovery time and impact to performance
6. **Zone outages:** Take down an entire availability zone (AZ)
7. **Regional outages:** Take down an entire cloud provider region



Through all these phases, the methodology measures throughput in transactions per minute (tpmC), latency (90th and 95th percentile), and recovery time to baseline.

Peak performance in perfect conditions is meaningless if your database fails when it matters most. A database must prove it can survive the unexpected (hardware failures, network partitions, regional or even entire zone outages) while maintaining consistent performance. **In creating and constantly measuring its own performance under real-world stress, CockroachDB is setting the standard for databases.**

PART 3:

Scalability

Scaling a database to handle today's large and ephemeral data loads is challenging. Scaling a relational database with strict ACID transaction requirements in a highly distributed environment is even more so, because guaranteeing data consistency and isolation gets increasingly difficult as nodes are added to handle increasing demand.

Just as architecture determines database resilience, it also determines how (and how well) a database scales. Let's examine how CockroachDB, Amazon Aurora Global Database, and Aurora DSQL each handle scaling for writes, reads, data volume, and scaling operations for the database itself.

Scaling database writes

CockroachDB's multi-active architecture enables multi-node writes across all nodes in any region with simultaneous writes in multiple regions for true active-active global deployments. Strong consistency is maintained through consensus-based replication using the Raft protocol. Write capacity scales horizontally by adding nodes with automatic data rebalancing. With no single primary dependency, all nodes can accept writes, enabling elastic scaling for write- or read-intensive workloads without manual sharding.

Aurora Global Database has a primary instance in one region accepting writes. Secondary regions are read-only. Write scaling requires vertically increasing instance sizes and manual sharding as workloads increase. While Aurora scales reads globally, clients in different regions experience high write latencies, and fine-grained geographic data placement is not supported.

Aurora DSQL improves on Aurora Global Database's write limitations with an active-active distributed architecture supporting multi-region writes and automatic sharding. However, DSQL is constrained to only two active regions plus one witness region in multi-region configurations, and is limited to single-continent deployments.

> **The upshot:** CockroachDB is purpose-built for write-intensive, globally distributed applications with no artificial limits, while Amazon Aurora Global Database fundamentally cannot scale writes beyond a single powerful instance. Aurora DSQL improves on this but deliberately sacrifices global reach for reduced latency.
>
> This architectural difference explains why customers migrate from Aurora to CockroachDB when they hit scaling limits: CockroachDB's design eliminates the write bottleneck entirely rather than just moving or minimizing it.

Scaling database reads

CockroachDB provides unlimited global read distribution with no practical limits on replicas or regions. Follower reads enable low-latency local access, while GLOBAL table locality serves infrequently updated data needing worldwide access and REGIONAL BY ROW tables enable geo-partitioned, granular data placement.

Aurora Global Database has strong multi-region read scaling capabilities, with up to 16 read replicas per region and up to five secondary read-only database clusters in different regions. Fast read failover to other regions is supported, with storage-level replication maintaining typically less than one-second lag and independent cluster sizing for each region.

Aurora DSQL provides improved read scaling over standard Aurora with automatic scaling across regions and multi-region architecture maintaining strong consistency without stale reads. Synchronous replication uses three-region quorum consensus, while serverless autoscaling adjusts capacity based on load.

> **The upshot:** Amazon Aurora Global Database provides strong read capabilities within architectural boundaries (five regions, 16 replicas each), while Aurora DSQL adds strong consistency with serverless convenience (limited to two regions). CockroachDB removes all practical limits: It provides expansive read scaling across unlimited regions with flexible consistency guarantees and granular geographic data placement controls.

Scaling data volume

CockroachDB removes data capacity constraints with no hard data limits, growing continuously as needed. Automatic sharding and data distribution require no manual intervention, with data automatically splitting into smaller ranges and rebalancing across nodes as the cluster expands, significantly reducing operational complexity and eliminating planned downtime.

Aurora Global Database has hard capacity limits of 256 TB maximum per cluster and max table size of 32TB. Once limits are reached, users must manually shard the database (causing planned downtime), necessitating:

- added application logic to route queries
- operational overhead from monitoring and rebalancing shards
- data consistency challenges maintaining transactional integrity across shards
- performance management difficulties when uneven data distribution creates “hot spots.”

Aurora DSQL has restrictive capacity limits with a default limit of 10 TiB of total storage and maximum 128 TiB with approved limit increase. Additional limitations include caps on database and column counts, DML capped at 3,000 rows per transaction, connections timing out after one hour, and single built-in database only.

> **The upshot:** CockroachDB’s seamless, elastic, and automatic growth means applications never hit a capacity wall. Aurora Global Database’s hard limits force a painful transition to manual sharding that increases engineering burden and downtime precisely when applications are experiencing the most growth. Aurora DSQL’s severe restrictions make it unsuitable for data-intensive applications, with limits that many production workloads would quickly exceed. This explains why customers migrate from Aurora to CockroachDB to avoid manual sharding at scale.

Scaling database operations

CockroachDB can be upgraded one node at a time with no application impact. The system scales seamlessly with zero downtime by adding nodes on-the-fly with automatic rebalancing. Changes and maintenance operations like rolling upgrades, rolling repaves for OS/security updates, and online schema changes can all take place without taking the database offline.

Aurora Global Database requires planned downtime for schema changes, updates, and major operations. Rolling upgrades require planning and architecture changes, with the single-writer model adding complexity for rolling operations. Database sharding adds manual work and complexity for updates or upgrades.

Aurora DSQL, fully managed and serverless, provides autoscaling for compute and storage independently and automatic sharding with no manual intervention. However, scaling DSQL requires cost monitoring and management. And scaling geographically is limited by DSQL's limited regional support.

The upshot: CockroachDB's architecture enables continuous delivery, so schema changes and upgrades happen transparently during business hours without user impact. Aurora Global Database forces teams to schedule maintenance windows and coordinate complex failover procedures, slowing development velocity; as the database grows, so does the need for manual sharding. Aurora DSQL reduces operational burden through serverless automation but at the cost of flexibility and control, making it suitable mainly for organizations committed exclusively to AWS infrastructure.

Head-to-head comparison: Scaling

CockroachDB scales both reads and writes seamlessly; elastically scales and rebalances as data loads increase; and requires no downtime for database scaling operations

These characteristics mean CockroachDB can scale in tandem with all your SQL applications across their entire lifecycle, for all types of data. This includes mission-critical workloads for financial services organizations like account transactions and payment processing systems, but also working with a broader array of operational data when, for example, running dev/test applications. CockroachDB can be used for existing SQL applications that rely on PostgreSQL compatibility including isolation level, or for net-new applications that need high availability and strongly consistent data.

Strong for:

- ✔ Write-heavy workloads, horizontal scaling, high-growth applications, global distribution
- ✔ Zero-downtime operations, automatic data distribution, operational simplicity
- ✔ Multi-region consistency, eliminating single points of failure

CockroachDB's distributed architecture is purpose-built for applications that must scale writes and reads simultaneously across global deployments, without operational overhead or architectural limitations.

Amazon Aurora Global Database scales reads well, especially if eventually consistent data is acceptable, but scales writes poorly, has hard capacity limits, and requires planned downtime for scaling operations

Aurora's single-write-master architecture makes it unsuitable for write-intensive or high-growth applications that need to scale both reads and writes globally. Many customers don't discover this limitation until they hit the scale wall, and customers have migrated specifically because they hit scale limitations.

Strong for: ✔ Read-heavy workloads, global read distribution, fast regional reads

Weak for: ✖ Write-heavy workloads, horizontal write scaling, avoiding manual sharding, limiting planned downtime for maintenance operations like schema changes or software updates

Watch out for: ⚠ Write bottlenecks, 256 TB storage limit, operational complexity (including data sharding) at scale

Amazon Aurora DSQL scales better than Aurora Global Database for writes, but is limited for large-scale workloads and requires refactoring applications for retries and for limited PostgreSQL compatibility

While Aurora DSQL scales by automatically distributing data, the use of OCC and snapshot isolation level can cause performance issues for some workloads, including concurrent writes. Additional schema planning and design is required to ensure isolation and mitigation of snapshot-related rollbacks. OCC aborts will occur when two or more concurrent transactions attempt to write to the same keys, impacting overall database performance and latency, and also requires applications to be rewritten to add application retries. Because PostgreSQL compatibility is limited, application-level changes may be required when migrating to DSQL.

Strong for: ✔ AWS-native apps, bursty/low baseline workloads, eliminating single-writer bottleneck

Weak for: ✖ Multi-cloud/hybrid strategies, mature production use cases, PostgreSQL feature parity

Limited for: ⚠ Cross-continent deployments, large-scale data (>10 TiB default), high transaction volumes

PART 4:

Flexible Deployment

Database architecture fundamentally determines where and how you can deploy your applications. The choices made in a database's core design – single-writer vs multi-active, cloud-agnostic vs cloud-provider-specific, monolithic vs distributed – create constraints and opportunities that ripple through every aspect of your infrastructure strategy. A database architected around a single primary writer can't be transformed into a multi-region active-active system, and one created for a single cloud provider can't be configured to support hybrid or multi-cloud environments. These foundational design choices are lasting, meaning **your database decisions today shape your deployment flexibility tomorrow.**

> Deployment flexibility matters now more than ever. Case in point: the October 2025 AWS outage that took down more than 70 AWS cloud services, including Aurora DSQL. The extended outage disrupted more than 1,000 businesses for 15 hours and caused an estimated $100bn in economic damage. This is a stark illustration of concentration risk in digital infrastructure, where a single DNS resolution failure in one AWS region cascaded globally and paralyzed everything from airline operations to major financial institutions to healthcare organizations (not to mention Snapchat).
>
> Organizations operating solely within the affected AWS region had no failover option when the outage struck, leaving them completely helpless, but the blast radius was much larger than aws-us-east-1. Organizations with multi-region AWS deployments still experienced service degradation and rolling outages, including overseas users like the UK government, but many were able to recover more quickly with fewer service disruptions. For enterprise companies that want to guarantee operational resilience, outages like these only cement the need for multi-cloud implementation options and vendor-agnostic data architectures.

Deployment: 2DC vs 3DC+

When comparing an Amazon Aurora two-data center deployment (2DC) with a CockroachDB three-data center (3DC) deployment, the key distinction is between an active-passive and an active-active architecture. Amazon Aurora Global Database uses a single-writer model, while CockroachDB is a fully distributed, multi-writer database designed for multi-region active-active operations.

CockroachDB uses the Raft protocol, a quorum-based consensus model, to tolerate complete data center failures while continuing to accept reads and writes and maintaining strong consistency through majority consensus. This 3DC+ configuration enables self-healing with zero operator intervention and distributed writes across all data centers simultaneously. However, CockroachDB also enables 2DC configurations through physical cluster replication (PCR).

Amazon Aurora Global Database functions as primary-secondary replication with only one region having write capability. In a 2DC setup, Aurora cannot distribute write load between locations, forcing vertical scaling by adding more machines to the primary cluster.

Multi-region deployments

CockroachDB eliminates both Aurora's single-writer constraint and DSQL's geographic limitations.

CockroachDB's multi-region deployments put the database where the users are, automatically partitioning to ensure data locality while the database looks (and feels) like it's running right in their region.

The system supports unlimited regions with no hard limit, and all regions can accept writes with every node accepting both read and write operations. CockroachDB clusters routinely span continents with strong consistency, maintaining serializable isolation by default across all regions, and with global ACID guarantees even when transactions span regions. Sophisticated declarative data placement includes REGIONAL BY ROW for localized customer data, GLOBAL tables for reference data replicated everywhere, and smart routing where reads use the nearest replica while writes go to nodes where data is housed.

Aurora Global Database trades off eventual consistency for geographic reach.

The primary in one region handles all writes, introducing 100-300ms latency for non-local users. Up to five secondary regions provide read-only replicas, with each secondary region supporting up to 16 read replicas for excellent read scalability and low-latency reads worldwide.

Data replicates asynchronously with typical lag of one second or less, meaning **secondary regions may serve slightly stale data.** Aurora Global Database can span continents but only for read replicas, *not* writes.

Aurora DSQL allows applications to write to multiple regions simultaneously.

However, DSQL is **fundamentally limited to only two active regions plus one witness region**. The witness region serves as a tie-breaker for consensus voting and participates in durability but doesn't serve application traffic; this provides fault tolerance benefits without performance or capacity benefits.

Also, **DSQL regions are confined to the same continent** because cross-continental latency would make DSQL's synchronous consensus protocol too slow for acceptable transaction commit times. This means **a global SaaS company with customers in North America, Europe, and Asia would need to run three separate DSQL database clusters (one per continent)** and write custom application logic to synchronize data between them. This completely defeats the purpose of a distributed database.

Multi-cloud, hybrid and on-prem deployments

Enterprise companies need flexibility and choice in deploying their data infrastructure, whether cloud-hosted, on-premise, in a hybrid environment, or serverless. There are many reasons: for compliance with operational resilience regulations such as DORA, meeting air-gapped and security requirements, avoiding vendor lock-in, or even simply having the option to add or change cloud providers.

Amazon Aurora Global Database and Aurora DSQL both limit enterprise data architecture options because they can only deploy on AWS infrastructure, period. And both AWS offerings are available in fewer regions for deployment than CockroachDB.

CockroachDB is vendor-agnostic on the other hand, giving enterprises an array of choices beyond single cloud provider deployment. Having options for multi-cloud deployments, hybrid cloud/on-prem, and self-hosted environments flexibility ensures that a company can leverage the best features and pricing from different providers while maintaining control over their cloud *and* data infrastructures. CockroachDB offers:

- **Self-hosted deployment** for teams that require complete control over the database environment, and deploy in their own private data centers (or on their own private cloud)
- **Public cloud deployment** on AWS, GCP, and/or Azure for single and multi-region applications
- **Hybrid deployment** in multi-cloud and hybrid cloud/on-prem, or for applications that need to run in a cloud other than AWS/GCP/Azure
- **Cockroach Cloud** for fully-managed serverless deployments; including bring-your-own-cloud for customers who want to use their own cloud infrastructure (and pricing discounts) but have Cockroach Labs manage the database.

CockroachDB enables the utmost deployment flexibility and eliminates vendor lock-in. That leaves companies free to innovate and leverage new cloud services as they are released, and to build with the application and data infrastructures of their choice.

Summary comparison: Flexible deployment

Multi-region:

Each of these databases handle geographic reach, write distribution, and consistency differently.

- ✔ **CockroachDB** is the only choice that provides unlimited geographic reach with unlimited write distribution while maintaining strong consistency, but this does require a somewhat more sophisticated data placement strategy.
- ✔ **Aurora Global Database** prioritizes geographic flexibility over write latency and strong data consistency, and lacks multi-region/multi-write capability. Aurora DSQL, conversely, trades off limited geography in favor of write distribution.
- ✔ **Both Aurora options** impose architectural constraints (single-writer or two-region limit) that simplify operations but limit both scalability and multi-region deployments.

Infrastructure options:

CockroachDB is platform-agnostic. It eliminates vendor lock-in by supporting deployment across AWS, GCP, Azure, self-hosted private data centers, hybrid cloud/on-prem environments, and bring-your-own-cloud or fully-managed Cockroach Cloud serverless options.

Both Aurora Global Database and Aurora DSQL lock enterprises into AWS infrastructure exclusively and open up cloud exit concerns, especially in regulated industries.

Table: Flexible deployment summary comparison (November 2025)

Capability	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Portability	AWS, GCP, Azure, Oracle OCI, CockroachDB Cloud, self-hosted, and hybrid environments	AWS	AWS
Active write regions	Unlimited; all regions accept writes	One only	Two only
Multi-region writes	Native; all nodes can accept reads and writes even across regions	Write operations must go through the primary region; secondary regions use write forwarding to route writes to the primary	All nodes can accept reads and writes, but limited to single continent
Geographic reach	Global (any continents)	Global (any continents)	Regional (same continent)
Multi-continent writes	✔ Yes	✖ No	✖ No
Data residency	✔ Yes: can tie data to a node to optimize performance or comply with data regulations	✖ No ability to tie data to a location	✖ No ability to tie data to a location
Best for	Single-region or multi-region applications that may need to scale and want the freedom to run in any cloud or self-hosted environment	Read-heavy, write-centralized, single-region applications that will not scale and are tied to AWS	Regional AWS-native apps (OCC-specific, greenfield) where a multi-tenant serverless database is appropriate

PART 5:

Customer Experience

Operational simplicity directly impacts developer productivity, system reliability, and total cost of ownership. Let's compare how CockroachDB, Amazon Aurora Global Database, and Aurora DSQL handle day-to-day database operations: backups, upgrades, and schema management. We'll also examine database migration tooling, Postgres compatibility, and the technical customer support experience for each provider.

Automated backups & recovery

CockroachDB provides incremental, locality-aware backups with point-in-time recovery (PITR) and granular restore options at table, database, or cluster level.

This enables faster, targeted recovery that reduces outage blast radius and data-loss exposure. Key capabilities include:

- continuous incremental backups without performance impact
- locality-aware backups optimizing storage costs and recovery time
- fine-grained restore options
- PITR for reconstructing database state at any point in time
- AS OF SYSTEM TIME queries for dispute investigations that can cut investigation time by more than 50%, with no customer-facing downtime

Aurora Global Database uses automated backups through daily snapshots with typical replication lag of less than one second, though unplanned failover can lead to data loss.

Manual configuration is required for PITR, with asynchronous cross-region replication at ~1 second lag and full database restore required with no granular table-level recovery. Aurora failover requires downtime and may require client reconnection or manual intervention, with failover taking 10-30 seconds and recovery procedures more time-consuming due to full snapshot restoration.

Aurora DSQL supports full snapshot backup/restore via AWS Backup.

DSQL has centralized management, automated schedules, cross-region/account support, and WORM protection, but lacks PITR. Only full cluster backups are supported, with no table or database-level granularity. Without PITR, teams cannot restore to specific points in time between snapshots, potentially increasing data loss exposure and complicating audit and compliance scenarios.

Table: Comparing backups & recovery (November 2025)

Feature	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Backup type	Incremental, locality-aware	Daily snapshots	Full snapshot via AWS Backup
Backup frequency	Continuous	Daily automated + manual	Automated schedules (configurable)
Point-in-Time Recovery (PITR)	✔ Yes, with AS OF SYSTEM TIME queries	⚠ Manual configuration required	✖ No
Restore granularity	Table, database, or cluster level	Full database only	Full cluster only
Recovery Point Objective (RPO)	Near-zero with incremental backups	<1 second (typical replication lag)	Depends on snapshot frequency
Recovery Time Objective (RTO)	Fast, targeted recovery	<1 minute (managed switchover); 10-30 seconds (failover)	Varies (full snapshot restore)
Data loss risk	Minimal	Zero (managed); some risk (unplanned failover)	Between snapshots (no PITR)
Historical data queries	✔ Yes, AS OF SYSTEM TIME	✖ No	✖ No
Operational impact	Faster recovery, minimal data loss, no production impact	Requires failover planning, possible downtime	Limited flexibility, higher data loss exposure

Database upgrades & maintenance operations

CockroachDB can be upgraded with zero downtime as each node is self-contained and operates independently. When a node is upgraded, it is seamlessly replaced by a new node running the latest version, transparent to the application and allowing continued database use without interruption. This allows for:

- zero-downtime rolling upgrades across all nodes
- zero-downtime rolling repaves for security patches or OS updates
- data replicated multiple times ensuring availability during node maintenance
- no application changes required during upgrades
- pause/resume control for maintenance operations

Aurora Global Database's single-write node replication model requires planned downtime to perform upgrades and changes, adding overhead and complexity. Maintenance windows are required for most upgrades, with the single-writer architecture limiting upgrade flexibility and potential failover to standby during major version upgrades. Application reconnection is often needed post-upgrade, requiring the coordination of downtime windows across teams.

Aurora DSQL fully-managed database and maintenance operations occur automatically without user intervention, though long-term operational patterns are still emerging. AWS manages all infrastructure upgrades but upgrade timing is not under user control, which may conflict with business-critical periods.

Table: Comparing upgrades & maintenance (November 2025)

Feature	CockroachDB	Amazon Aurora Global Database	Aurora DSQL
Live, zero-downtime upgrades	✔ Yes, node-by-node rolling upgrades	✖ No, requires maintenance windows	N/A (AWS-managed)
Rolling repaves	✔ Yes, zero downtime	✖ No, requires maintenance windows	N/A (AWS-managed)
Upgrade control	Full control with pause/resume	Limited; requires maintenance windows	✖ No control (AWS-managed timing)
Application changes required	✖ No	⚠ Often requires reconnection	✖ No
Replication model impact	Multi-active (all nodes accept writes)	Single-writer (limits flexibility)	Multi-region active-active (limited)
Upgrade timing control	✔ User-controlled	⚠ Requires maintenance window scheduling	✖ AWS-controlled
Version management	Self-service with instrumentation	Managed but requires planning	Fully AWS-managed
Operational maturity	10+ years in production	Mature (AWS-managed)	Recently GA (May 27, 2025)
Operational impact	Shorter maintenance windows, zero downtime	Planned downtime, coordination overhead	Minimal burden, but no timing control

Migration experience, tooling, and support

Evaluating database platforms requires data architects to look beyond technical specifications. Smooth migration paths accelerate time-to-market, while responsive support is critical for troubleshooting production issues and architecting for scale.

PostgreSQL compatibility

CockroachDB is PostgreSQL compatible and the majority of PostgreSQL syntax, meaning that **existing applications built on PostgreSQL can be migrated to CockroachDB with minimal, if any change** to the application. Postgres compatibility covers the majority of typical use cases with core SQL features including:

- triggers
- stored procedures
- UDFs
- temporary tables
- sequences
- materialized views
- foreign keys with full referential integrity
- partial/inverted indexes
- vector type and indexes
- SERIALIZABLE isolation by default
- LDAP/Active Directory integration

Amazon Aurora Global Database uses the native PostgreSQL engine with AWS enhancements, providing near-complete PostgreSQL compatibility with broad extension support, tools and ORMs working out-of-the-box, minimal application changes required, and standard PostgreSQL tooling and frameworks. Aurora offers the smoothest migration path for applications heavily reliant on PostgreSQL-specific features and extensions.

Aurora DSQL has the most PostgreSQL limitations, lacking foreign keys, triggers, sequences, temporary tables, materialized views, PL/pgSQL, and many extensions (including pgvector and PostGIS). Without foreign key enforcement, applications must implement integrity checks in code, risking data drift, higher testing burden, and longer audits. DSQL migration requires the most application refactoring of the three.

> **The upshot: Amazon Aurora Global Database wins in terms of pure PostgreSQL compatibility.** However, this is mitigated by operational limitations such as AWS-only deployment, planned maintenance downtime, manual sharding at scale, and limited multi-region write capabilities.

Additionally, the fact that Aurora Global Database is a fork of the open-source PostgreSQL code creates technical debt. This is because proprietary changes must be constantly "forward ported," and reconciled with code changes every time a newer version of PostgreSQL is released. This process becomes increasingly complex and time-consuming as the fork diverges further from the main project, effectively creating a maintenance burden that slows development and updates.

In comparison, CockroachDB provides strong compatibility for the majority of common Postgres use cases. Superior operations include zero-downtime, multi-region writes, and horizontal scaling, which are valuable for applications requiring global distribution and operational resilience. Technical debt is not an issue, because CockroachDB "owns the stack": the PostgreSQL compatibility layer native to our implementation, which will never require forward porting.

Customer support experience

CockroachDB received Gartner Customers' Choice recognition in **2023** and **2024**, with 97% customer recommendation rate in the **2024 Gartner report** and strong G2 reviews praising **engineering accessibility.**

CockroachDB support emphasizes direct access to actual database engineers, with deep problem-solving expertise plus proactive guidance for operational database tuning, like performance optimization. Customer support also includes professional services covering migration assessment with MOLT tools, architecture design for multi-region/multi-cloud scenarios, performance tuning, and disaster recovery planning. Tailored training is available for enterprises of all sizes, from two-person startups to large organizations.

Even some AWS Enterprise support customers have said that it can be difficult to get knowledgeable help for complex database issues like distributed scenarios, failover planning, and performance optimization. Numerous online reviews of AWS support reveal customers voicing frustration. Some customers report difficult engineering access, tiered barriers making technical escalation challenging, limited expertise on database infrastructure versus optimization, long resolution times (some AWS users say it can sometimes take weeks for AWS support to close a ticket), and generic responses.

Table: Customer support comparison (November 2025)

Customer support comparison	CockroachDB	Amazon Aurora Global Database/ Aurora DSQL
Engineering access	✔ Direct access	⚠ Requires escalation
Database expertise	✔ Distributed SQL experts	⚠ General AWS expertise
Architecture guidance	✔ Proactive consultation	Self-service or expensive TAM
Multi-region expertise	✔ Core competency	⚠ Limited
Customer satisfaction	**97% recommend** (Gartner)	Mixed reviews on support

Migration tooling and support: CockroachDB vs Amazon Aurora

CockroachDB's migration tooling and Amazon Aurora migration tooling serve two different purposes:

CockroachDB MOLT (Migrate Off Legacy Technology) is a seamless path for migrating PostgreSQL, Oracle, MySQL, and Microsoft SQL Server to CockroachDB.

The free MOLT toolkit along with our experienced professional services organization enables safe, minimal-downtime database migrations to CockroachDB:

- **MOLT Fetch** extracts schema and data from the source database
- **MOLT Convert** converts schemas with CockroachDB-specific optimizations
- **MOLT Verify** validates data consistency post-migration

AWS Database Migration Service (DMS) is a paid service that supports heterogenous migrations to databases in the AWS ecosystem.

AWS DMS is a fully-managed paid service for database migrations to any AWS database. Components consist of a managed replication server to run migration tasks, endpoints for source and target database connections, and AWS's SCT (schema conversion tool) for heterogeneous migrations.

Table: MOLT vs DMS side-by-side comparison (November 2025)

Feature	CockroachDB MOLT	AWS DMS
Cost	Free	~$350+/month
Target	CockroachDB only	Any AWS database
Sources	PostgreSQL, MySQL, Oracle, MS SQL Server, and CockroachDB	15+ databases
Downtime	Minimal downtime	Yes (with CDC)
Migration type	One-time full load migrations with continuous replication	Full load, CDC (continuous replication), or both
Schema conversion	CockroachDB-specific	General-purpose (SCT)
Management	Self-managed (you run the tools)	AWS-managed
Validation	Detailed (row-level)	Basic (table-level)

Migration support services

CockroachDB: CockroachDB's Professional Services team combines tooling and expert guidance, using MOLT together with comprehensive migration engagements:

- **Assessment:** Analyze source database and migration complexity
- **Planning:** Design migration strategy and downtime windows
- **Schema optimization:** Leverage MOLT for CockroachDB-specific tuning
- **Execution:** Use MOLT tools with hands-on guidance
- **Validation:** MOLT Verify ensures data integrity
- **Optimization:** Post-migration performance tuning

Amazon Aurora (Global/DSQL): AWS DMS alone doesn't offer engineering support for migrations, requiring expensive consulting engagements when an organization doesn't have (or is unable to dedicate) experienced technical staff to design and run the migration.

Use both together for complex migrations: Many teams leverage the strengths of both tools in tandem: DMS for initial heterogeneous conversion and CDC, then MOLT for final CockroachDB-specific optimization and cutover.

The upshot: Customer experience

The optimal distributed SQL database choice comes down to organizational priorities. Amazon Aurora Global Database suits organizations requiring absolute PostgreSQL compatibility, if they are willing to make the tradeoff for single-writer architecture, AWS vendor lock-in, and potentially frustrating technical support experiences.

CockroachDB is the stronger choice for organizations prioritizing strong PostgreSQL compatibility for real-world applications, in step with:

- operational resilience
- global distribution
- horizontal scalability
- guaranteed-accessible engineering support

Aurora DSQL is best suited for greenfield applications with simple data models. For long-term success, choose CockroachDB: its distributed architecture, proactive support, and free migration tooling converge to deliver superior total cost of ownership.

CONCLUSION:

CockroachDB: Distributed Without Compromise

This comprehensive analysis explored fundamental architectural differences between CockroachDB, Amazon Aurora Global Database, and Amazon Aurora DSQL. Their design divergences directly impact enterprise operations across four critical database dimensions:

- resilience
- scalability
- deployment flexibility
- customer experience

Our comparative analysis shows that Amazon Aurora Global Database doesn't live up to its name. It's limited by its own single-region write architecture, best serving organizations that choose to prioritize absolute PostgreSQL compatibility. The tradeoffs: eventual consistency, data latency, and the potential risk of data loss in failover scenarios.

We've also seen how Aurora DSQL is indeed a step forward for AWS databases when it comes to supporting modern distributed application architecture. But DSQL's far-from-complete Postgres compatibility and geographical constraints keep it from being a serious contender for truly global enterprise-scale workloads.

Organizations that deploy CockroachDB demand:

- unwavering dependability for critical operations
- native scaling capabilities
- flexible deployment options
- straightforward operation with dedicated engineering support

As the pioneering distributed SQL database, CockroachDB eliminates compromise. Enterprises can have uptime, scalability, and data consistency at the core of their database architecture.

Enable resilient applications.
Scale effortlessly across any environment.

Connect with us to learn more about how you can build your applications to be robust, consistent, and scalable with CockroachDB.

Trusted by enterprises for mission-critical use cases

booksy · BOSE · checkout.com · CITY STORAGE SYSTEMS · COATS · DigitalOcean · FAIRE · Five9 · FORM3 FINANCIAL CLOUD · FORTINET · Hard Rock DIGITAL · LANDMARK GROUP · LaunchDarkly · CoreWeave · riskified · ROUTE · sumup · Shipt

Cockroach Labs