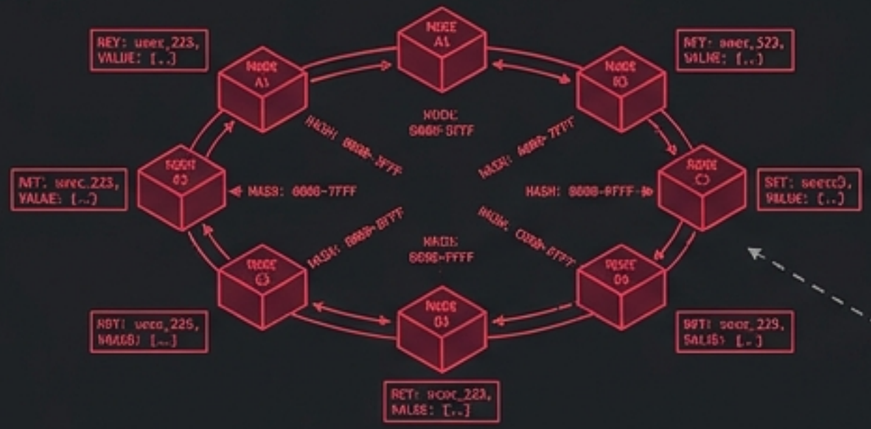
Modern Data Stores

S05E03: Key-Value, Document, Wide-Column, & Graph Internals and Query-First Data Modelling

Audience: Software Engineers and Technical Leads
Prerequisites: Familiarity with OLTP B-trees and SQL fundamentals



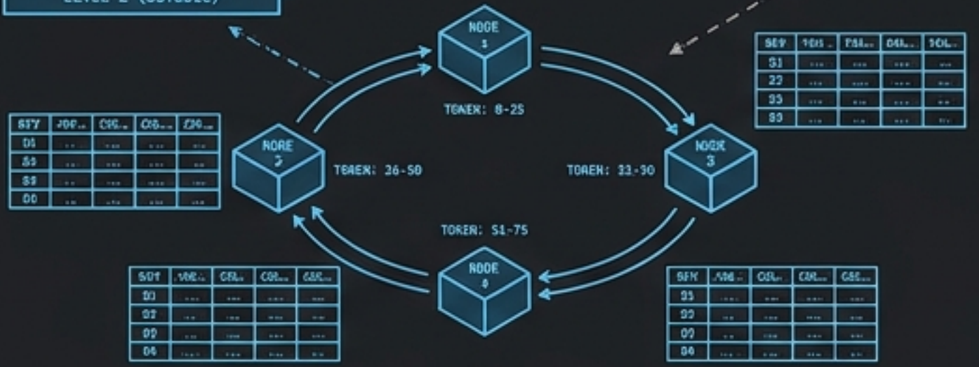
// Key-Value Store:
Distributed Hash Table,
Consistent Hashing

// Document Store:
JSON/BSON Hierarchy,
Schema Flexibility

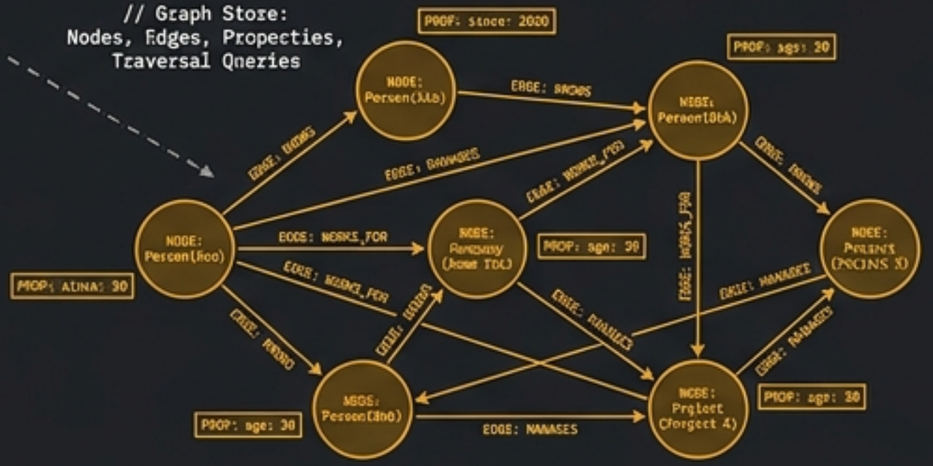




// Wide-Column Store:
LSM Tree, Token Ring,
Columnar Storage



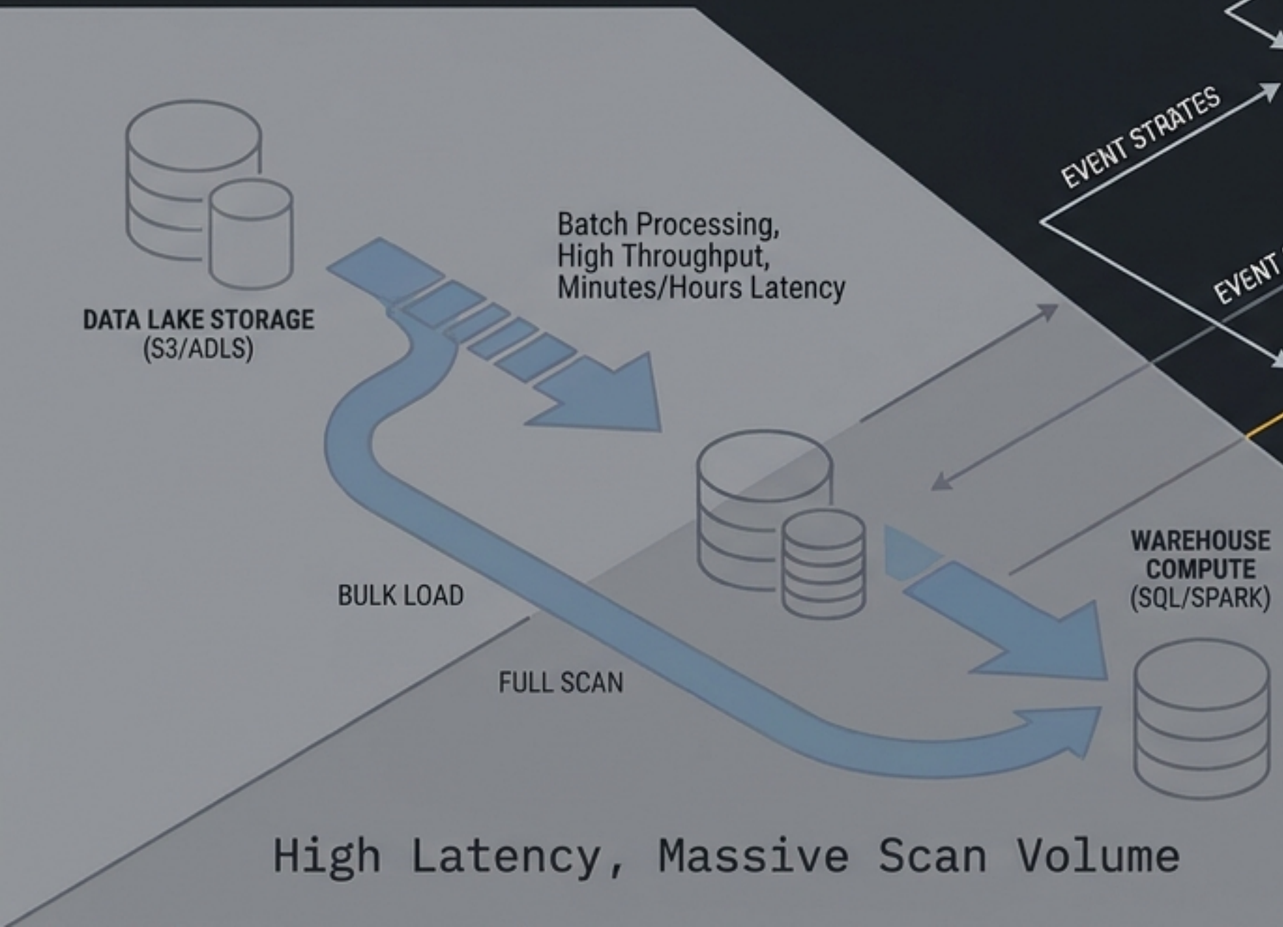
// Graph Store:
Nodes, Edges, Properties,
Traversal Queries

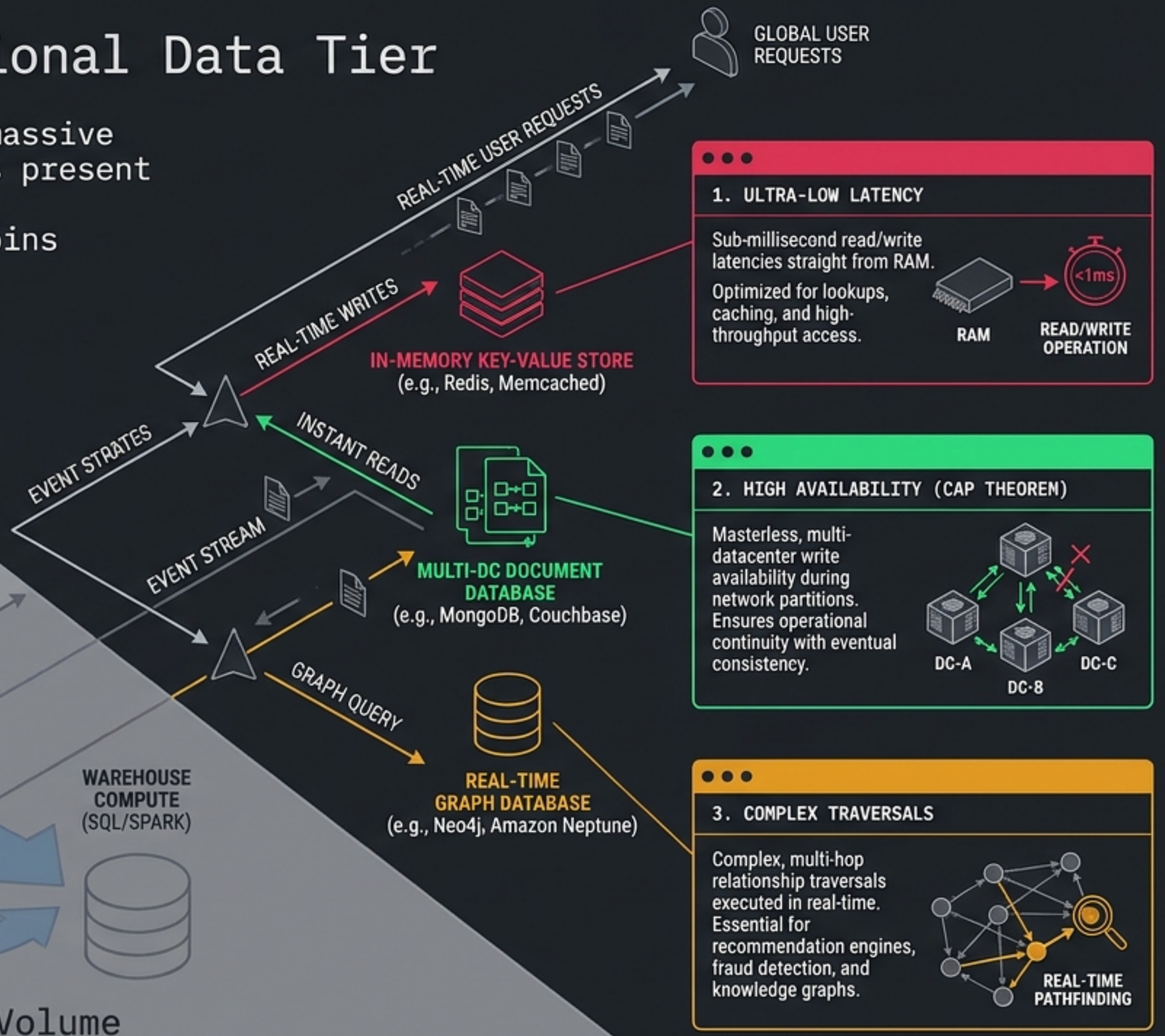


Gemini Notebook

Returning to the Operational Data Tier

While analytical lakehouses conquered massive batch processing, operational workloads present a different physics problem. You must surrender general-purpose relational joins when your system explicitly requires:





Gemini Notebook

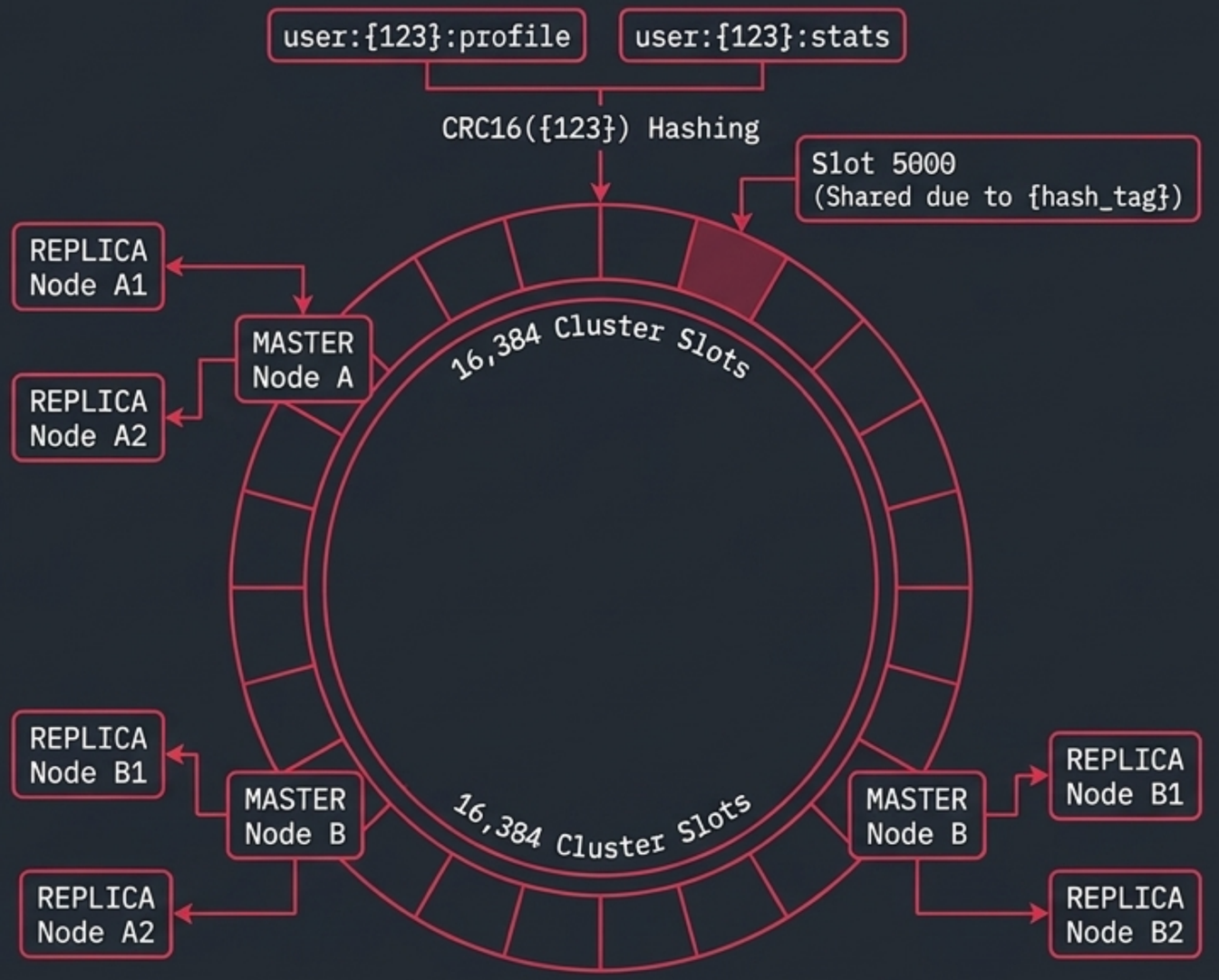
Redis Internals: RAM-Speed Key-Value & 16,384 Cluster Slots

Engine Core:
In-memory structures built for absolute microsecond speed.

Routing:
16,384 fixed cluster slots. Cross-node transactions are invalid unless keys share a {hash_tag} to force single-slot locality.

Durability Trade-off:
Speed is prioritized over strict durability. Persistence relies on asynchronous fsync via point-in-time RDB snapshots or append-only AOF logs.

Memory Pressure:
Eviction relies on LRU or LFU heuristics when RAM limits are breached.



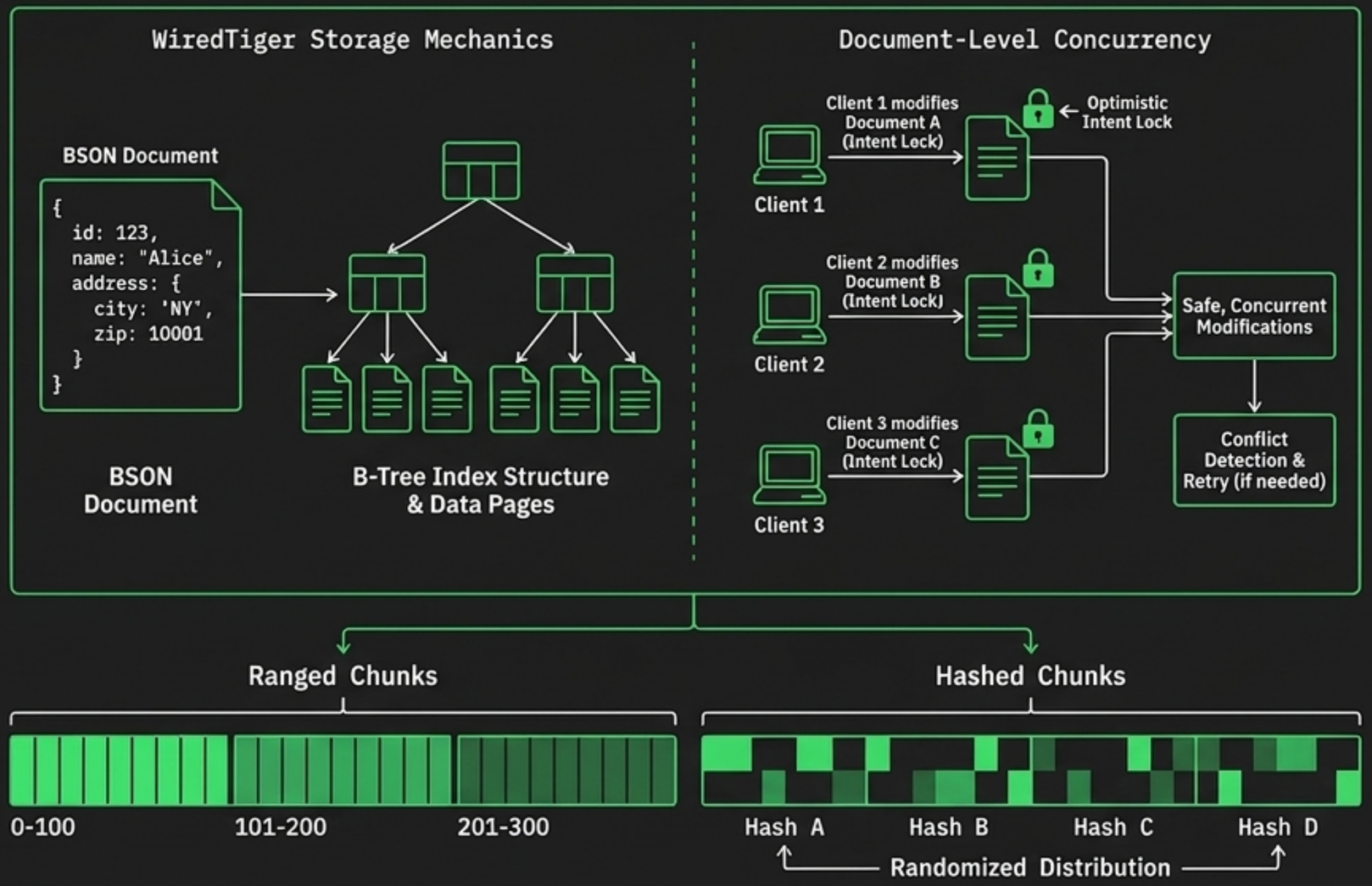
Gemini Notebook

MongoDB Internals: WiredTiger B-Trees & Document Concurrency

Storage Mechanics: Stores embedded BSON documents powered by the WiredTiger engine.
Uses optimistic concurrency control; conflicts transparently trigger write retries, avoiding heavy global locks.

Scale-Out Strategy: Scales horizontally via Shard Keys causing chunk migrations across instances.

The Trade-off: Sacrifices normalized relational joins in favor of deep, physical data locality within hierarchical documents.



Gemini Notebook

Wide-Column Routing: Masterless Token Rings & Tunable Quorums

The Lineage: Built on Amazon Dynamo’s consistent hashing and Google Bigtable’s column-family data model.

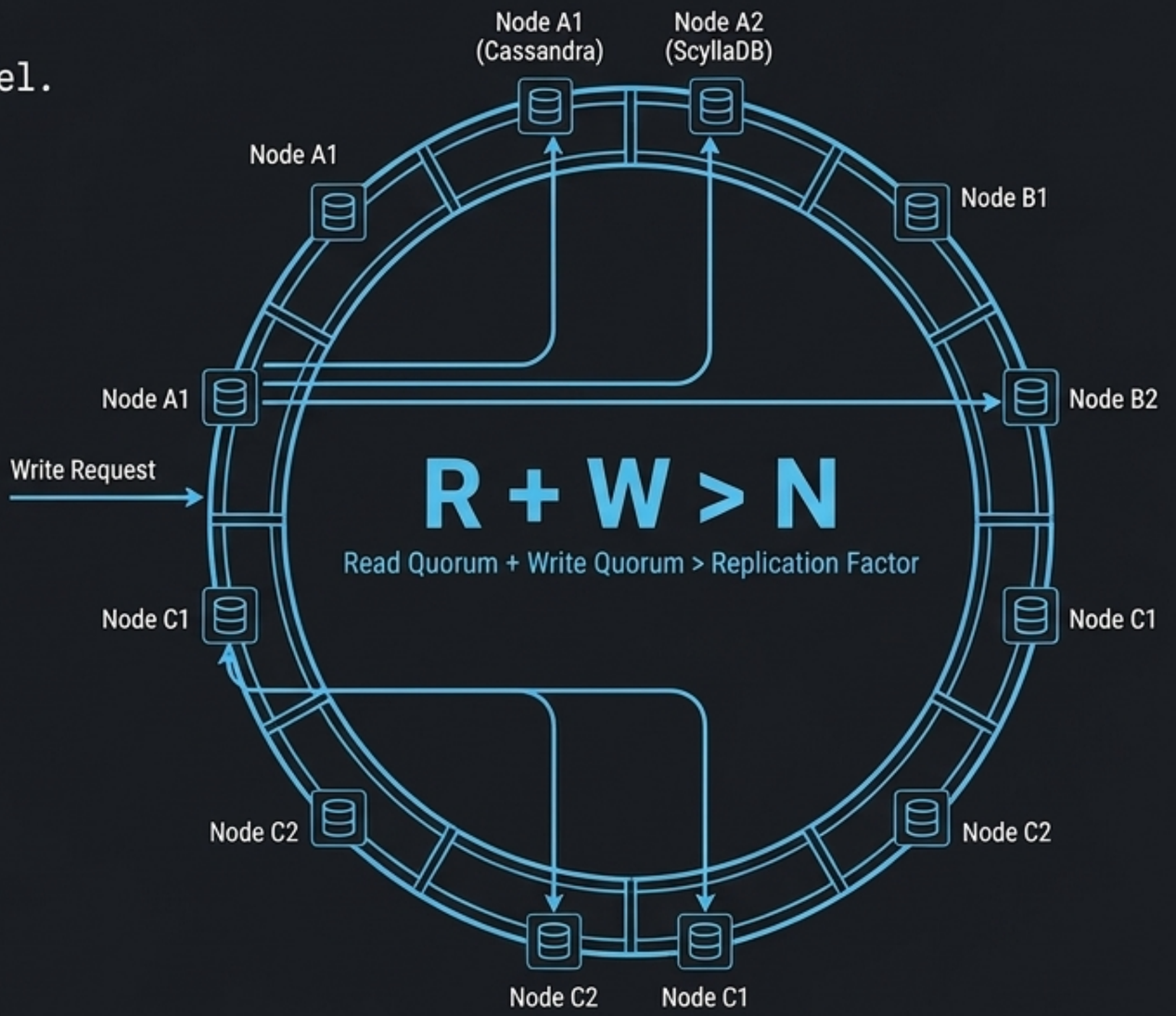
The Lineage: Built on Amazon Dynamo’s consistent hashing and Google Bigtable's column-family data model.
code-class => token-Bigtable, column(a))

Masterless Write Scale: No single point of failure. Virtual nodes (vnodes) uniformly distribute the hash space across commodity hardware.

Masterless Write Scale: No single point of failure. Virtual nodes (vnodes) uniformly distribute the hash space across commodity hardware.

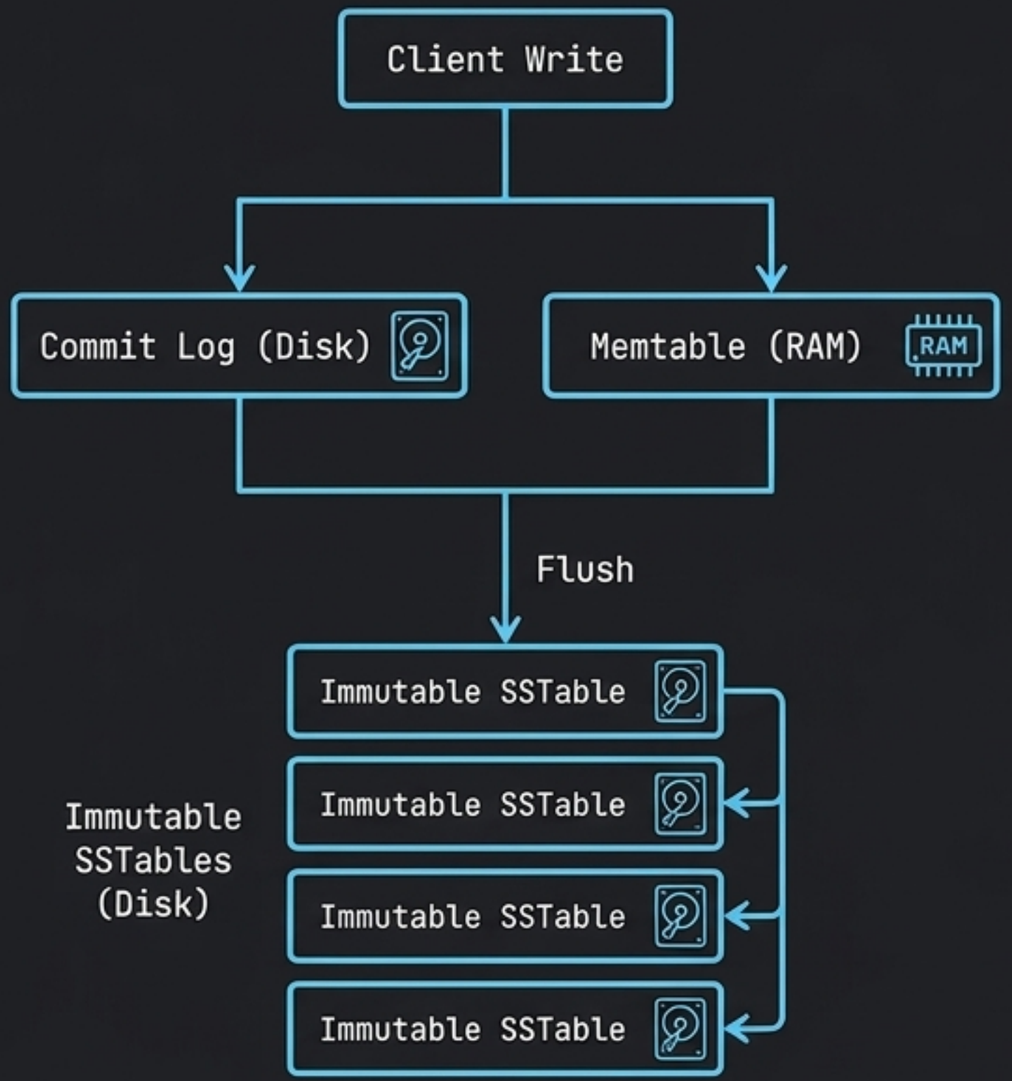
Eventual to Strict: Replicas accept mutations independently. Operators utilize tunable consistency to mathematically guarantee reading the latest write, balancing strict ACID-like behavior against raw availability.

Eventual to Strict: Replicas accept mutations independently. Operators utilize tunable consistency to mathematically guarantee reading the latest write, balancing strict ACID-like behavior against raw availability.



Gemini Notebook

Wide-Column Storage: The Log-Structured Merge Tree Write Path

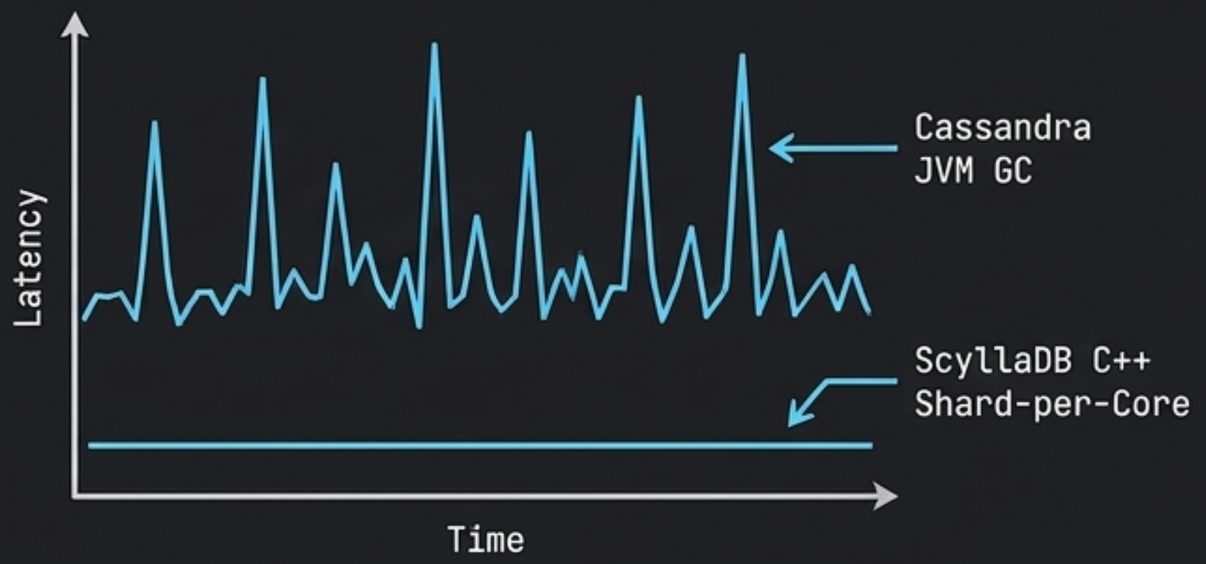


Optimized for Heavy I/O: Data is exclusively appended, never overwritten. Updates and deletes are newer timestamped appends.

The Read Path: Utilizes Bloom Filters to rapidly skip unnecessary disk I/O when hunting for keys across multiple SSTables.

Compaction: Background processes merge fragmented SSTables.

ScyllaDB vs Cassandra: ScyllaDB rewrote the architecture in C++, binding specific CPU cores to specific data shards, eliminating JVM GC pauses.



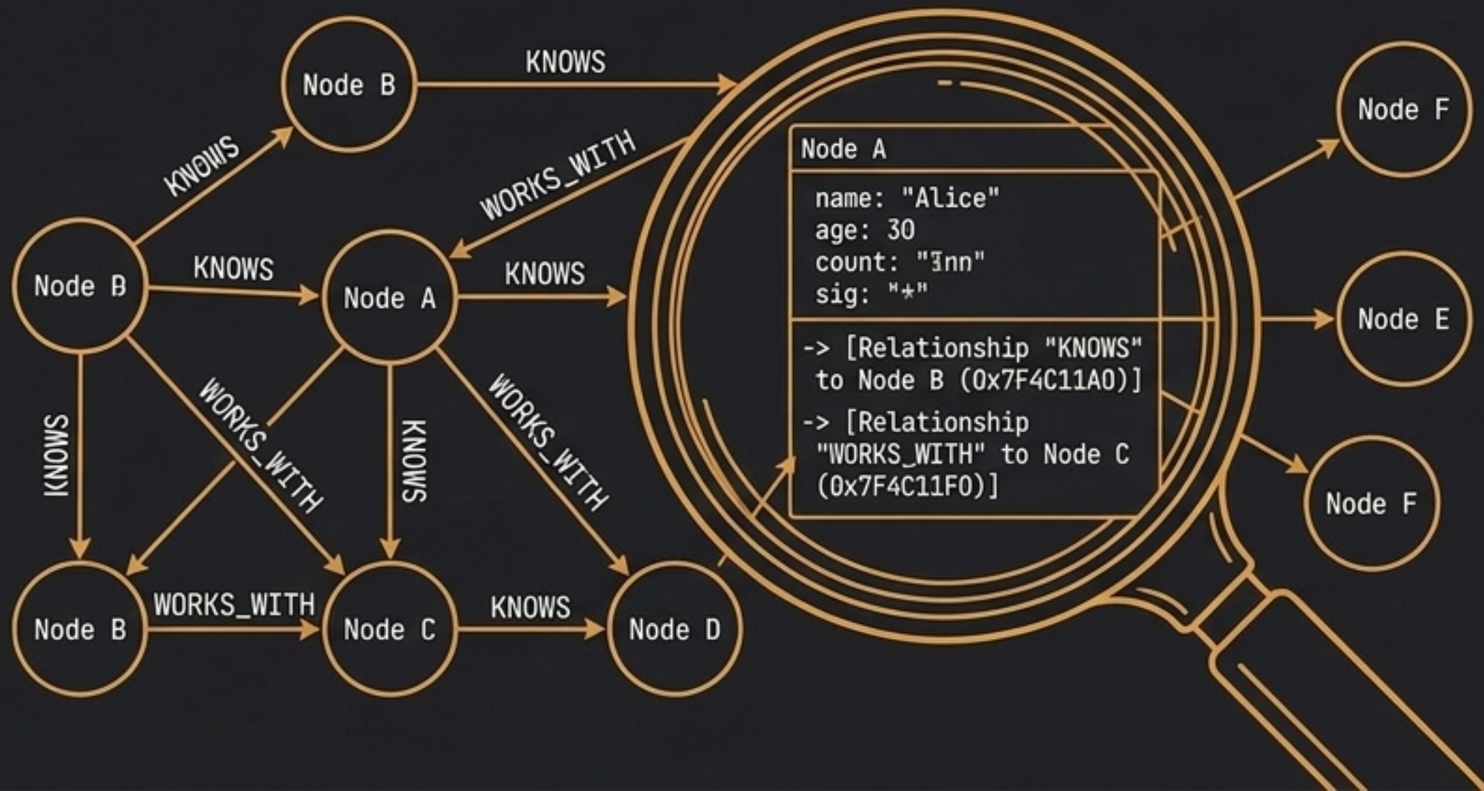
Gemini Notebook

Graph Internals: Index-Free Adjacency in Neo4j

The Relational Join Killer: Property graphs treat relationships as first-class physical entities rather than compute-time matching operations.

Index-Free Adjacency: Nodes contain direct memory pointers to their connected relationships. Bypasses global index lookups entirely.

The Math: Multi-hop traversals execute in predictable O(1) time per hop via native pointer chasing. Fundamentally avoids the exponential performance decay of SQL recursive CTEs.





Gemini Notebook

The Database Trade-Off Matrix

Criteria	PostgreSQL (jsonb)	Redis	DynamoDB / Cassandra	MongoDB	Neo4j
Primary Access Pattern	Evolving / Ad-Hoc	Exact Key	Partitioned Time-Series	Hierarchical Documents	Deep Traversals
Ad-Hoc Query Support	Maximum	Minimal	Non-existent	High	High (Graph specific)
Consistency	Strict ACID	Async	Tunable Quorums	Raft Primary	Causal Clustering
Tail-Latency Cliffs	Lock Contention	RAM Exhaustion	Hot Partitions / Tombstones	Scatter-Gather Sharding	Celebrity Supernodes

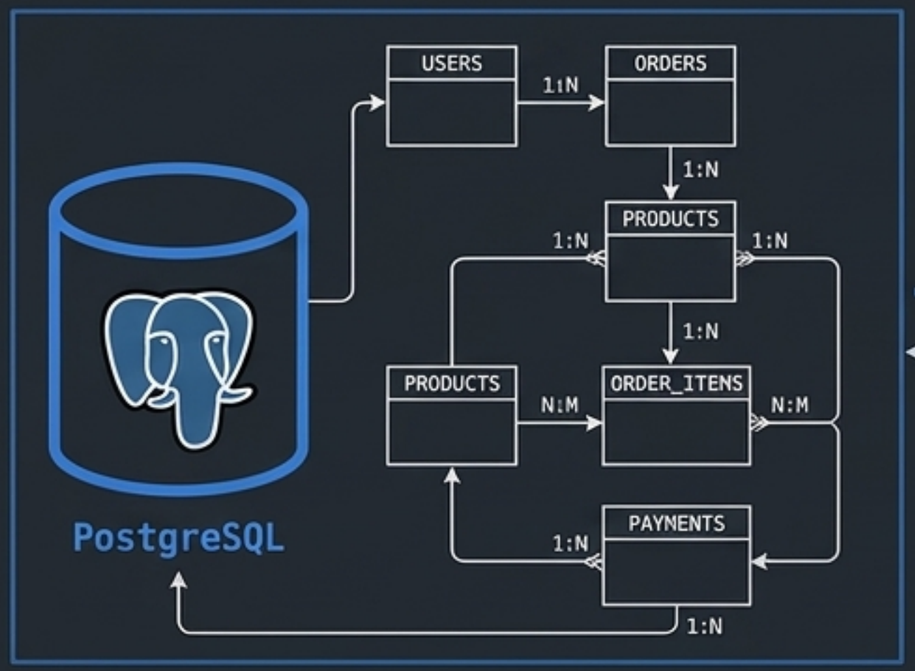
Gemini Notebook

The "Schemaless" Myth & Pre-Computed Joins

The Myth: NoSQL does not mean "no schema". Dropping RDBMS normalization does not free you from data modeling.
The Reality: "Schemaless" actually means your schema is implicitly hidden in your application code.

The Rule: You must design your NoSQL schema backward from your exact read/write access patterns. Drop 3NF joins to pre-compute those joins at write-time, forcing physical data locality on disk.

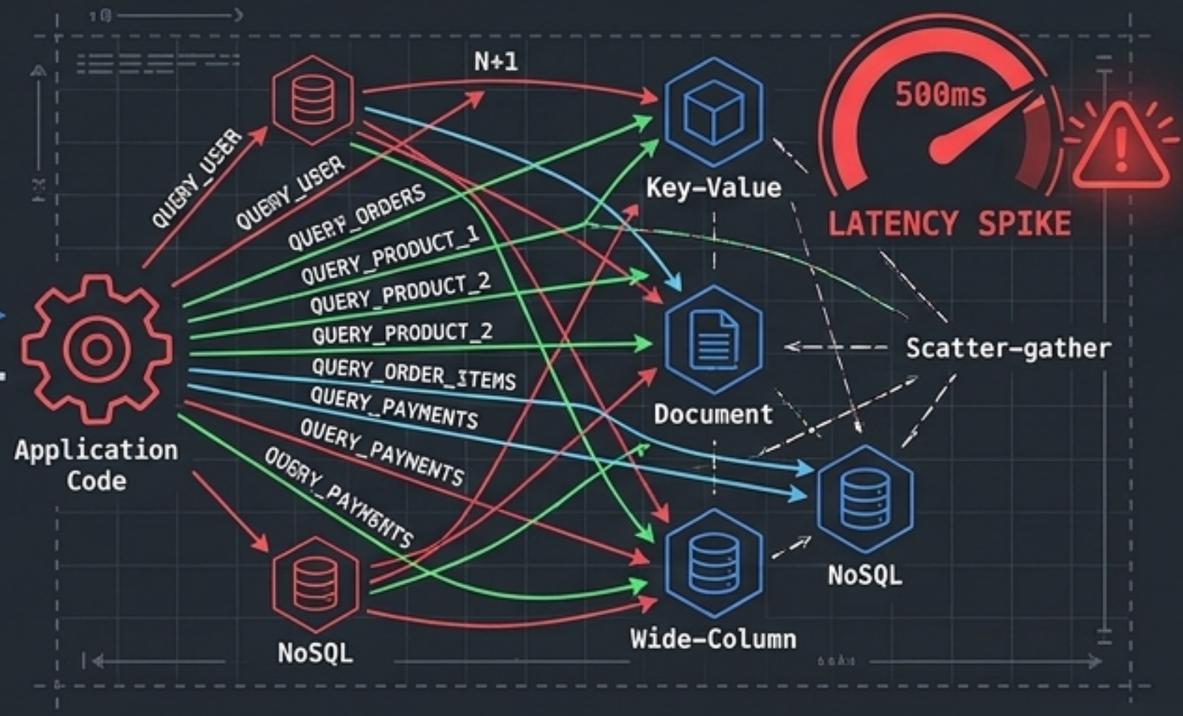
Before: Normalized 3NF



Optimized Query

Structured Result

After: App-Side Join Nightmare (>500ms)



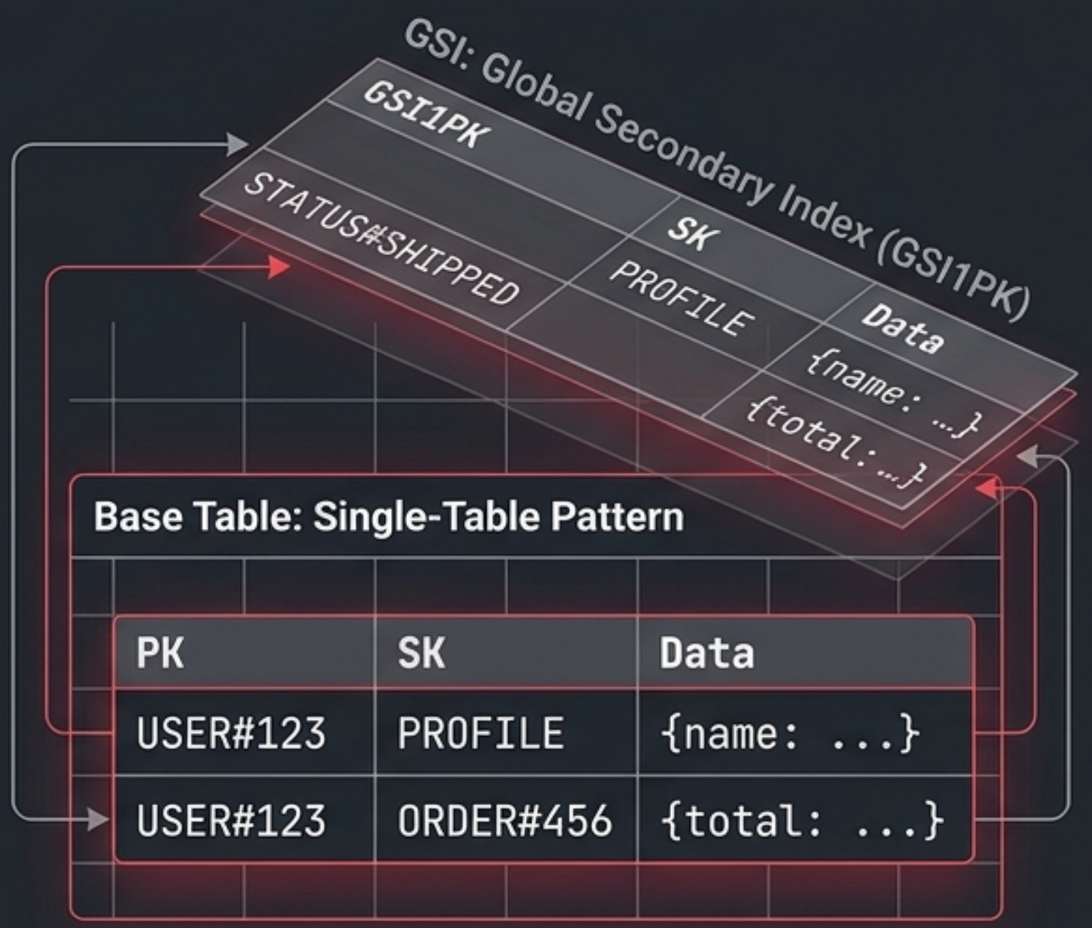
Gemini Notebook

Key-Value Modeling: DynamoDB Single-Table Design

Item Collections: Group disparate entity types (Users, Orders) into a single physical partition by sharing a Partition Key. Retrieves a complete entity graph in one request.

Overloaded Keys: PKs and SKs have no semantic meaning to the database. Define hierarchy via string concatenation.

Global Secondary Indexes: Because base tables only support one access pattern, use GSIs to asynchronously pivot data.



Gemini Notebook

Wide-Column Modeling: Cassandra Partitioning & Ordering

```
PRIMARY KEY ((tenant_id, sensor_id), timestamp)
```

Partition Key: Dictates physical node placement.

Clustering Column: Dictates on-disk sorting within the node.





Physical Segregation: The Partition Key hashes data across the cluster to distribute write load.

Time-Series Slicing: The Clustering Column sorts data sequentially on disk, enabling lightning-fast range queries.

One-Table-Per-Query: You cannot filter by arbitrary columns. Data must be written exactly how it will be queried.

Gemini Notebook

Document Modeling: MongoDB Embedding vs. Referencing

The Bounded Rule:
Embed data that is accessed together, modified together, and has a predictable, bounded growth rate.

The Unbounded Reality:
Reference data that is unbounded, grows continuously, or is updated independently of the parent entity.

The Hard Limit:
Required to avoid hitting MongoDB's strict 16 MB BSON limit and latency penalties of massive arrays.

Embedded Pattern

```
{
  User Profile: {
   "_id": "user_123",
   "name": "Alice Doe",
   "email": "alice@example.com",
   "addresses": [
    {
     "street": "123 Main St",
     "city": "Tech City",
     "zip": "10001"
    },
    {
     "street": "456 Oak Ave",
     "city": "Dev Town",
     "zip": "20002"
    }
   ]
  }
}
```

Referenced Pattern

```
{
  User Profile: {
   "_id": "user_123",
   "name": "Alice Doe",
   "email": "alice@example.com"
}
```

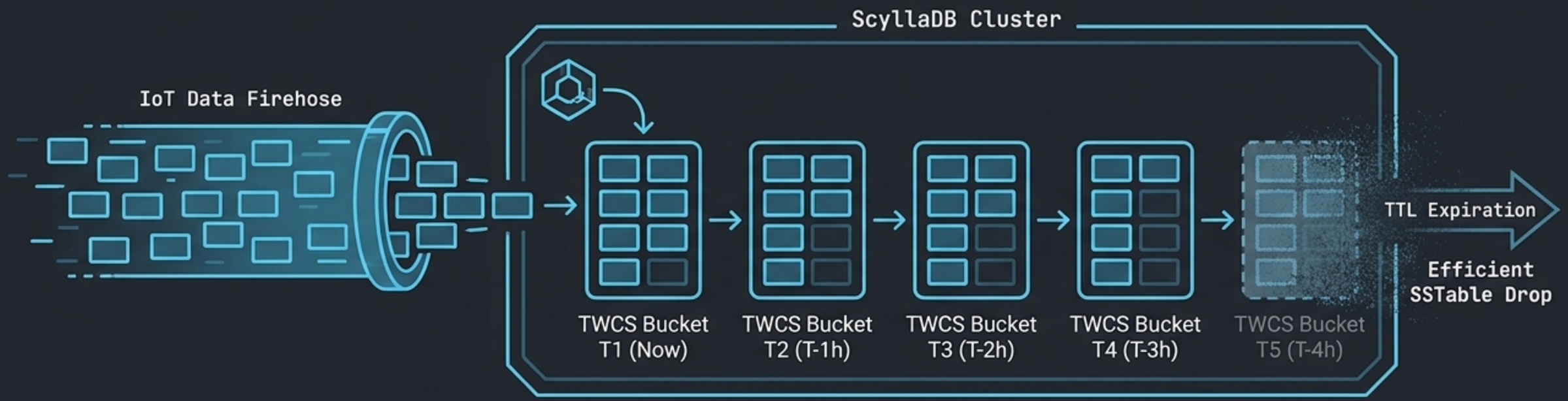
```
{
  Social Post: {
  "_id": "post_987",
  "user_id": "user_123",
  "content": "Just deployed my app!",
  "timestamp": "2024-01-01T12:00:00Z"
}
```

Gemini Notebook

Sizing Scenario: 200,000 Writes/sec IoT Ingest

The Setup: An IoT ingest pipeline requiring 200k continuous writes per second, queried by recent time windows.

The Solution: Wide-column LSMs (ScyllaDB/Cassandra) dominate this profile.



Compaction Strategy: Utilize Time-Window Compaction Strategy (TWCS). Buckets data by time, allowing old SSTables to drop efficiently via Time-To-Live expiration.

The Threshold: Ensure partition sizes remain strictly bounded to <100 MB to prevent node instability.

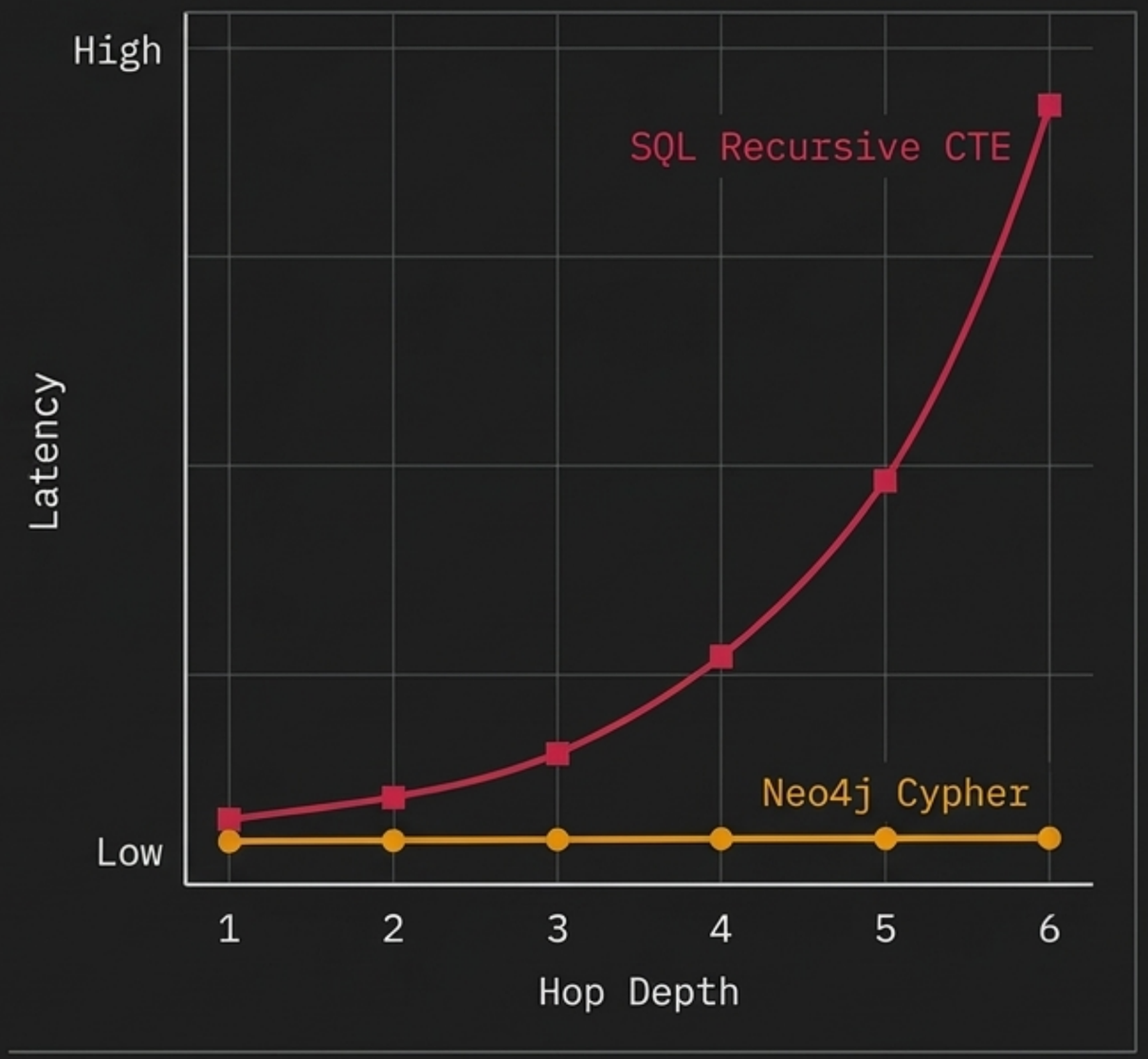
Gemini Notebook

Sizing Scenario: 6-Hop Fraud-Ring Traversal

The Setup: Financial fraud detection mapping indirect connections between flagged accounts.

The Failure: Relational databases require exponential join-table conversions. A 6-hop query in Postgres computes massive cartesian products, spiking latency.

The Solution: Neo4j. O(1) edge pointer chasing makes 6 hops computationally trivial, touching only records in the path.



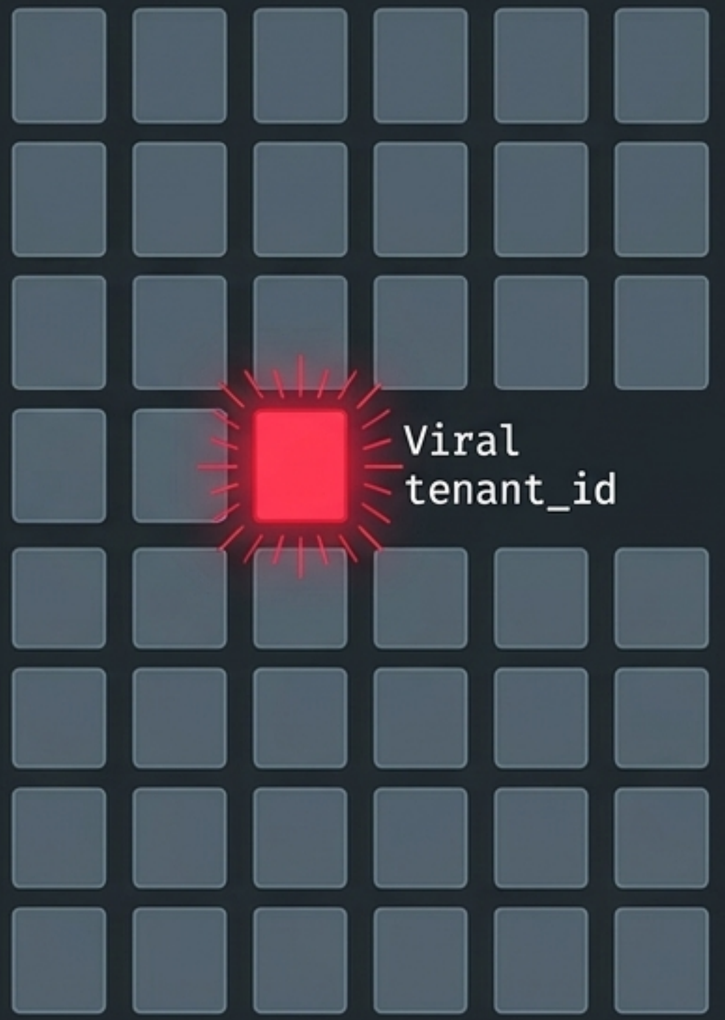
Gemini Notebook

Failure Cliff 1: Hot Partitions in DynamoDB

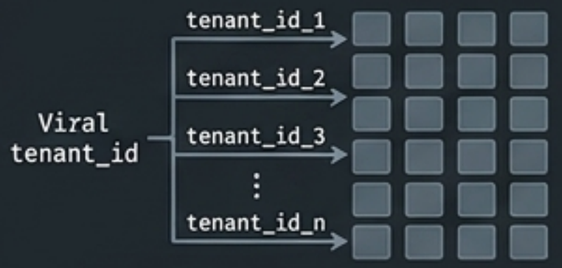
The Trap: Writing too fast to a single partition key. Hard physical limits of 1,000 WCU / 3,000 RCU per partition.

The Symptom: If one viral item spikes, your application receives severe throttling exceptions even if total provisioned capacity is high.

The Fix: Implement write-sharding. Append random or calculated suffixes (e.g., tenant_id_1) to distribute the hot load uniformly.





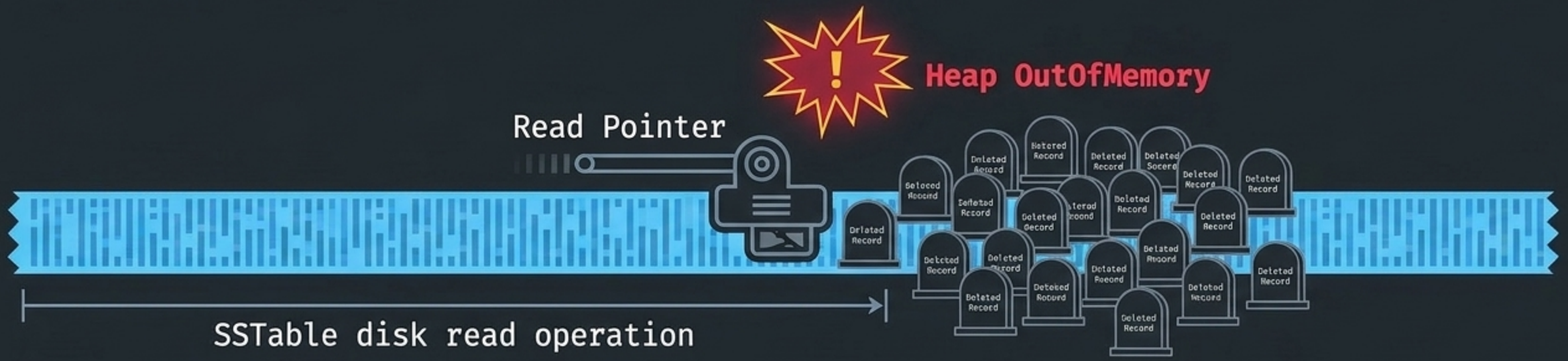


Gemini Notebook

Failure Cliff 2: Cassandra Tombstones & LWTs

The Delete Tax: Wide-column stores don't delete data immediately; they append deletion markers (tombstones). Scanning over accumulated tombstones forces the database to read dead data into memory, stalling reads.

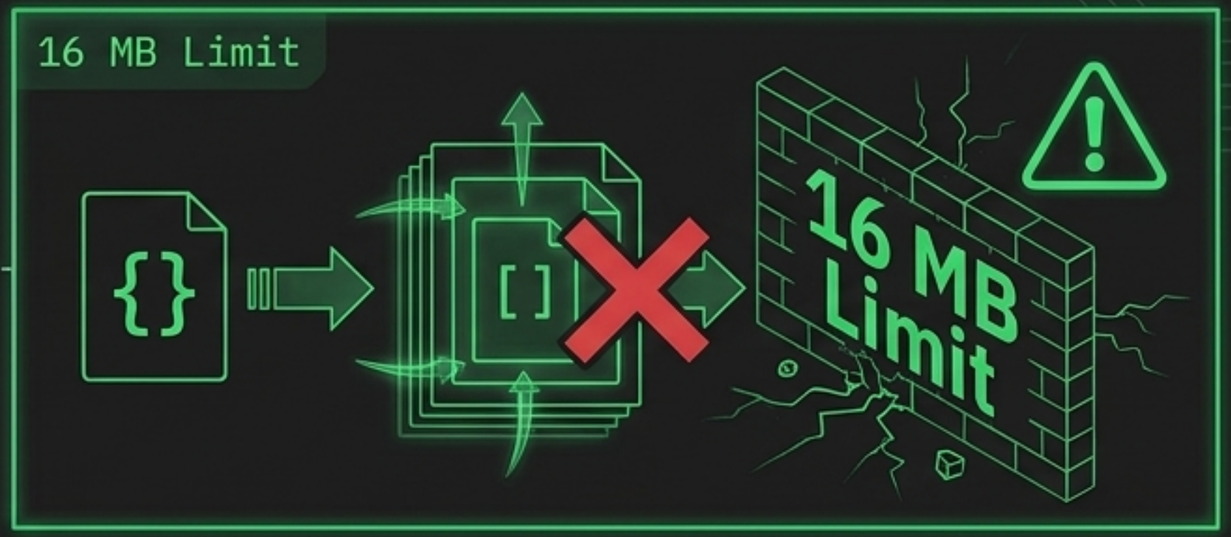
The ACID Illusion: Relying on Paxos Lightweight Transactions (LWTs) for general consistency requires 4 network round-trips. Overusing them destroys throughput advantages.



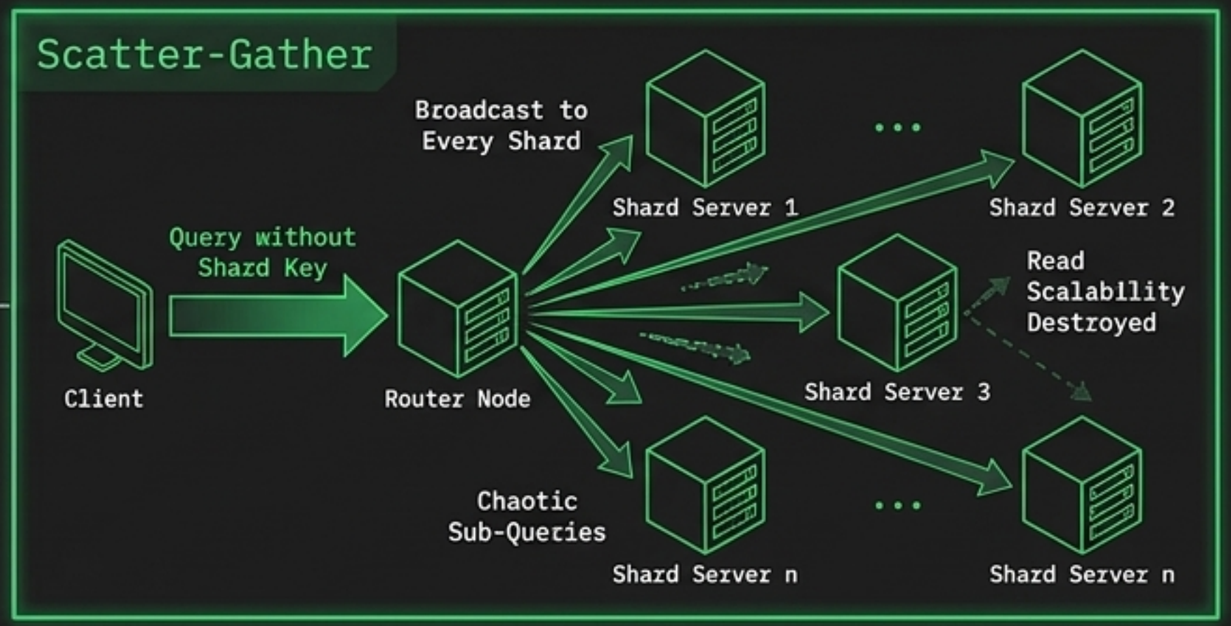
Gemini Notebook

Failure Cliff 3: MongoDB Unbounded Arrays & Scatter-Gather

The Document Trap: Continually appending to arrays inside a user document guarantees hitting the hard 16 MB BSON size limit, breaking the application.



The Query Trap: Queries without the Shard Key force the router to broadcast to every shard in the cluster, destroying read scalability.



Gemini Notebook

Failure Cliff 4: Graph Supernodes & Implicit Schemas

The Supernode Fan-Out



Node	
id	name
	username
	usersname
	program
	[illegible]
	post
	[illegible]
	fain
	[illegible]
	content
	[illegible]
	succeed
	...

Relationships	
id	name
	Celebrity [illegible]
	[illegible]
	...

Traversing nodes with millions of relationships (e.g., a massive NAT gateway IP) exponentially explodes the traversal path, stalling the graph engine.

The Result: Implicit Schemas

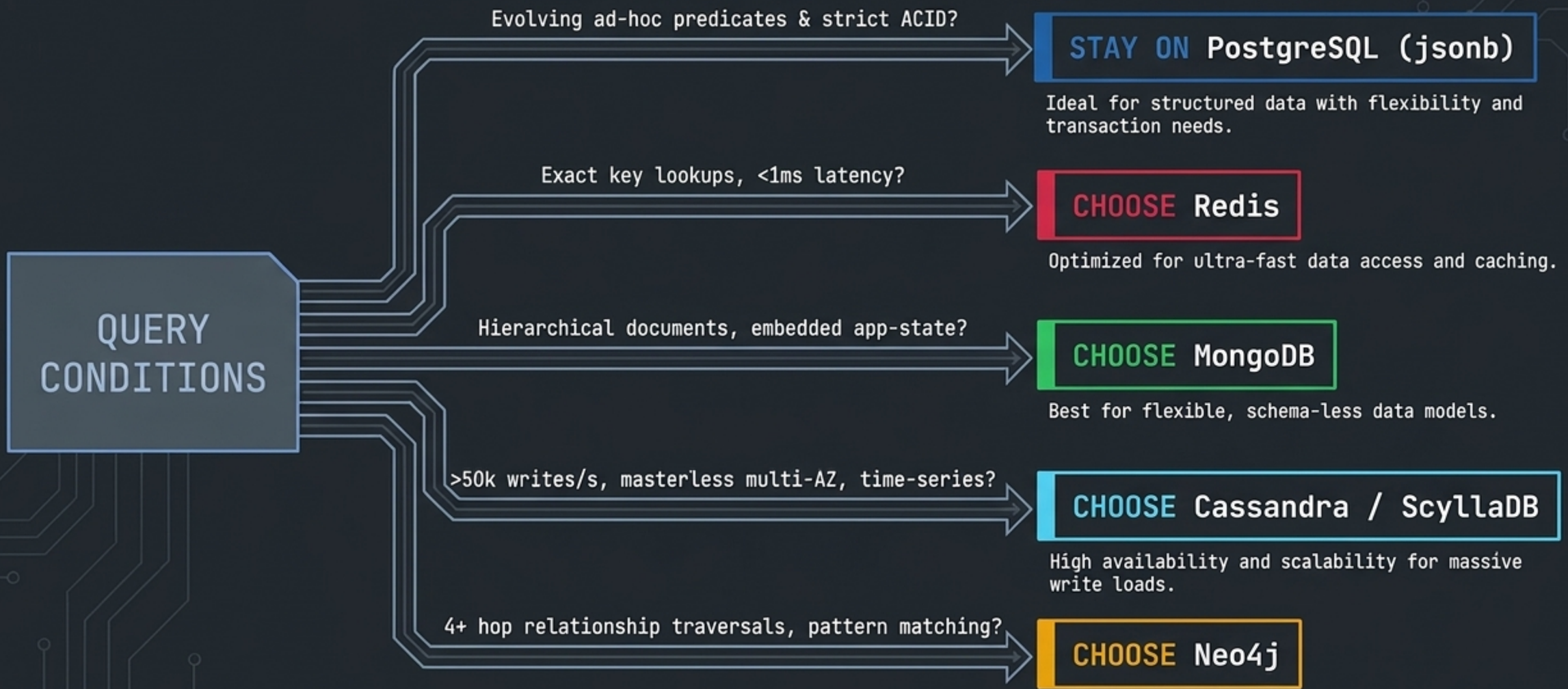
Skipping schema design means future engineers must reverse-engineer data structures from app code.

Editorial:

> “Schemaless just means your schema is hidden in your application code.”
>
> — Martin Fowler

Gemini Notebook

The NoSQL Selection Flowchart

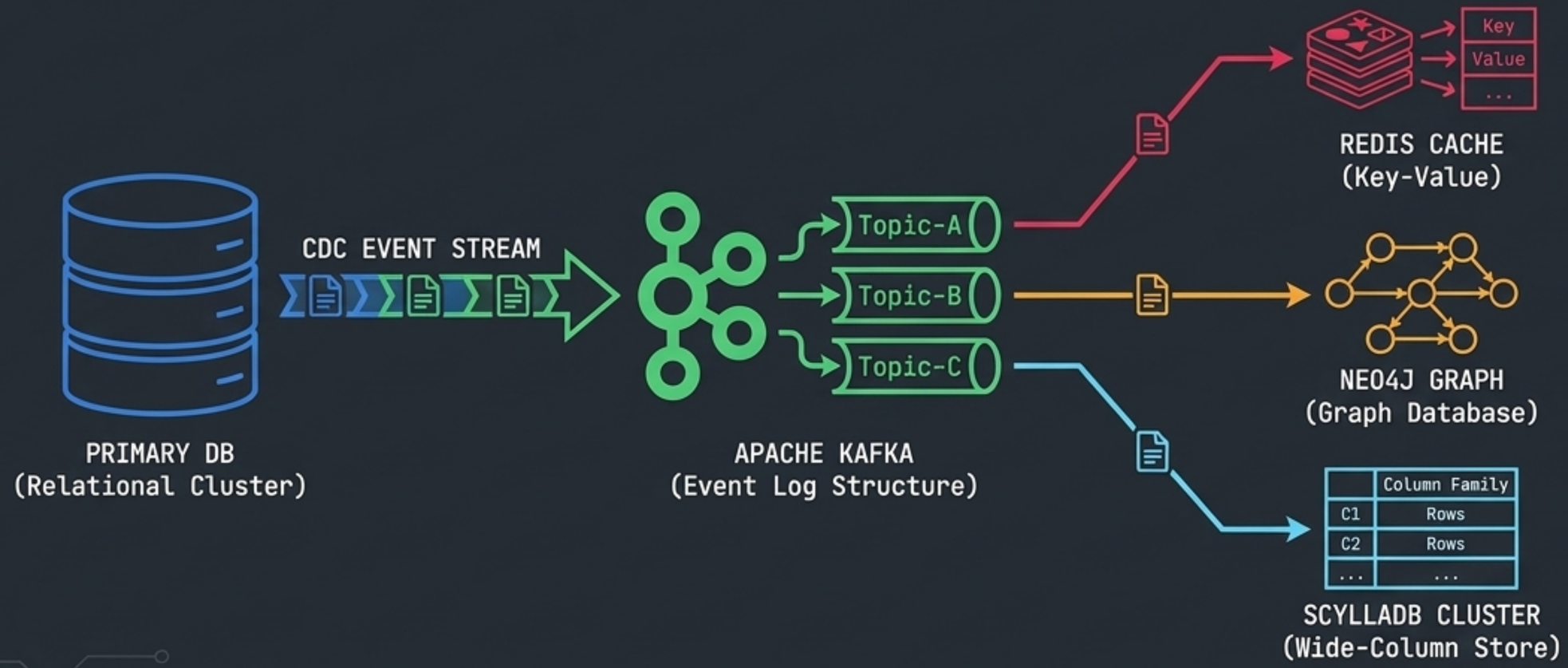


Gemini Notebook

Next Up in S05E04: Streaming and Event-Driven Data

The Reality: Pre-computing joins is hard. Keeping them synchronized across multiple operational data tiers is harder.

Next Episode: We explore how immutable event logs, Change Data Capture (CDC), and stream processing keep your denormalized NoSQL views safely in sync without distributed lock contention.



Gemini Notebook