■ Database System Design Cheat Sheet

1. Types of Databases

We'll discuss the different types of databases, their advantages, and disadvantages.

■ *Relational (RDBMS)*

Examples: MySQL, PostgreSQL, Oracle Pros: ACID compliance, mature ecosystem Cons: Hard to scale, rigid schema Use Cases: Banking, E-commerce

■ *Key-Value*

Examples: DynamoDB, Redis Pros: Very fast, scalable Cons: Limited query capabilities Use Cases: Cache, sessions

■ *Document*

Examples: MongoDB, Couchbase Pros: Flexible schema, JSON friendly Cons: Weaker transactions, costly at scale Use Cases: CMS, user profiles

■ *Column-Family*

Examples: Cassandra, HBase Pros: Handles massive datasets, high write throughput Cons: Complex schema design Use Cases: IoT, time-series

■ *Graph*

Examples: Neo4j, Neptune Pros: Great for relationship-heavy queries Cons: Hard to scale horizontally Use Cases: Social networks, fraud detection

■ *Time-Series*

Examples: InfluxDB, TimescaleDB Pros: Optimized for timestamp data Cons: Niche use case Use Cases: Monitoring, telemetry

■ *Search*

Examples: Elasticsearch, OpenSearch Pros: Full-text search, analytics Cons: Eventual consistency, expensive at scale Use Cases: Logs, search

■ *NewSQL*

Examples: Spanner, CockroachDB Pros: SQL + horizontal scalability Cons: Expensive, complex infra Use Cases: Global-scale apps

2. Data Replication

Replication stores copies of data across nodes for fault tolerance and performance.

■ *Single-Master*

Simple, strong consistency Cons: Write bottleneck, replication lag

■ *Multi-Master*

High availability, multiple write nodes Cons: Conflict resolution is hard

■ *Peer-to-Peer*

Fully distributed, no SPOF Cons: Complex conflict handling, network overhead

■ *Synchronous*

Strong consistency Cons: Higher latency

■ *Asynchronous*

Faster writes Cons: Possible data loss on failover

3. Data Partitioning

Partitioning splits large datasets into smaller parts (shards).

■ *Horizontal (Sharding)*

Scales out, reduces load Cons: Hard joins, rebalancing complexity

■ *Vertical*

Separates frequently vs rarely accessed data Cons: Cross-partition queries

■ *Range-Based*

Easy to query Cons: Hot spots possible

■ *Hash-Based*

Even distribution Cons: Hard for range queries, rehashing issues

■ *Directory-Based*

Flexible, easy rebalancing Cons: Central directory = SPOF

4. Cost-Benefit Analysis (Databases)

Evaluating database options based on costs and benefits.

Type	Benefits	Costs	Best Use Cases
RDBMS	ACID, complex queries	Scaling writes hard, rigid schema	Banking, e-commerce
Key-Value	Fast, scalable	Limited queries	Cache, sessions
Document	Flexible schema, JSON	Weaker transactions	CMS, profiles
Column-Family	Big data, high writes	Complex modeling	IoT, logs
Graph	Relationship queries	Hard to scale	Social, fraud
NewSQL	SQL + scale	Expensive, complex infra	Global apps