Database Design

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

CHOOSE BY WORKLOAD

- **Cloud SQL** — Compatible SQL
- **Spanner** — Distributed SQL
- **Bigtable** — Key & time series
- **Firestore** — Documents

01 Design database solutions

Relational platform selection

- **Cloud SQL** manages MySQL, PostgreSQL and SQL Server. Choose it for compatible relational applications that need managed backups, patching and regional availability with limited application changes.
- **AlloyDB for PostgreSQL** suits demanding PostgreSQL-compatible transactional and analytical workloads. Its primary handles writes; read pools isolate and scale reads, and the columnar engine accelerates eligible analytics.
- **Spanner** supplies distributed relational transactions, strong consistency, horizontal scaling and JSON support. Use it when global transactional correctness and growth justify schema and application adaptation.
- **Compute Engine** preserves OS access, custom agents and unsupported database features. Assess SQL Server dependencies such as SSAS, SSRS and SSIS separately from the core database engine.

NoSQL and analytical storage

- **Bigtable** is a wide-column database for high-throughput, low-latency key and range access: sensor readings, clickstreams and versioned time series. Model around access patterns rather than relational joins.
- **Firestore** stores flexible documents with indexed queries, transactions and real-time updates. **MongoDB Atlas** is a managed document platform across supported cloud providers; assess multicloud requirements explicitly.
- **BigQuery** is an analytical warehouse for large scans, aggregation and machine learning. Separate warehouse reporting from latency-sensitive operational transactions.
- **Memorystore** provides in-memory caching and key-value access. Keep durable business records in an appropriate system of record and define cache freshness and recovery behavior.

Requirements, locality and tenancy

- Start with **data model, consistency, access patterns, peak throughput, latency, growth, RTO and RPO**. A familiar query language alone does not establish engine compatibility.
- **Regional placement** lowers local write latency and controls residency. Multi-region replication improves geographic resilience but adds coordination latency, storage and network cost.

Database Design

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

CHOOSE BY WORKLOAD

- **Cloud SQL** — Compatible SQL
- **Spanner** — Distributed SQL
- **Bigtable** — Key & time series
- **Firestore** — Documents

- **Tenant isolation** can use shared tables, schemas, databases or instances. Stronger resource boundaries simplify independent placement and permissions while increasing cost and fleet management.
- Apply residency requirements to **primaries, replicas, backups, exports and keys**. A cross-region read copy still contains customer data; reconcile global read access with legal placement requirements.

Spanner keys and transactions

- **Primary-key order** determines row ordering and data distribution. A leading monotonic timestamp or sequence can concentrate writes; spread writes with UUIDs, suitable key prefixes or bit-reversed sequences.
- **BIT_REVERSED_POSITIVE sequences** retain integer identifiers while distributing generated values. Avoid application-generated MAX(id) + 1, which creates contention and races.
- **Interleaved tables** colocate related parent and child rows when their access patterns align. Keep transaction working sets small and avoid creating a hot parent range.
- **Secondary indexes** improve selective reads but add storage and write work. Check index key distribution too; a well-distributed table can still have a hot index.

Bigtable row-key design

- **Rows are ordered lexicographically by key**. A device identifier followed by time supports per-device range scans; a timestamp-first key can funnel concurrent writes into one range.
- **Hash or salt prefixes** distribute hot writes, but readers may need fan-out across prefixes. Choose the prefix granularity using measured device skew and required time-range queries.
- **Column families** group columns with shared retention needs. Cells can hold timestamped versions; configure garbage-collection policies by age, version count or an appropriate combination.
- **SSD storage** suits latency-sensitive random reads. Archive cold history separately when it no longer needs interactive access, and test row-key changes before adding capacity to address hotspots.

Firestore application patterns

- **Mobile and web SDK offline persistence** caches data and synchronizes local changes when connectivity returns. Check SDK support; a server-side Node.js client does not automatically gain mobile offline behavior.
- **Real-time listeners** update clients as documents change. Downstream warehouse updates require an explicit event-driven integration or export pipeline.

Database Design / Secure Deployment

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

CHOOSE BY WORKLOAD

- **Cloud SQL** — Compatible SQL
- **Spanner** — Distributed SQL
- **Bigtable** — Key & time series
- **Firestore** — Documents

- **Bundles** package common query results for distribution through a CDN. They reduce repeated reads for shared content; define how clients refresh stale bundles.
- **Document and index design** affect query cost and contention. Avoid concentrating high write rates on a single document or sequential key range, and enforce appropriate client security rules.

Caching, semantic search and AI

- **Cache-aside** reads the cache first and loads misses from durable storage. Define TTL, invalidation and cache stampede protection; cache availability alone does not ensure business-data durability.
- **Evicted keys** indicate memory pressure or eviction-policy effects. Inspect memory and hit rate, tune working-set size and TTL, and add memory when needed; read replicas do not create more primary key capacity.
- **Embeddings** represent semantic meaning. Embed stored descriptions and queries with the same model, then use vector distance or supported approximate nearest-neighbor search to retrieve candidates.
- **Full-text search** matches lexical content; vector search finds semantic neighbors. A supported Vertex AI model integration can generate responses from retrieved context, with access and output validation.

02 Deploy secure and highly available databases

Cloud SQL availability boundaries

- **Regional HA** places primary and standby resources in different zones of one region. Synchronous protection supports zonal failover; the standby is not a read-serving replica.
- **Read replicas** offload reads through separate endpoints. Asynchronous replication can omit recent writes; retain correctness-sensitive reads on the primary. Evaluate supported replica HA when read-serving availability is independently critical.
- **Cross-region replicas** support regional recovery. Forced promotion can lose changes that have not arrived; measure lag and include application reconnection in the recovery plan.
- **Failover** switches an HA primary to its standby. **Replica promotion** creates an independent writable instance. Neither procedure repairs logical corruption already replicated from the primary.

Secure Deployment

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

PROTECT THE DATABASE

- **Identity** — Authenticate
- **SQL grants** — Authorize
- **HA** — Survive zone failure
- **PITR** — Recover history

Disaster recovery and failback

- **RTO** is the maximum acceptable recovery duration; **RPO** is the maximum acceptable data loss. Include detection, operator decisions, provisioning, DNS and application recovery in measurements.
- **Planned switchover** can wait for synchronization while writes are controlled. Unplanned failover after losing a region cannot assume zero loss from asynchronous replication.
- For eligible configurations, **advanced disaster recovery** offers a designated DR replica and controlled switchover. Confirm engine and edition support, replica compatibility and topology before adopting it.
- **Fence the old primary** to prevent split-brain writes. After promotion, re-establish replication and deliberately plan failback; restarting the former primary does not merge divergent changes.
- **Exercise regional recovery** against the remote replica and run real application transactions. Test network paths, credentials, flags, extensions, jobs and dependencies as well as SQL queries.

Cloud SQL backups and PITR

- **Automated backups** create scheduled recovery points. **On-demand backups** serve deliberate milestones or longer retention needs. Select backup location and retention explicitly for the recovery policy.
- **Point-in-time recovery** combines a base recovery point with transaction logs. Enable it before an incident and retain sufficient history; restore to just before corruption and validate the recovered instance.
- **MySQL binary logs** and **PostgreSQL WAL** support their respective recovery mechanisms. Backup retention, log retention and log storage are distinct settings; preserve the recovery chain during changes.
- **Managed backups** use service-managed storage and a configured location. **Exports** write portable objects into a customer bucket; the export bucket is not the managed-backup location setting.
- **Serverless exports** offload supported export work from production compute. Dumps and CSV files serve portability or extracts; they do not replace continuous log-based recovery.

Secure Deployment

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

PROTECT THE DATABASE

- **Identity** — Authenticate
- **SQL grants** — Authorize
- **HA** — Survive zone failure
- **PITR** — Recover history

Spanner, AlloyDB and cache recovery

- **Spanner PITR** retains historical database versions within a configured retention window. Timestamped reads can inspect earlier data; recover a small affected set carefully or create a recovered database for larger damage.
- **Spanner backups** provide retained recovery points. Restore and test access before switching applications. Dataflow-based Avro export/import is useful for portable data movement and test datasets.
- **AlloyDB continuous backup and recovery** complements scheduled backups with point-in-time recovery. Daily backups alone cannot satisfy an RPO shorter than the backup interval.
- **Redis snapshots and exports** provide recovery data rather than synchronous regional failover. Keep export objects in a suitable independent location, and include import time and cache warming in the regional runbook.

IAM identity and SQL authorization

- **IAM database authentication** uses supported IAM identities and short-lived tokens. Enable the feature and create the mapped database user; SQL GRANT privileges still control access to schemas and tables.
- **Cloud SQL Client** authorizes connector access; **Cloud SQL Instance User** permits supported IAM database login. Neither role alone grants SELECT or write privileges inside the database.
- **AlloyDB IAM authentication** also separates identity from SQL permissions. A user who connects successfully but cannot SELECT usually needs the appropriate database privileges.
- **Dedicated service accounts** separate applications and environments. Use attached identities, Workload Identity Federation for GKE or external workload federation to avoid long-lived service-account keys.

Spanner least-privilege access

- **roles/spanner.databaseReader** permits database reads. **roles/spanner.databaseUser** permits reads, writes and schema updates without granting database deletion.
- **roles/spanner.databaseAdmin** manages databases and their access. **roles/spanner.backupWriter** creates backups; reserve broader backup administration for identities that need it.
- **Database-level grants** reduce scope compared with project-wide access. Use separate reader identities for dashboards and broader operational identities only where justified.
- **Fine-grained database roles and views** can restrict exposed tables, columns or partner datasets. Design view security and underlying privileges explicitly; verify that partners cannot bypass the view.

Secure Deployment

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

PROTECT THE DATABASE

- **Identity** — Authenticate
- **SQL grants** — Authorize
- **HA** — Survive zone failure
- **PITR** — Recover history

Private connectivity and connectors

- **Private IP** requires an actual network route from the client. Use a suitable VPC, Shared VPC or hybrid VPN/Interconnect path; remove public exposure when applications need only private access.
- **Private services access** allocates service networking connectivity. VPC peering is non-transitive: peering an application VPC with a database consumer VPC does not automatically reach the managed-service network.
- **Private Service Connect** provides supported service endpoints in consumer networks. Plan endpoint placement, DNS, permissions and routing for each VPC rather than assuming private IPs are universally reachable.
- **Auth Proxy and language connectors** secure and authorize connections but do not create missing private routes or pool connections. Cloud Run needs Direct VPC egress or a suitable VPC connector for private destinations.
- **Connection pools**, such as PgBouncer for PostgreSQL, control connection reuse and concurrency. Configure read/write routing separately; the Cloud SQL Auth Proxy is not a read-replica load balancer.
- **Connection troubleshooting** checks Cloud SQL Admin API enablement, instance connection name, IAM connect permissions, database login, DNS and network reachability. Separate API authorization errors from database permission failures.

Encryption, perimeters and governance

- **CMEK** gives control over encryption-key location, permissions and lifecycle. Use TLS for data in transit; encryption at rest alone does not protect a database connection.
- **VPC Service Controls** restrict supported service access across a perimeter, including Bigtable exfiltration paths. Combine them with IAM and private API access; valid credentials alone should not bypass the perimeter.
- **Organization resource-location policies** constrain eligible deployments. Separate projects where different workloads need different residency policies, and validate backups, replicas and keys independently.
- **IAM conditions with resource tags** can govern supported access centrally. Tags participate in access policy; labels primarily organize resources and cost reporting.
- **Secret Manager** stores database passwords when native authentication is required. A rotation schedule signals a rotation workflow; implement the database password change, secret version update and client refresh.

Operations

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

TUNE WITH EVIDENCE

- **Observe** — Metrics & logs
- → **Diagnose** — Queries & keys
- → **Adjust** — Schema & capacity
- → **Validate** — Load & recovery

03 Manage and optimize database operations

Choose the right diagnostic signal

- **Cloud Monitoring and System Insights** show CPU, memory, disk, connections and replication lag. Compare incident periods with baseline demand before deciding to resize.
- **Query Insights** identifies expensive query shapes, execution time and database load. Use query tags or sqlcommenter metadata to connect inefficient SQL to the originating application.
- **Cloud Trace** follows application requests across services and database calls. Correlate traces, query data and logs to separate application latency from database execution time.
- **Cloud Logging** holds engine errors and operational logs. **Cloud Audit Logs** track cloud API activity; **pgAudit** records configured PostgreSQL audit activity. Neither replaces query-performance profiling.
- **Alert policies** notify supported channels when actionable thresholds are met. Track trends such as disk growth and replica lag alongside immediate saturation and error alerts.

Queries, locks and replica lag

- **Execution plans** reveal scans, join methods and index usage. Improve selective predicates, indexes and statistics before increasing resources; retain query semantics when rewriting SQL.
- **Deadlocks** occur when transactions form a lock cycle. Inspect engine deadlock logs, lock waits and active queries; shorten transactions, use consistent lock ordering and retry aborted transactions safely.
- **PostgreSQL log_lock_waits** helps expose blocking. More CPU or a longer deadlock timeout does not remove the lock-ordering problem; monitor both query load and contention.
- **Replica lag** can come from heavy writes, long transactions, insufficient replica resources or competing read queries. Tune supported parallel replication and isolate analytics when measurements justify it.

Capacity, storage and cost

- **Right-size with measurements** across CPU, memory, connections, IOPS, throughput, table count and growth. SQL Server edition limits and disk capacity can constrain performance independently of vCPU count.
- **Vertical scaling** changes Cloud SQL compute capacity and can interrupt connections. **Read replicas** scale reads, not the primary write path; separate databases or services only when workload boundaries support it.

Operations

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

TUNE WITH EVIDENCE

Observe Metrics & logs → **Diagnose** Queries & keys → **Adjust** Schema & capacity → **Validate** Load & recovery

- **Storage auto-increase** reduces out-of-space risk but needs growth alerts and budget planning. Automatic storage growth does not automatically increase compute resources.
- **SSD migration** addresses disk-latency constraints, but storage-type changes can require a new instance and data migration. Verify the supported operation rather than assuming it is an editable flag.
- **Recommender** helps identify idle, oversized or storage-constrained instances. Reduce backup retention only within RPO and compliance limits; budget alerts notify but do not optimize resources automatically.

Spanner locality and scaling

- **Leader placement** matters for writes and read-write transactions. Put latency-sensitive writers near the leader region; cross-region consensus still has a network cost.
- **Leader-aware routing** directs supported requests toward the leader. Stale reads can reduce read coordination when acceptable, but do not speed up write commits.
- **SPANNER_SYS query statistics** identify expensive SQL. **Read statistics** analyze read operations, while **Key Visualizer** reveals hot key ranges. Match the diagnostic to the observed symptom.
- **Processing units or nodes** supply compute capacity. Monitor CPU and storage headroom, and use autoscaling bounds sized for expected peaks; scale ahead of predictable bursts and allow redistribution time.
- **Benchmark representative transactions** to size capacity. Published per-node throughput examples depend on row size, indexes, locality and workload mix; avoid treating them as universal guarantees.

Bigtable routing and resilience

- **A cluster occupies one zone**. Add replicated clusters in separate zones or regions for the required failure scope; replication between clusters is asynchronous.
- **Multi-cluster routing** can route around an unavailable cluster. Replicated reads can be eventually consistent; assess consistency and transaction requirements before enabling automatic cross-cluster routing.
- **Single-cluster app profiles** pin workloads to selected clusters. Separate serving and batch analytics profiles to isolate compute contention, and define recovery behavior for pinned clients.
- **Key Visualizer** reveals skew and hot ranges that aggregate CPU can hide. Adding nodes helps distributed load but may not fix a hot row or poorly distributed key prefix.
- **Autoscaling and failover headroom** must cover surviving-cluster demand. Use load tests and measured utilization rather than copying a fixed CPU target across every topology.

Operations / Migration & Integration

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

TUNE WITH EVIDENCE

- **Observe** — Metrics & logs
- → **Diagnose** — Queries & keys
- → **Adjust** — Schema & capacity
- → **Validate** — Load & recovery

AlloyDB read and analytical scaling

- **Read pools** isolate reporting from primary transactions. Scale node count for read concurrency and node size for per-node resources; read pools do not increase primary write capacity.
- **Columnar engine** accelerates eligible analytical scans, joins and aggregations while row storage supports transactions. Identify useful columns and inspect query plans instead of assuming all queries benefit.
- **Scheduled scaling** fits predictable business-hour reporting. Metric-driven adjustments fit variable demand; keep minimum capacity and maximum cost bounds aligned with workload recovery needs.
- **Query Insights and model tuning** help decide whether indexes, query changes, columnar optimization or more compute will give the best improvement.

Maintenance and operational automation

- **Maintenance windows and deny periods** control eligible planned updates. Test updates earlier in non-production, enable notifications and use supported self-service maintenance for urgent patching.
- **HA does not eliminate every maintenance interruption**. Select engine and edition capabilities for uptime needs, and make applications reconnect with bounded retries and backoff.
- **Cloud Scheduler with Cloud Run functions** suits lightweight recurring health reports and exports. Add Pub/Sub when useful, use least-privilege identities and track asynchronous operation completion.
- **gcloud sql operations list** inspects instance operations. **gcloud sql instances failover** tests HA failover; **gcloud sql instances promote-replica** changes a replica into a writable primary.
- **Terraform** standardizes instances, flags, labels and network settings. Separate persistent database lifecycle from ephemeral application infrastructure; use deletion protection and retain recovery copies.
- **Versioned schema migrations**, such as Liquibase changesets, make rollouts reviewable and auditable. Rotate server certificates through the supported overlap workflow and validate clients before old trust material expires.

04 Migrate and integrate database workloads

Discovery and cutover planning

- **Inventory compatibility**: engine/version, extensions, procedures, triggers, users, jobs, data types, table count and external dependencies. A PostgreSQL interface does not make Spanner a drop-in PostgreSQL server.
- **Offline migration** stops writes for export and import. **Continuous migration** combines initial load with CDC to reduce the final outage; it still needs a coordinated cutover.

Migration & Integration

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET



- **Transfer time = data size / effective throughput**. Include compression, protocol overhead, validation and restore time; large databases may require more bandwidth or online replication.
- **Cutover sequence**: stop source writes, drain changes, verify lag and data, promote the destination, update clients and validate transactions. Define a rollback boundary before target writes begin.

Database Migration Service fundamentals

- **Database Migration Service (DMS)** orchestrates supported source-to-target migrations. Choose continuous replication for low downtime, and verify engine, version and feature support for the exact path.
- **Connection profiles and migration jobs** define endpoints and replication workflow. Test network reachability, privileges and source logging before starting the initial load.
- **MySQL continuous migration** depends on supported binary logging, replication privileges and source configuration. Retain logs long enough for the initial copy and interruptions.
- **Initial-load parallelism** trades copy speed against source CPU, I/O and connections. Use a supported replica source when appropriate, and benchmark rather than blindly maximizing parallelism.
- **DMS is migration orchestration**; an engine-native external replication topology is a separate option. Distinguish replication from an external source into Cloud SQL from replication out to an external server.

PostgreSQL replication fidelity

- **Logical replication prerequisites** include supported replication settings, privileges and extensions such as pglogical for applicable DMS paths. Tables need suitable primary keys for replicated updates and deletes.
- **Schema changes are not automatically equivalent to row CDC**. Freeze DDL or use the supported coordinated procedure; pglogical.replicate_ddl_command can apply supported changes consistently through the replication stream.
- **Roles, grants, extensions, jobs and sequences** need explicit validation. A successful data copy does not prove that application login and every object behave correctly.
- **Unlogged tables** do not have normal WAL-backed replication and crash-recovery behavior. Inventory them before cloning or migrating; use an appropriate logical copy or convert intentionally when durable replication is required.
- **Self-managed PostgreSQL** can use an online base backup plus continuous WAL streaming to a standby. Catch up, fence the old writer and promote; recovery configuration depends on PostgreSQL version.

Migration & Integration

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

CONTROLLED MIGRATION

Assess Compatibility → **Seed** Initial load → **Replicate** Capture changes → **Cut over** Validate clients

Oracle conversion and migration

- **DMS conversion workspaces** assist Oracle-to-PostgreSQL schema and code conversion for supported Cloud SQL and AlloyDB paths. Review conversion issues and test procedures, functions, triggers and application behavior.
- **Schema conversion and data replication** are separate workstreams. Convert objects, load historical data, capture ongoing changes and rehearse promotion before the production cutover.
- **Oracle CDC** requires appropriate logging, privileges and supported object types. If one table stops replicating, inspect table-specific identity, types and limitations before changing database-wide settings.
- **AlloyDB Omni** brings the PostgreSQL-compatible engine to customer-managed environments. For on-premises HA, use a supported deployment such as the Kubernetes operator and plan quorum, failover and backups explicitly.

Legacy platforms and self-managed care

- **Bare Metal Solution** preserves demanding Oracle dependencies such as RAC, shared storage and Data Guard with fewer database changes. Customers retain responsibility for OS, database licensing, patching and recovery.
- **Google Cloud VMware Engine** preserves suitable VMware application footprints during a datacenter exit. Validate database support and licensing separately from the choice to preserve virtualization.
- **Shared VPC and hybrid connectivity** connect application projects to legacy databases. Plan routes and firewall rules; Bare Metal Solution internet updates may require a controlled proxy or NAT appliance path.
- **Oracle RMAN and Data Guard** address backup/recovery and replication respectively. Keep a fast recovery tier plus economical long-term object storage; test restore throughput against RTO and log capture against RPO.
- **Compute Engine disk growth** expands the disk first, then the partition when applicable and filesystem. For ext4, resize2fs expands the filesystem; preserve backups and verify the actual mount layout.

Migration & Integration

Google Cloud Professional Cloud Database Engineer

KNOWLEDGE CHEATSHEET

CONTROLLED MIGRATION

Assess Compatibility → **Seed** Initial load → **Replicate** Capture changes → **Cut over** Validate clients

Exports, CDC and analytical integration

- **Cloud SQL exports** create supported SQL, CSV or SQL Server BAK output in Cloud Storage. For SQL Server native backups, gcloud sql export bak starts the export; grant bucket access to the instance service identity.
- **Offloaded export or a temporary copy** reduces production impact during one-time moves. CSV carries table data, not a complete schema or recovery history. Large imports also need connection and memory headroom.
- **Datastream** captures supported source changes for analytical pipelines, including BigQuery destinations. Dataflow can transform or apply change records when the pipeline requires it.
- **BigQuery federation** queries supported external databases without maintaining a full warehouse copy. It still consumes source resources; use replicated analytical data for heavier or more isolated reporting.
- **Spanner change streams and Firestore events** can feed downstream analytics through explicit pipelines. Design retries, idempotency, ordering and reconciliation; change events alone do not guarantee end-to-end exactly-once effects.
- **Cross-platform NoSQL moves** need an initial backfill plus a reliable change stream. Validate connector support, row-key mapping, versions, deletes and cutover consistency before moving traffic.