Modern Data Stores

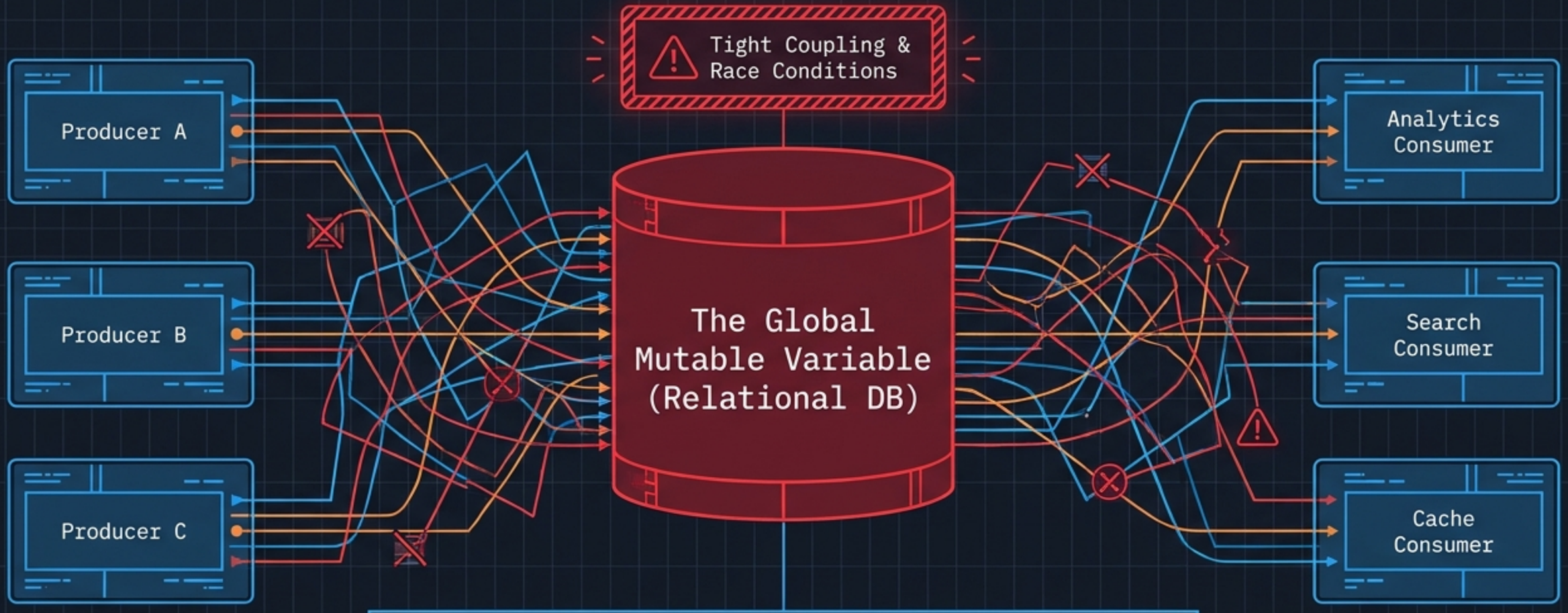
S05E04: Append-Only Event Logs, CDC, Stream Processing & What “Exactly-Once” Really Means

A Systems Engineering Deep Dive



Gemini Notebook

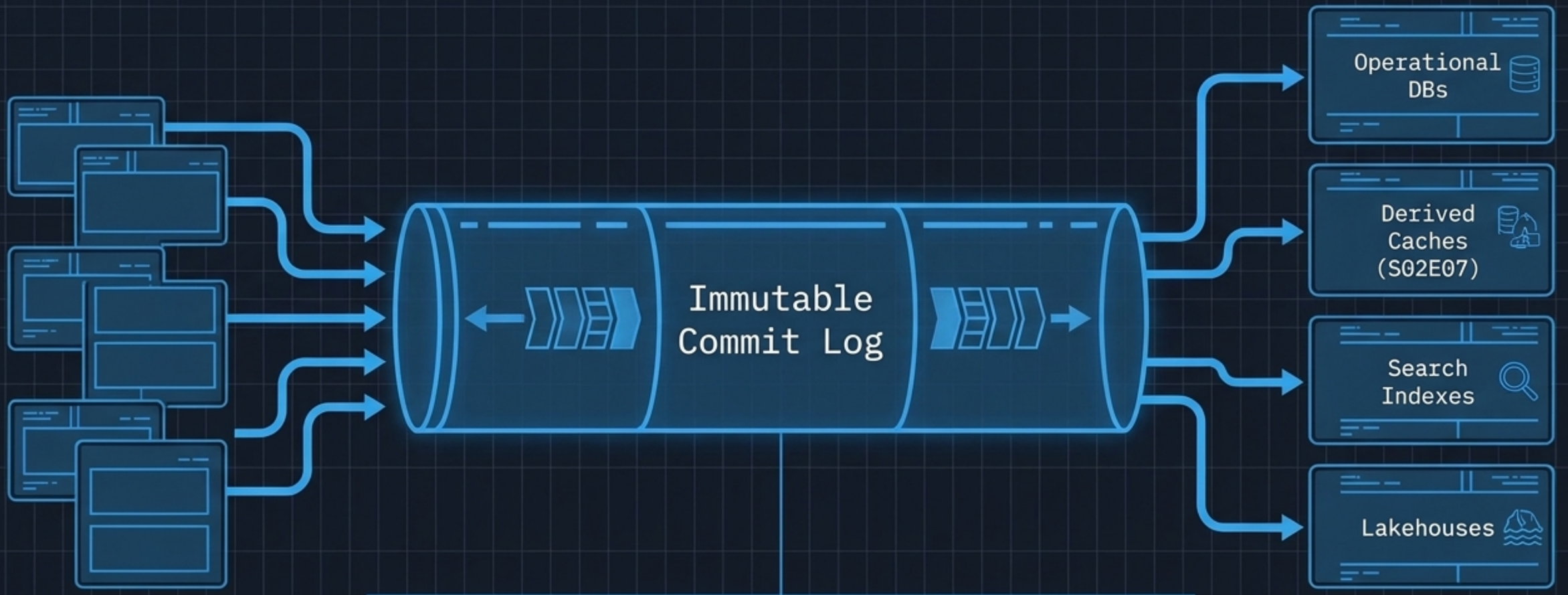
The Global Mutable State Problem



Recap of S05E03: Traditional architectures treat the database as a massive, shared global variable. Mutating state in-place forces tight coupling between data producers and downstream consumers, leading to fragile architectures at scale.

Gemini Notebook

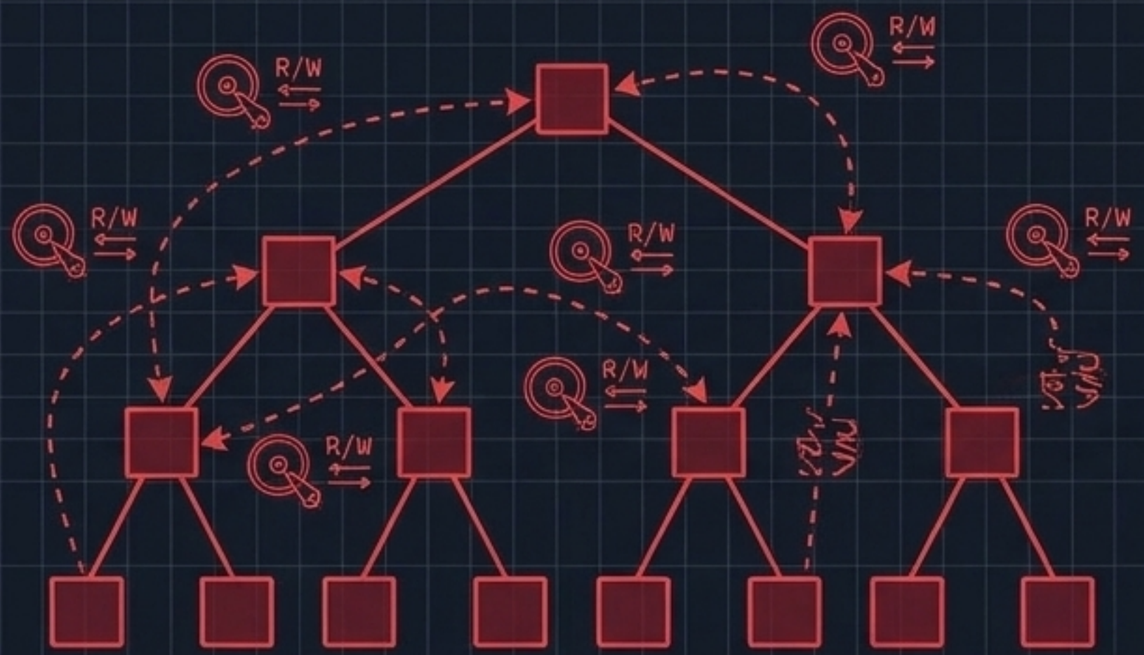
Turning the Database Inside Out



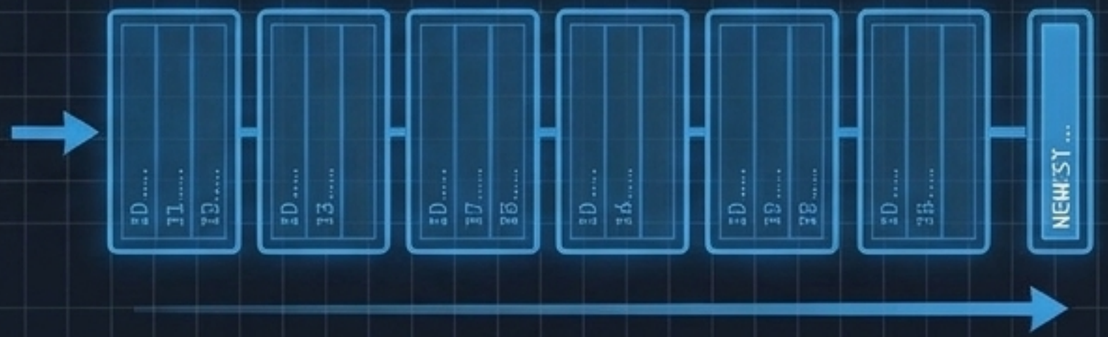
The unifying abstraction: extract the database's internal transaction log and make it the central nervous system. Producers append immutable facts; consumers build independent, derived materialized views.

Gemini Notebook

The Append-Only Partitioned Commit Log



O(log N) Random I/O

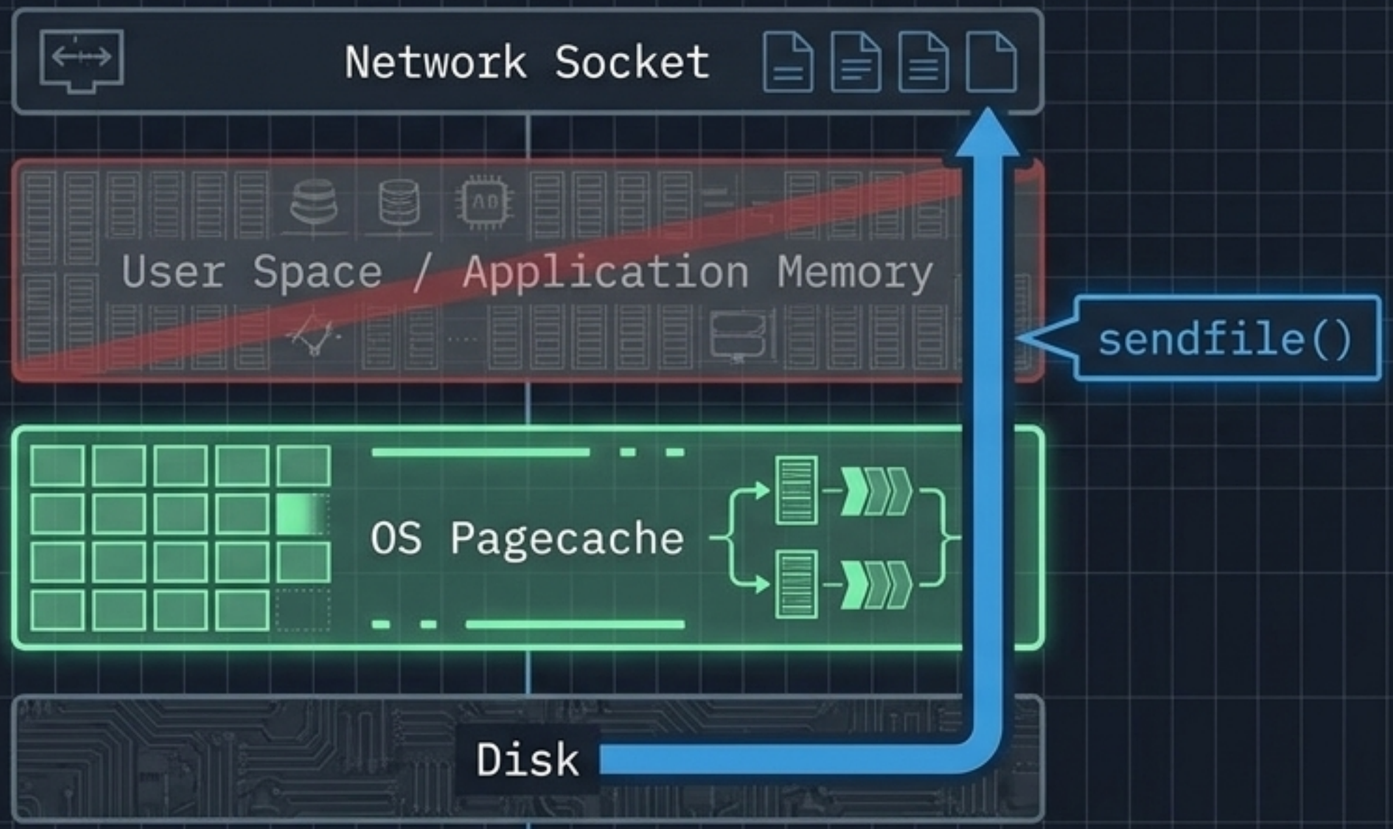


O(1) Sequential I/O

Kafka achieves massive throughput by treating the disk as a sequential tape drive. Immutable segment files guarantee O(1) sequential disk writes, bypassing the heavy penalty of random B-Tree mutations.

Gemini Notebook

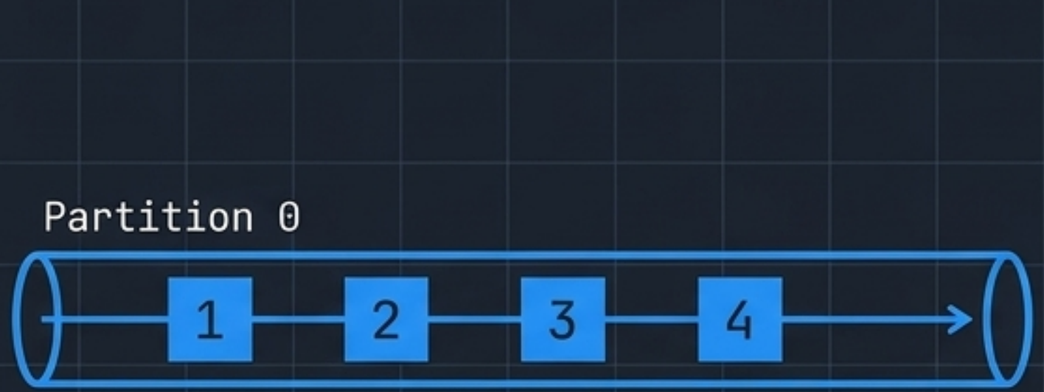
Zero-Copy I/O at Pagecache Speed



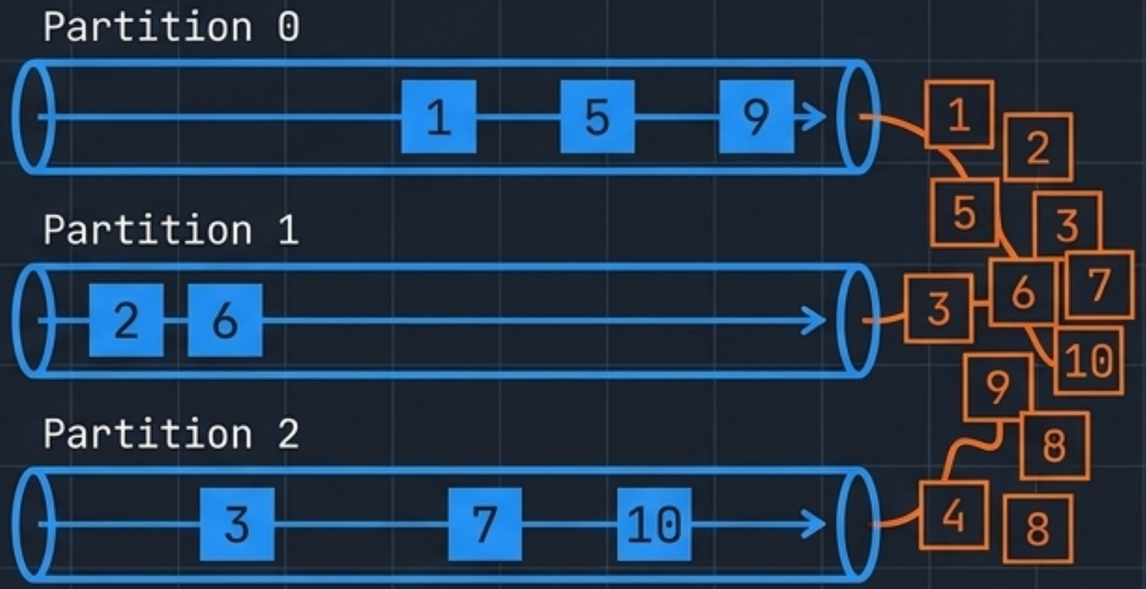
Data transfer bypasses user space entirely. By leveraging the OS pagecache and sendfile zero-copy network transfers, brokers serve data at the physical limits of the network interface.

Gemini Notebook

The Physics of Ordering



Strict Offset Ordering (Within Partition)

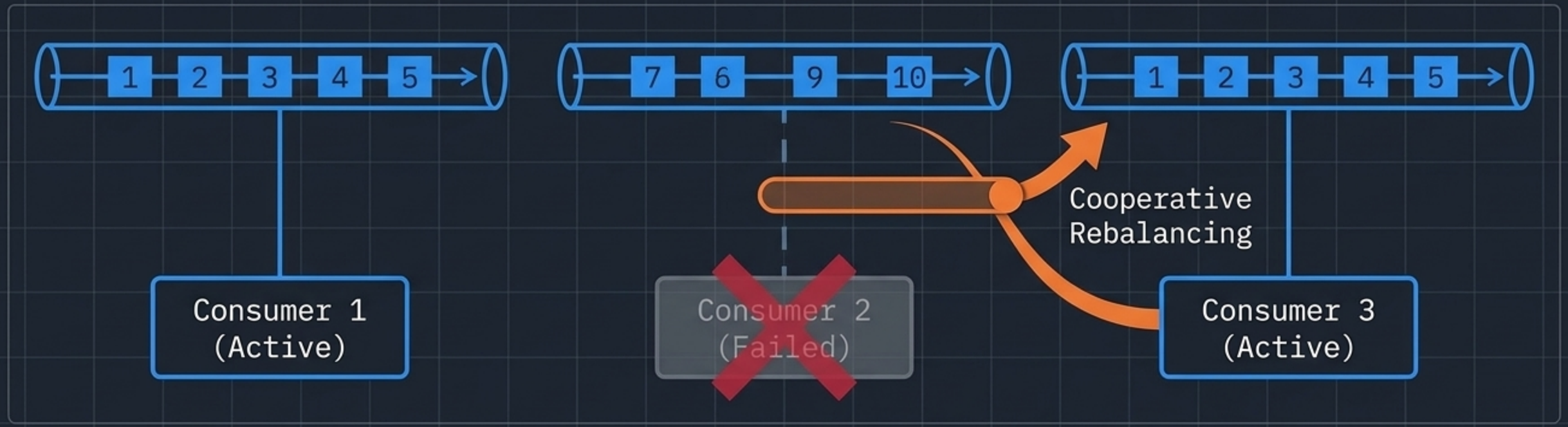


Unordered Concurrency (Cross-Partition)

Strict time-ordering is physically impossible at scale. Kafka guarantees absolute offset ordering only within a single, key-hashed partition, never globally across the cluster.

Gemini Notebook

Consumer Groups & Log Compaction

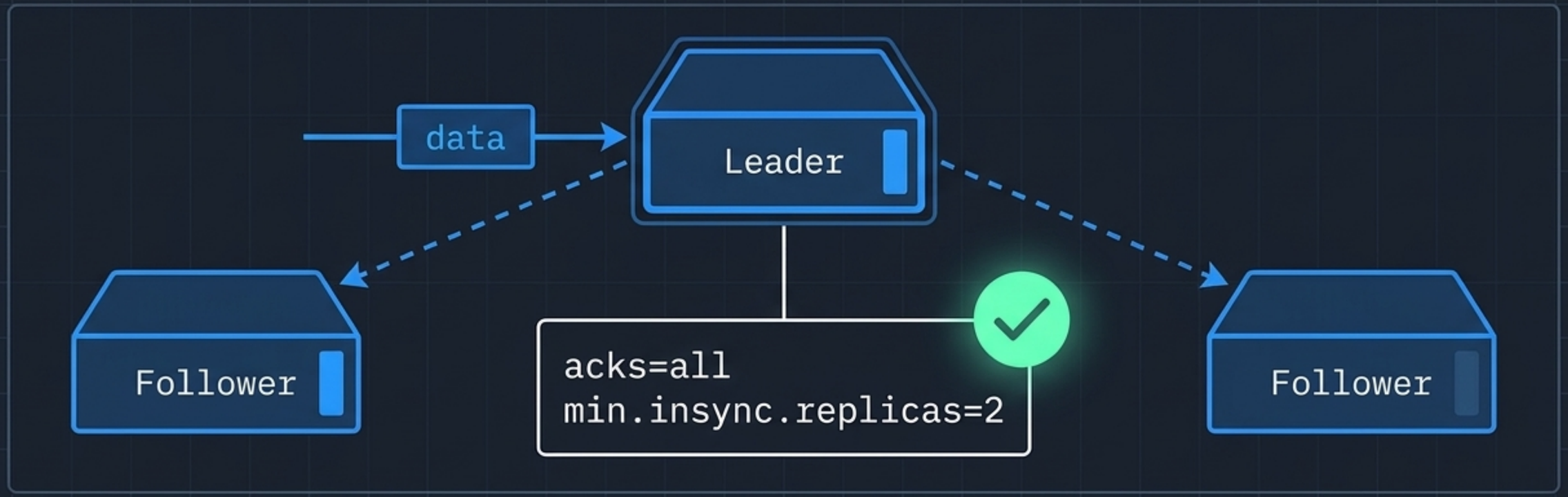




Consumer groups manage scale via offset tracking and cooperative rebalancing. For long-term state, key-based log compaction collapses history into the latest values and purges deletes via tombstones.

Gemini Notebook

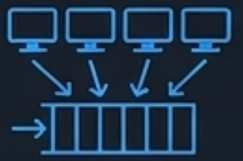
Broker Durability & Quorums



‘Effectively once’ starts with getting data safely into the log. Configuring acks=all with min.insync.replicas guarantees data survives leader elections and network partitions without silent drops.

Gemini Notebook

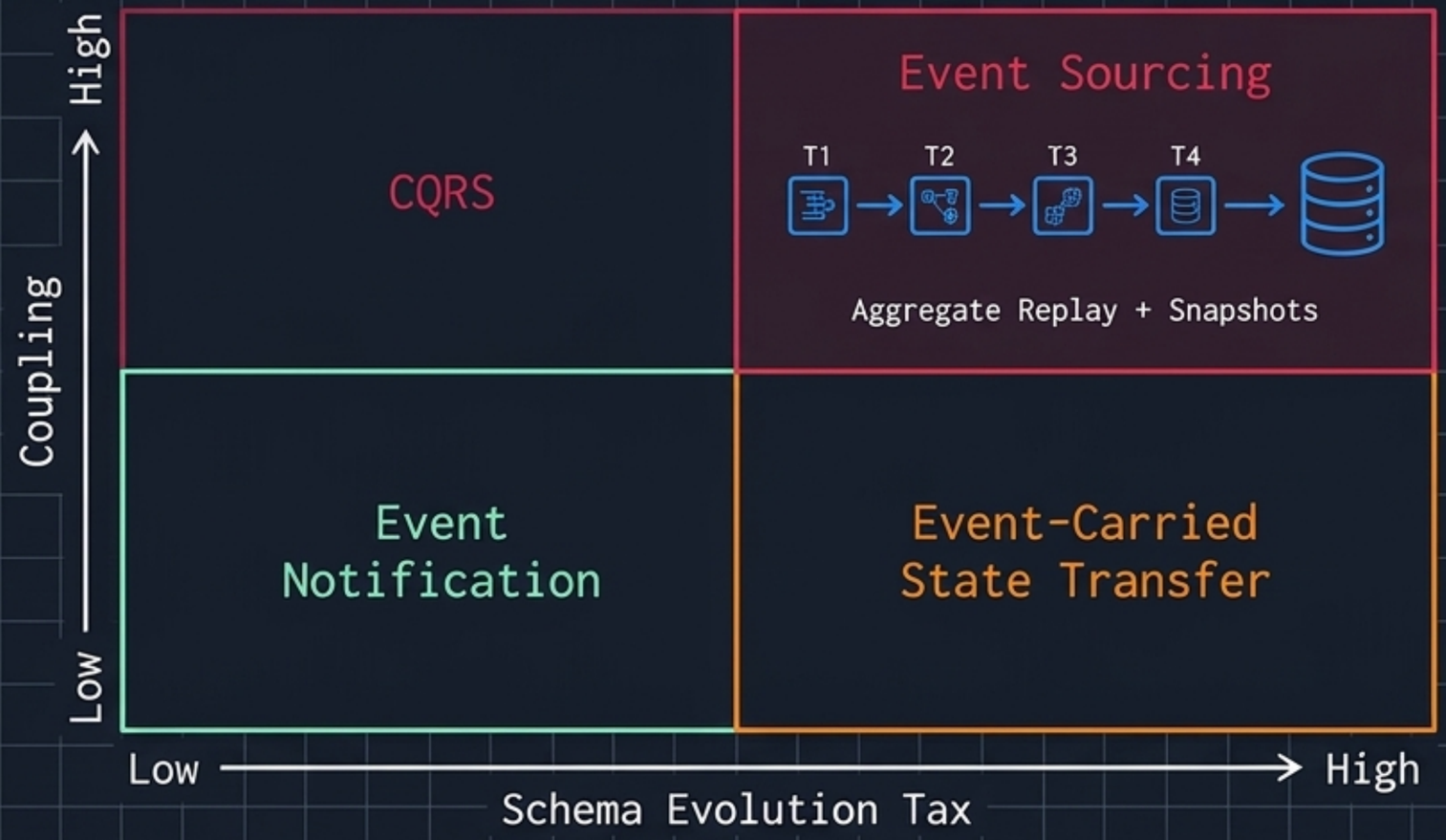
Message Brokers vs. Partitioned Logs

	Traditional MQs (RabbitMQ/SQS)	Partitioned Logs (Kafka)
Read Paradigm	Destructive / Ack-Delete 	Immutable Offset Cursor 
Routing	Smart Broker / Dumb Consumer 	Dumb Broker / Smart Consumer 
Replayability	Gone once consumed 	Independent Multi-Consumer Replay 

Unlike traditional work queues that destroy messages upon read, append-only logs decouple consumers, allowing independent systems to replay history at their own pace without interference.

Gemini Notebook

Fowler's Four Event Patterns



Not all events are equal. While Event Sourcing (replaying aggregates from birth) offers pristine auditability, it imposes a massive "schema evolution tax" requiring complex event-upcasters over time.

Gemini Notebook

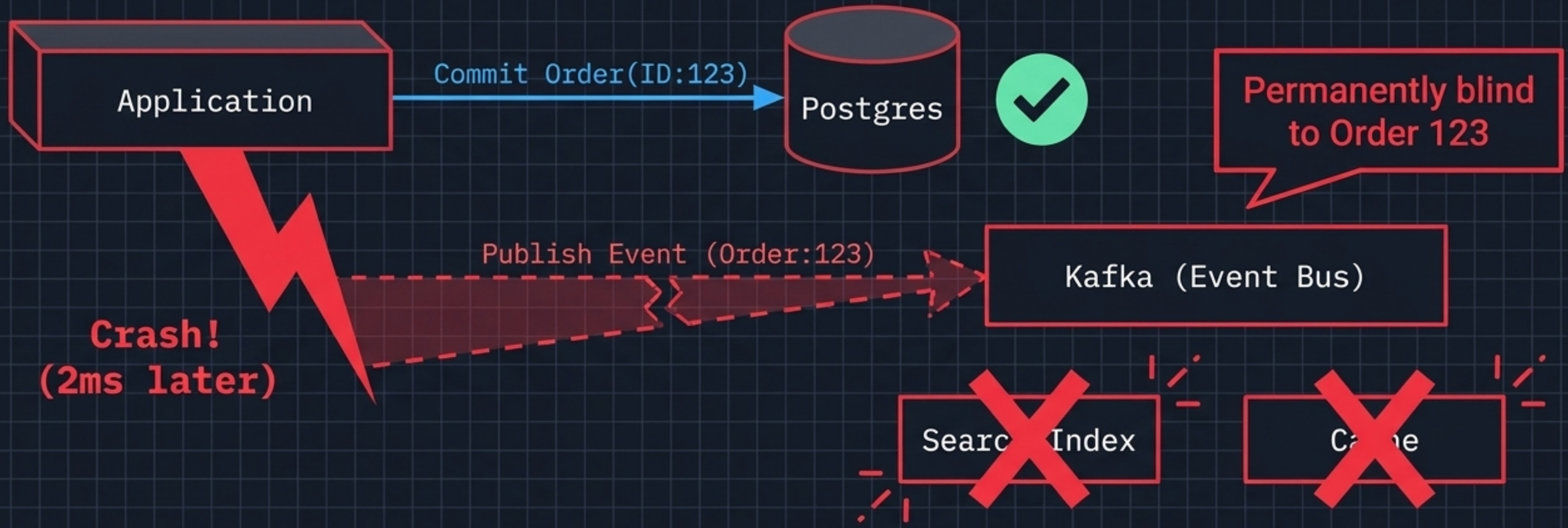
The Stream Engine Matrix

	Stateless Consumers	Kafka Streams	Apache Flink
State Management	None 	KStream / KTable + RocksDB 	Distributed Snapshots 
Use Case	Simple ETL Pipes 	Embedded in Microservices 	Heavy Analytical Streaming 
Operational Overhead	 Low	 Medium	 High

Choose the engine based on state. Stateless consumers suffice for simple ETL, Kafka Streams embeds local RocksDB state inside existing microservices, while Flink offers heavy-duty distributed processing.

Gemini Notebook

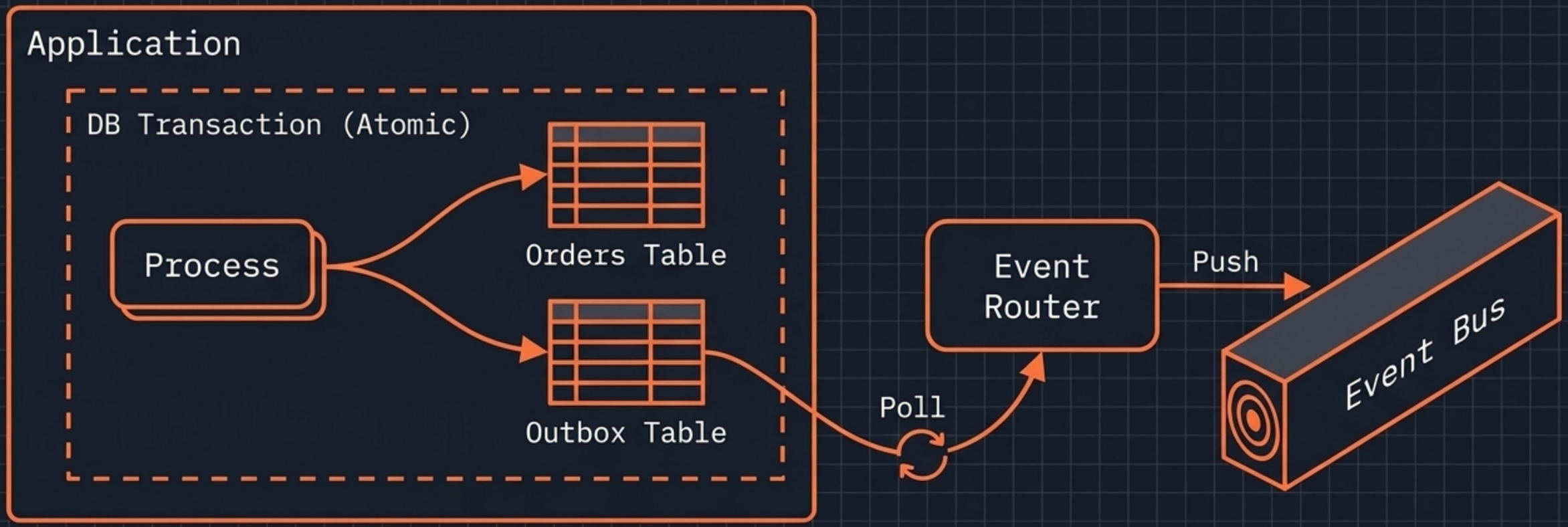
The Dual-Write Anti-Pattern



Writing to a database and a message broker from application code in a try/catch block guarantees eventual inconsistency. Partial crashes leave downstream caches and analytics permanently corrupted.

Gemini Notebook

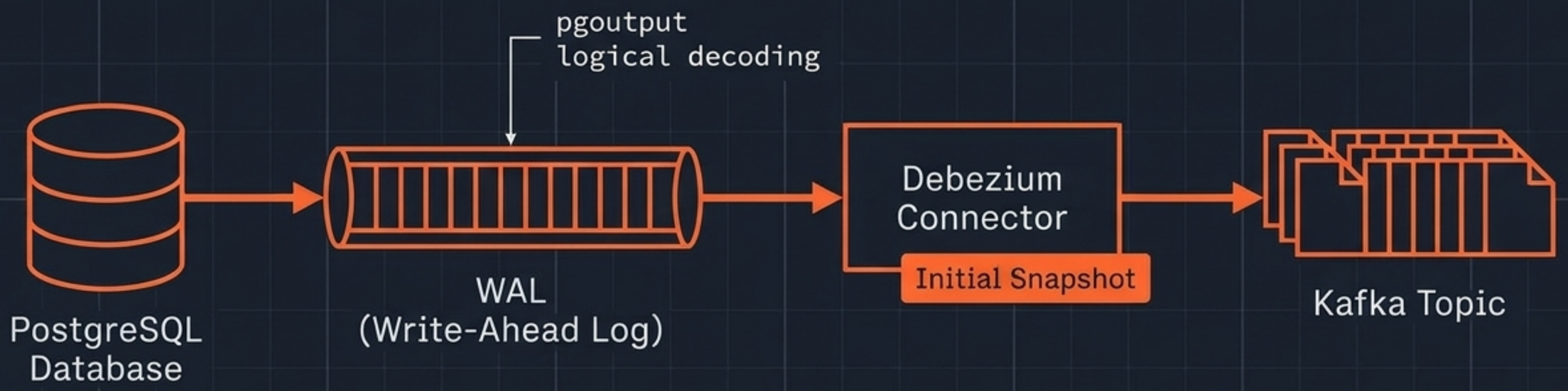
Transactional Outbox & Event Router



Guarantee atomicity by saving the state mutation and the outbound event within the same relational database transaction. If the transaction rolls back, the event is never published.

Gemini Notebook

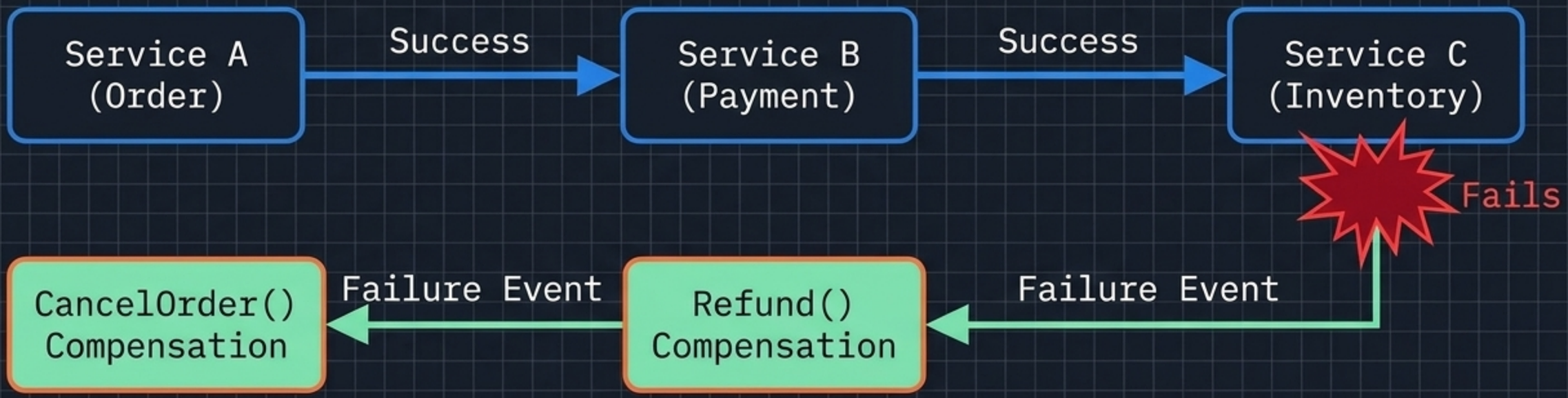
Log-Based CDC Pipelines



Eliminate polling overhead entirely. Debezium executes an initial snapshot, then tails the PostgreSQL WAL via logical decoding, reliably streaming committed outbox changes directly into the immutable log.

Gemini Notebook

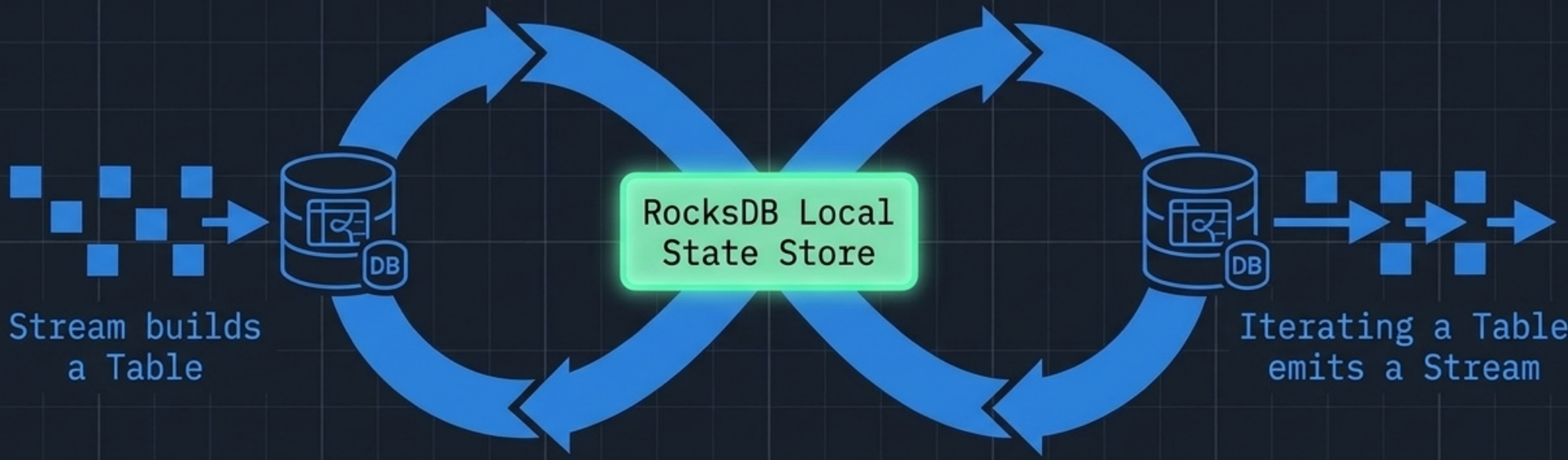
Sagas for Distributed Mutations



Distributed locking is a scalability killer. Sagas manage cross-service transactions through choreographed, asynchronous events, relying on explicit “compensating actions” to roll back state upon failure.

Gemini Notebook

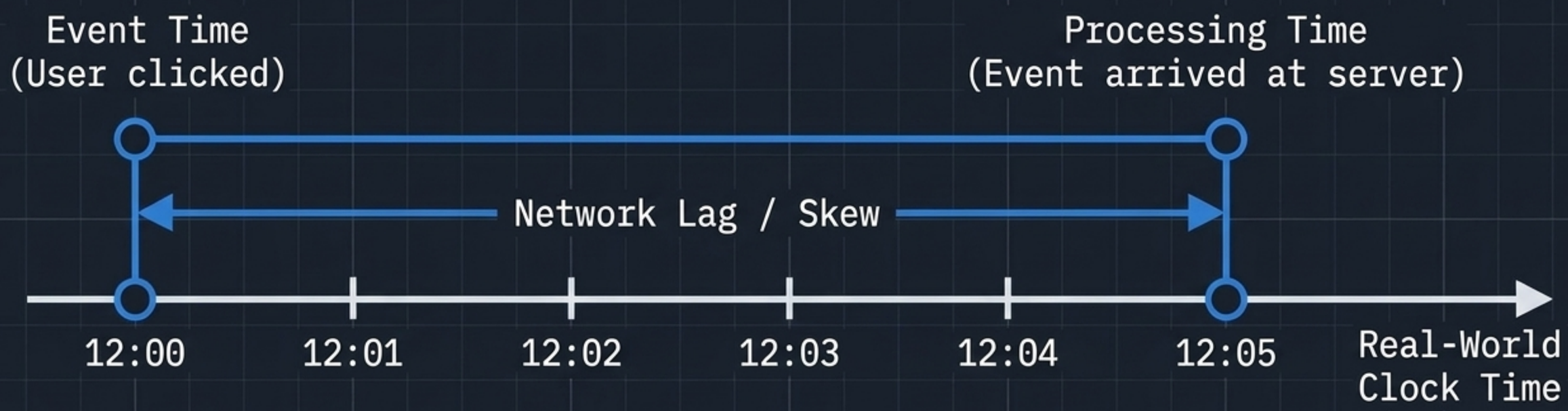
Stream-Table Duality



A stream is a table in motion; a table is a stream at rest. Kafka Streams leverages this duality, utilizing embedded RocksDB to materialize highly-available, queryable local state directly from changelogs.

Gemini Notebook

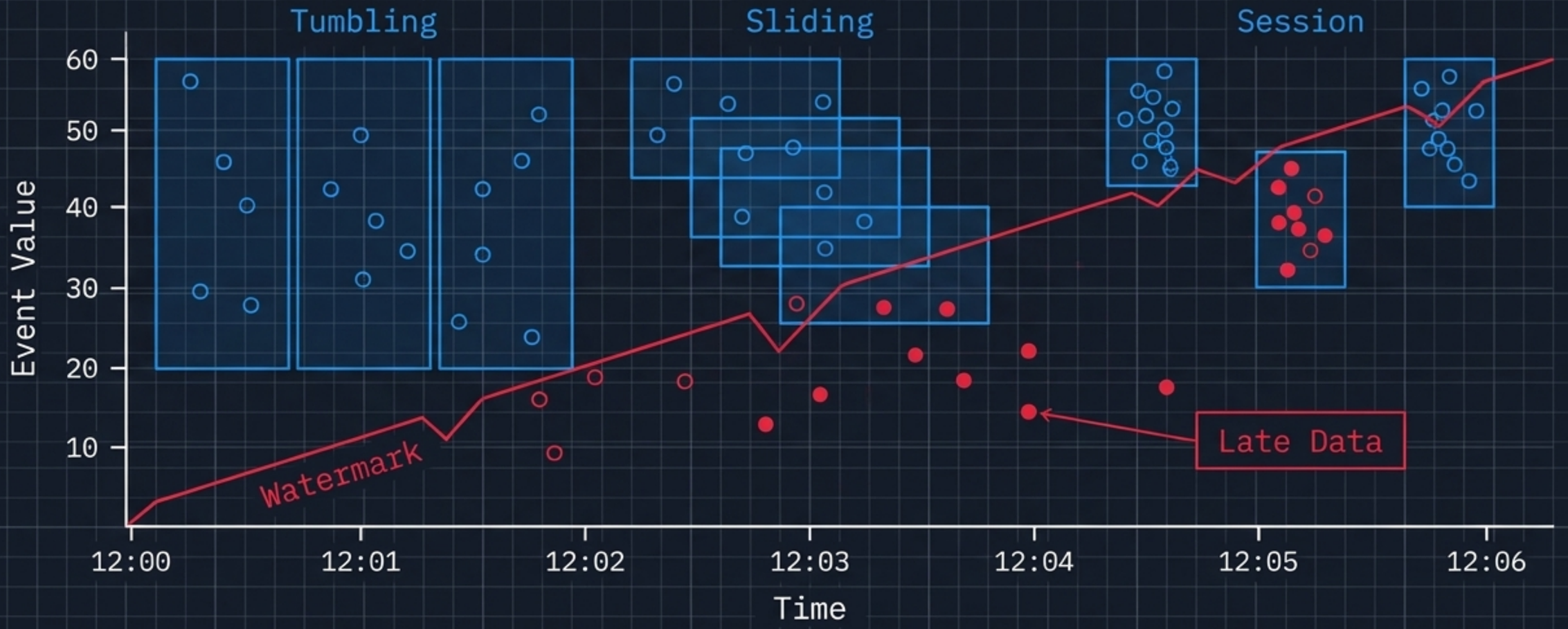
Time Semantics in the Dataflow Model



Processing unbounded data requires separating when an event occurred (Event Time) from when the system sees it (Processing Time). Stream engines must buffer and calculate based on Event Time to ensure correctness.

Gemini Notebook

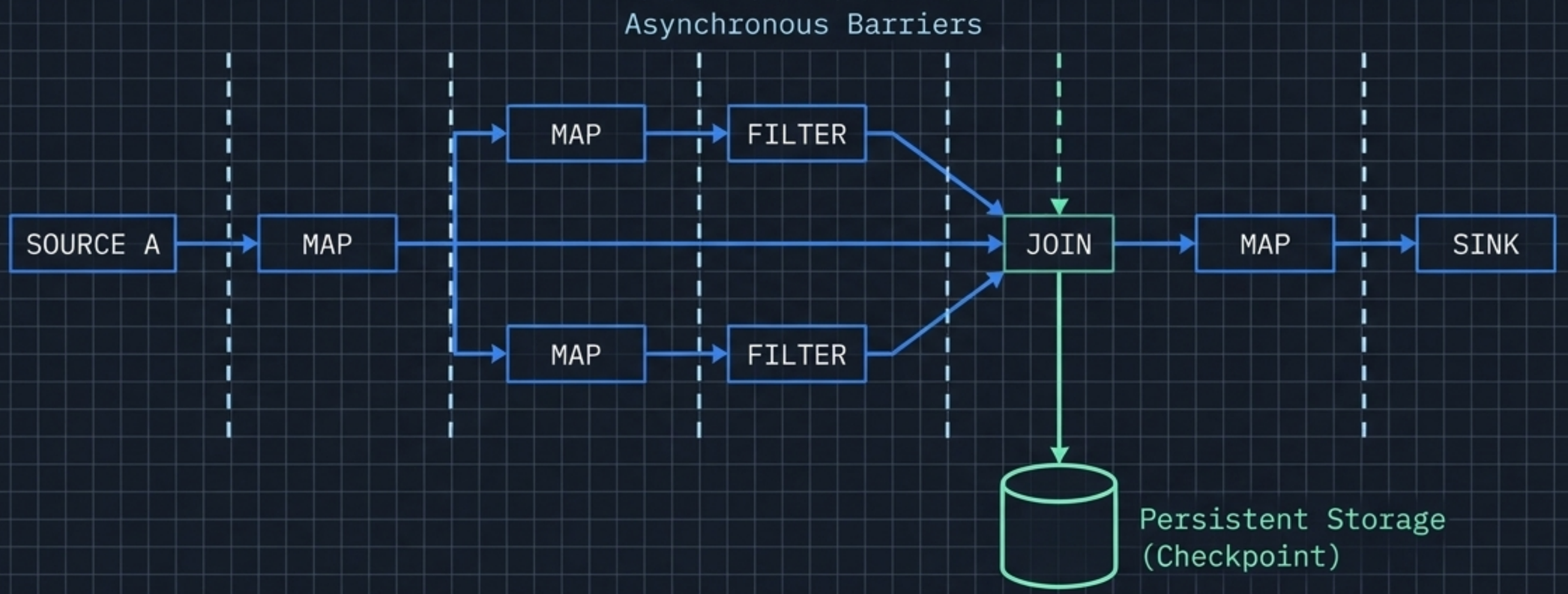
Windows, Watermarks & Late Data



Watermarks are heuristic thresholds that bound out-of-order network lag, signaling when a window should close. Late-arriving data triggers specific resolution paths, balancing latency against absolute completeness.

Gemini Notebook

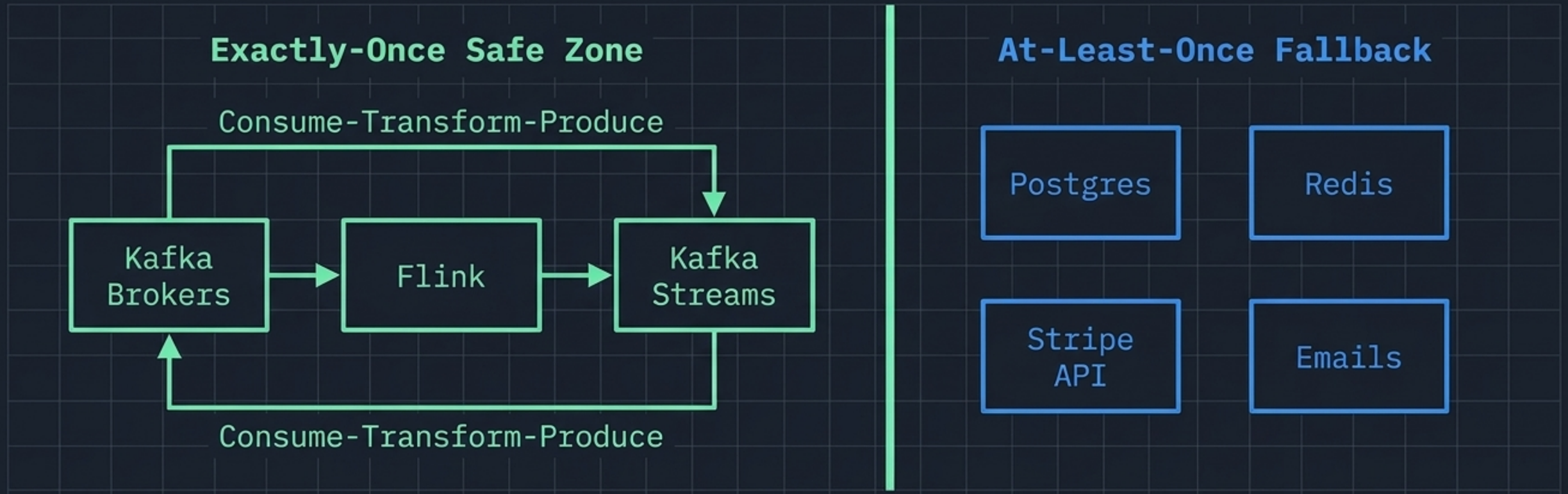
Chandy-Lamport Asynchronous Snapshots



Apache Flink guarantees exactly-once state via distributed checkpointing. Chandy-Lamport barriers flow continuously through the stream DAG, orchestrating globally consistent state snapshots without “stopping the world.”

Gemini Notebook

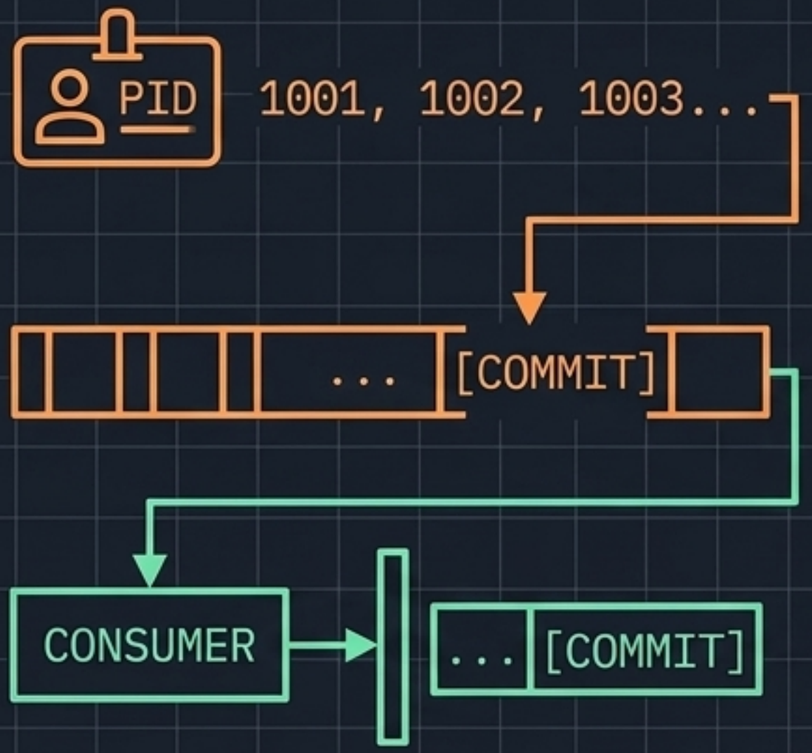
Deconstructing “Exactly-Once” Claims



The “Exactly-Once” reality check: KIP-98 and Flink checkpoints provide strict atomic guarantees only inside the broker/processor boundary. Moving data out to the real world fundamentally breaks this seal.

Gemini Notebook

Inside the Log: KIP-98 Mechanics

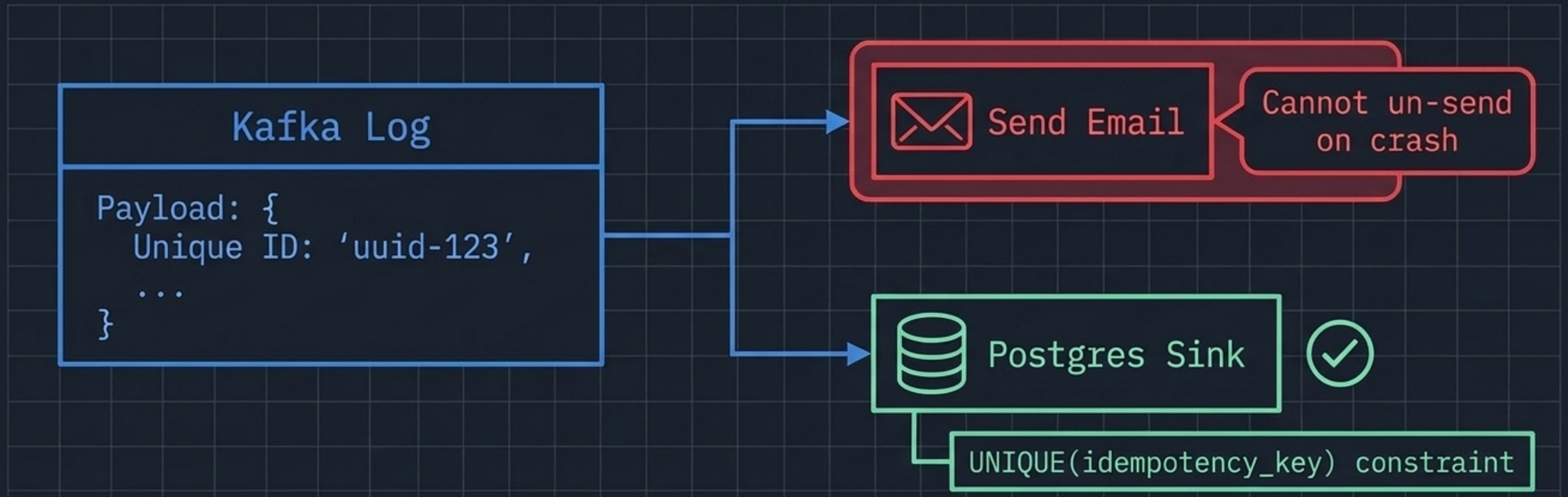


1. **enable.idempotence** assigns a Producer PID and Sequence Number.
2. **Two-Phase Transaction Coordinator** writes COMMIT markers to the log.
3. Consumer isolation level is set to `read_committed` to buffer uncommitted batches.

Kafka achieves transactional atomicity across multiple partitions using producer PIDs, strictly monotonically increasing sequence numbers, and a two-phase commit coordinator ensuring consumers only read finalized transactions.

Gemini Notebook

The External Sink Boundary

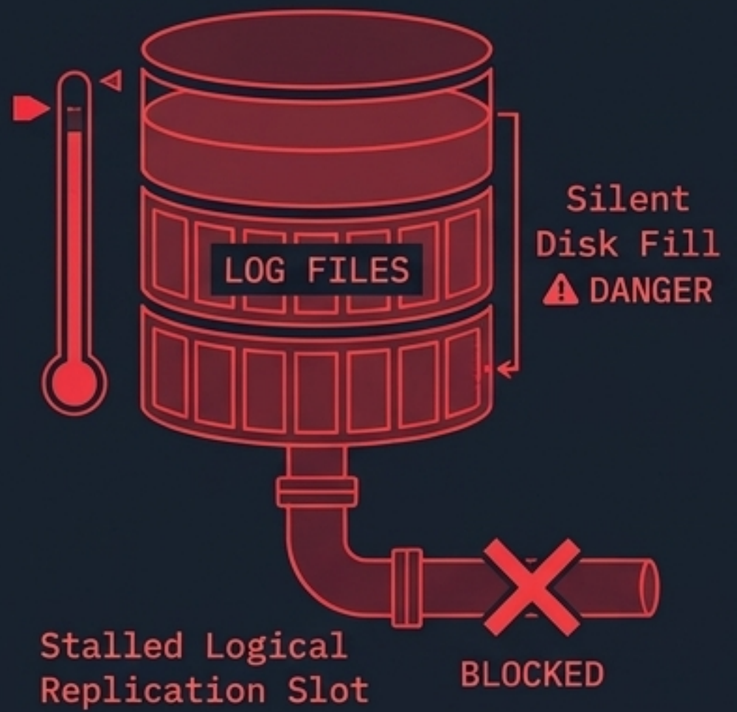


Because external side effects (charging credit cards, REST calls) cannot participate in Kafka’s internal commit markers, all external sinks must rely on deterministic, idempotent deduplication keys (S02E06).

Gemini Notebook

Operational Cliffs & Landmines

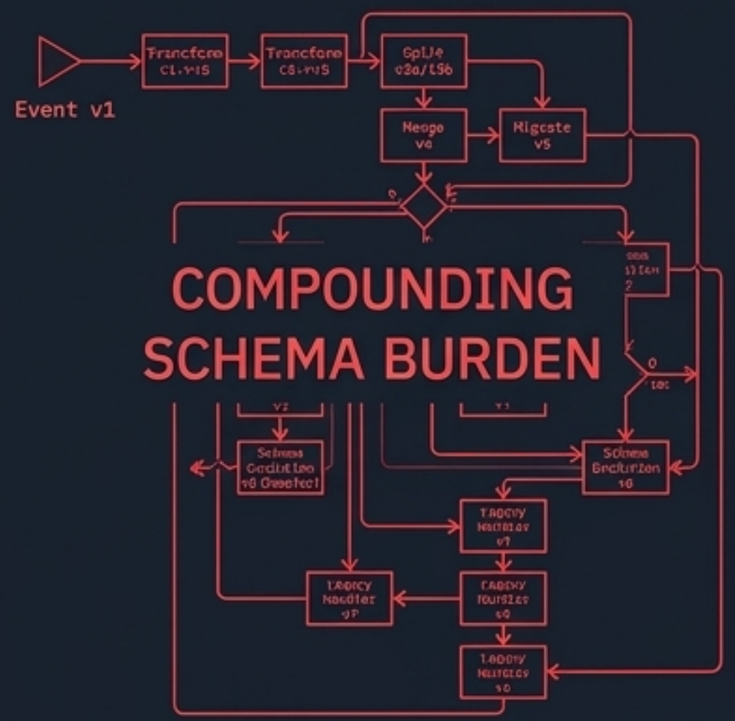
Disk Exhaustion



Poison Pills



Upcasting Debt

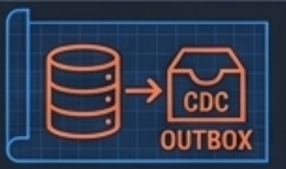


Beware the operational realities: stalled CDC slots will silently exhaust primary database disks, poison-pill messages require explicit dead-letter queues, and full Event Sourcing demands rigorous, long-term schema governance.

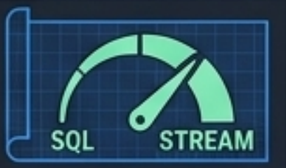
Gemini Notebook

Streaming Architecture Decision Checklist

1. Never dual-write app code. Use Outbox/CDC.



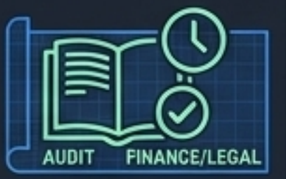
2. Synchronous SQL suffices for < 1K TPS. Stream over 1K.



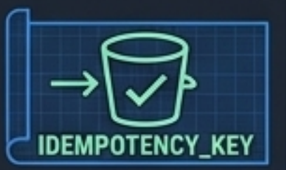
3. Size Kafka partitions strictly for consumer paralelism goals.



4. Only use full Event Sourcing for strict temporal audit domains (finance/legal).



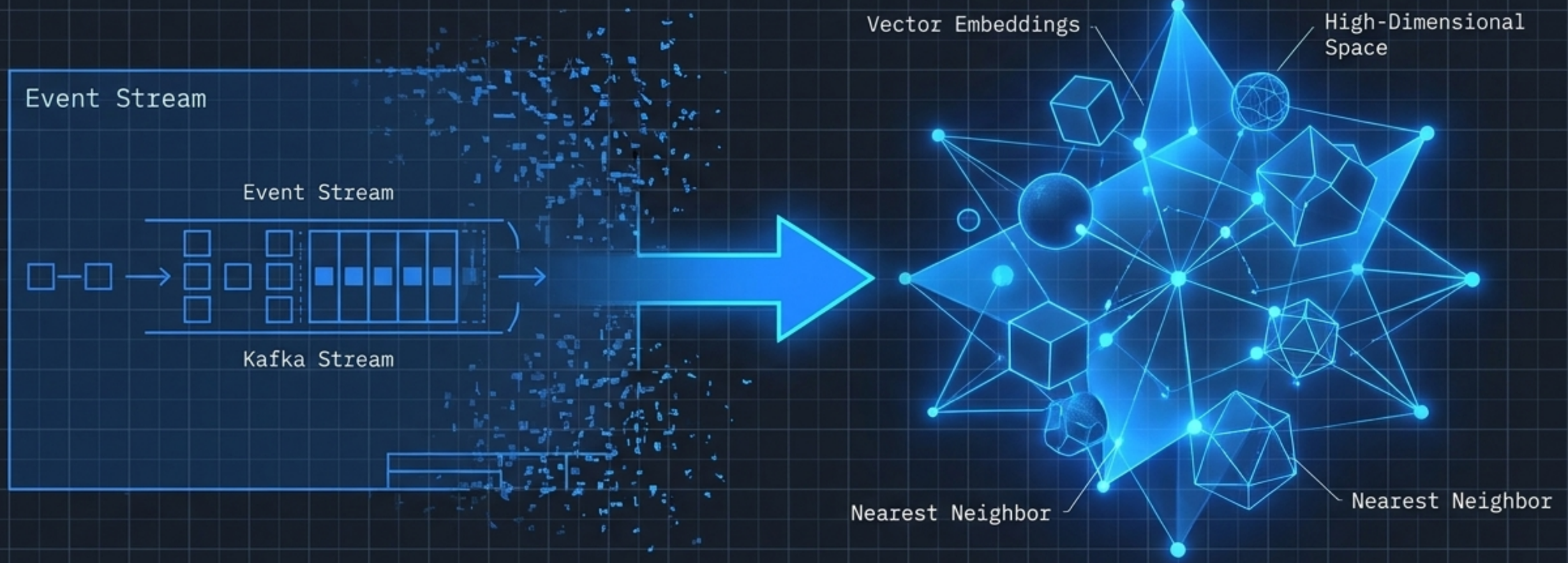
5. External sinks MUST be idempotent.



Quantified heuristics for distributed data design. Skip the hype and match the architectural complexity strictly to your organization’s throughput and audit requirements.

Gemini Notebook

Conclusion & Next Up



We've unbundled the database into reliable, real-time streams of truth. Next week in S05E05: 'Vector Databases'—How to feed these real-time event pipelines directly into RAG and AI embedding architectures.

Gemini Notebook