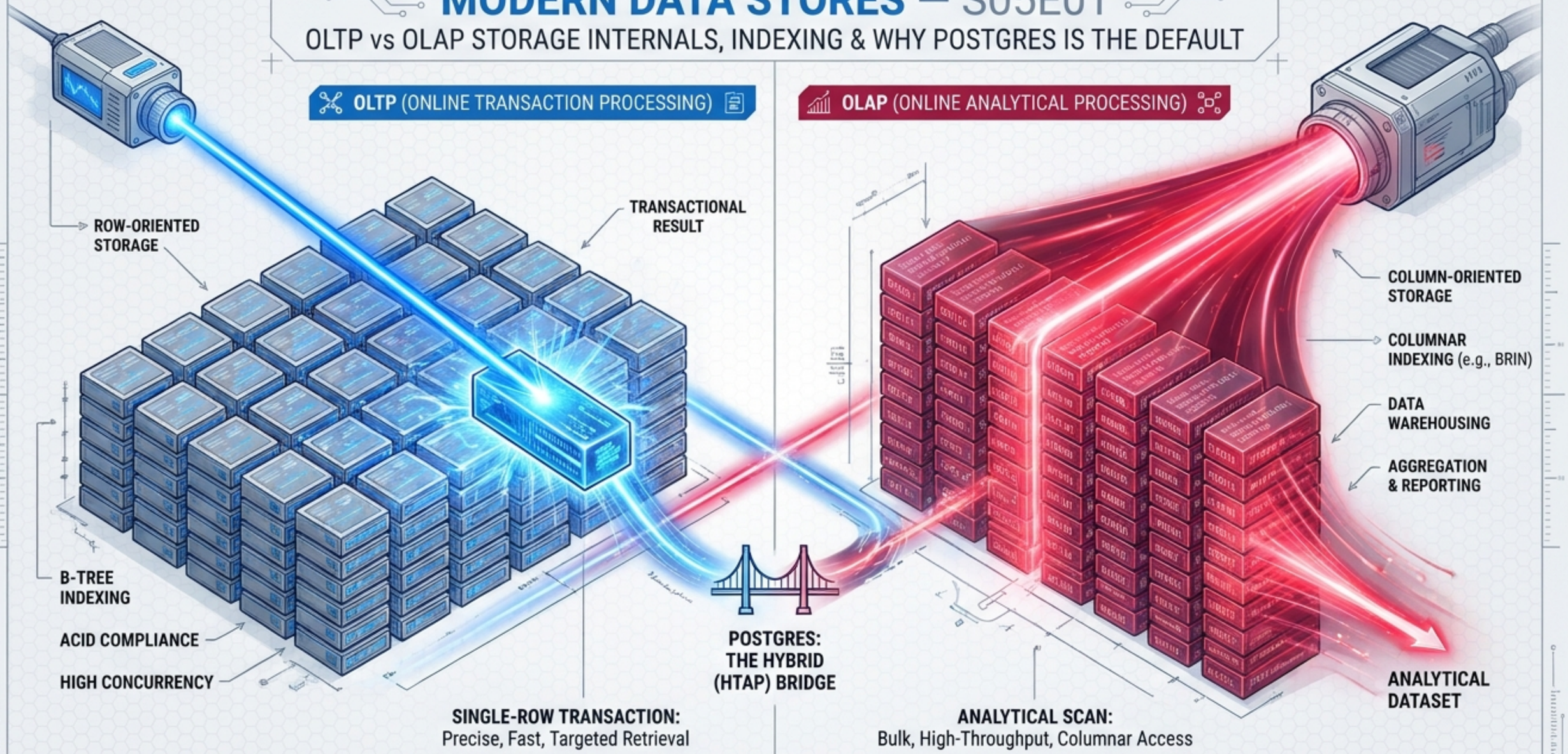
MODERN DATA STORES — S05E01

OLTP vs OLAP STORAGE INTERNALS, INDEXING & WHY POSTGRES IS THE DEFAULT



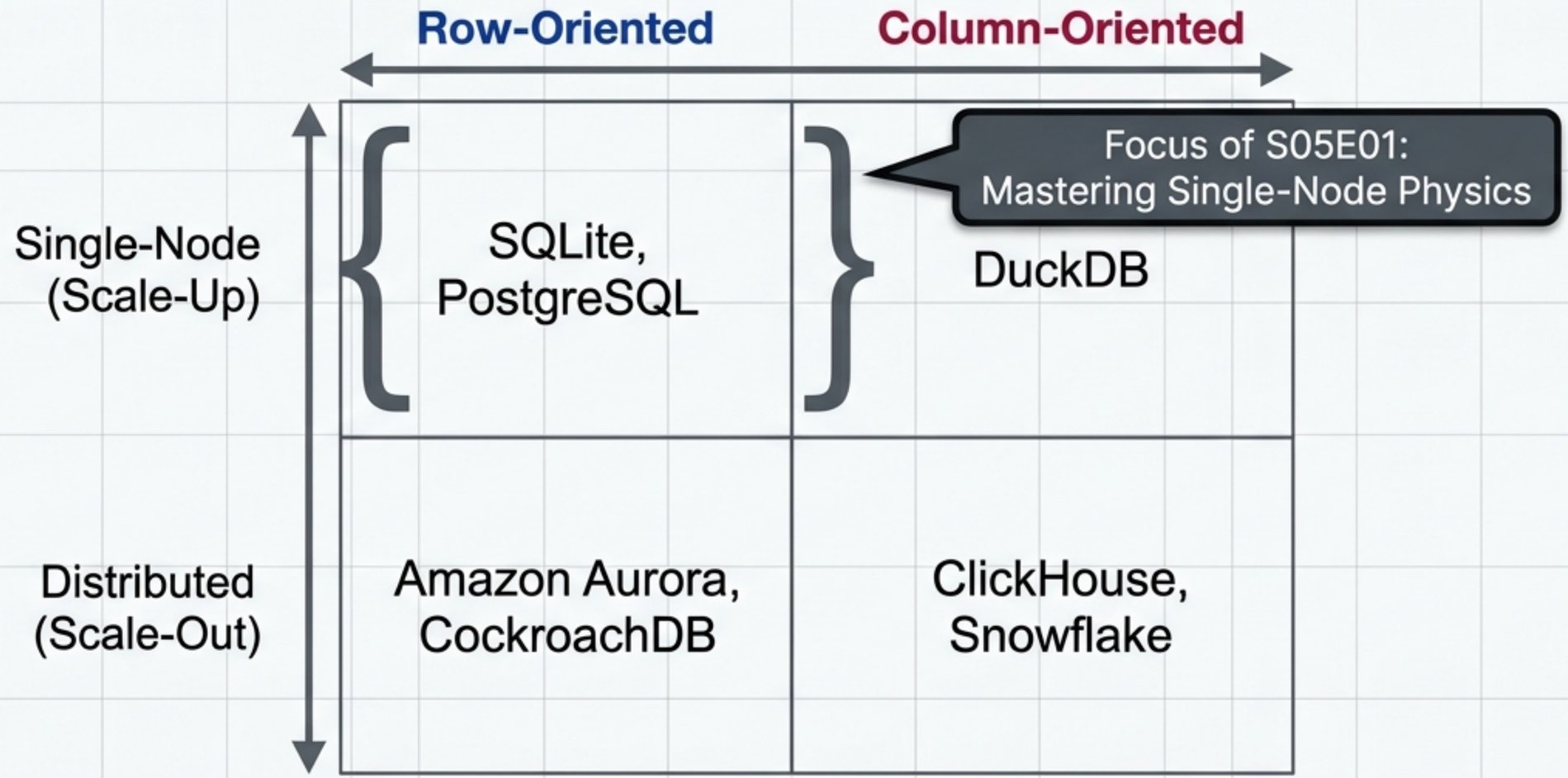
Postgres leverages versatile storage engines, extensible indexing (e.g., GIN, GIST, BRIN), and a powerful query planner to handle diverse workloads effectively, making it the robust default choice for many modern applications.

Design: System Schematic / Light Mode / Engineering Print. S05E01

Gemini Notebook

The Fundamental Data Store Taxonomy

Data stores divide along two primary axes: the **physical layout** of bytes on disk and the architectural topology of the system.

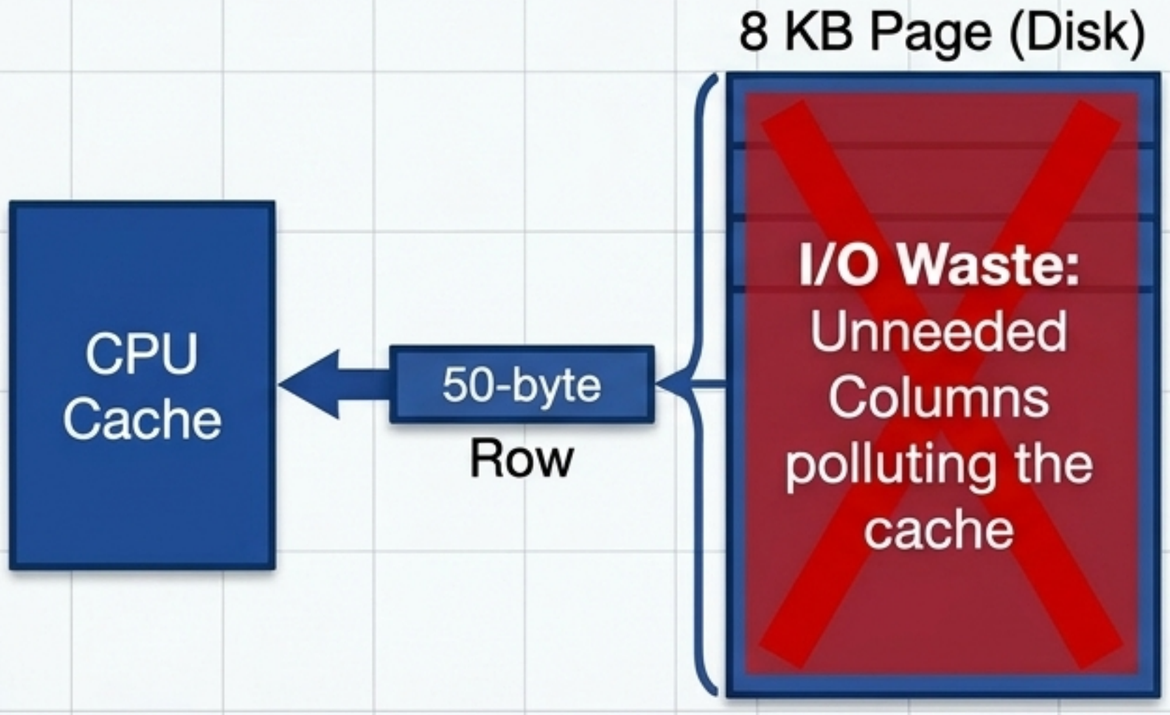


Gemini Notebook

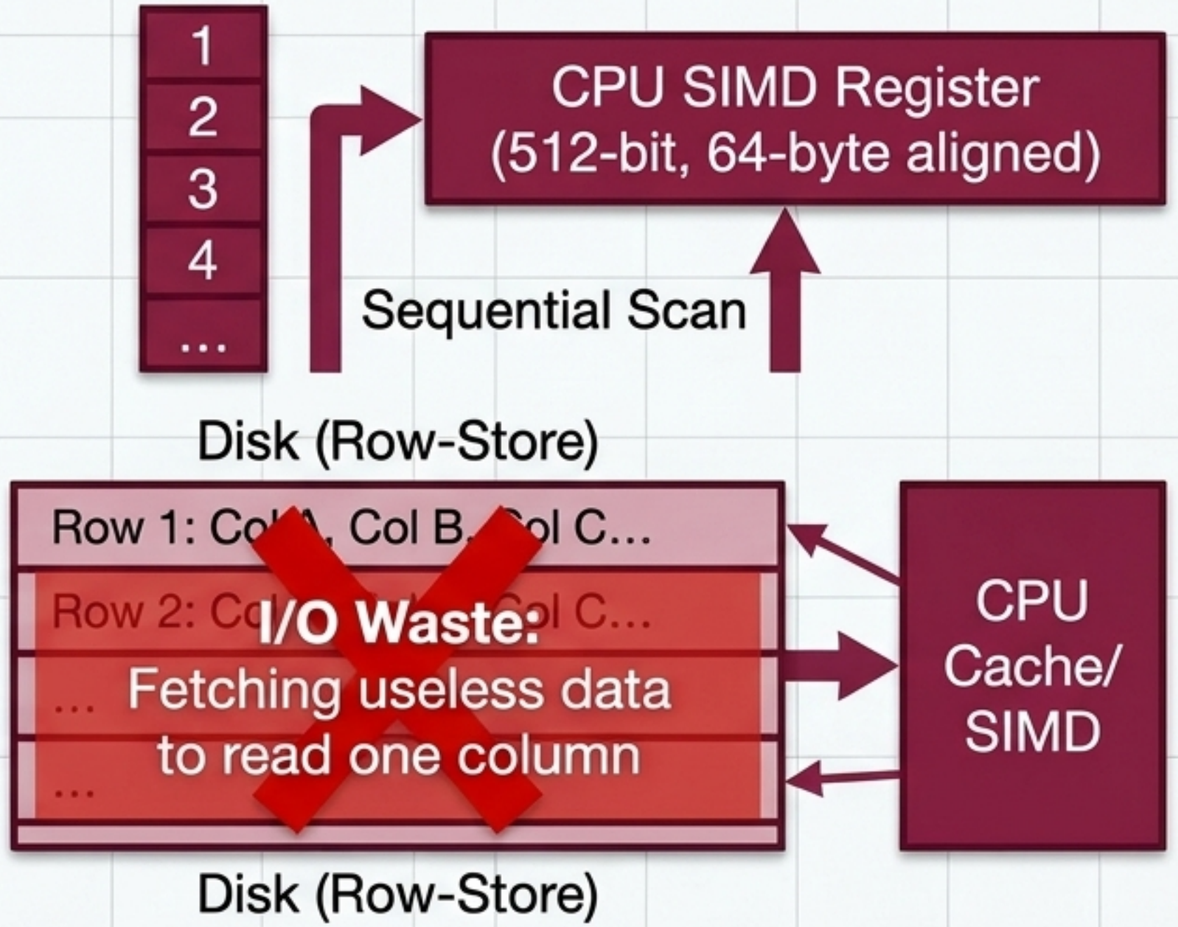
The Physics of Access Patterns

Single-row transactions and wide-table analytics pull CPU caches and disk I/O in violently opposite directions.

OLTP: Single-Row Transactions



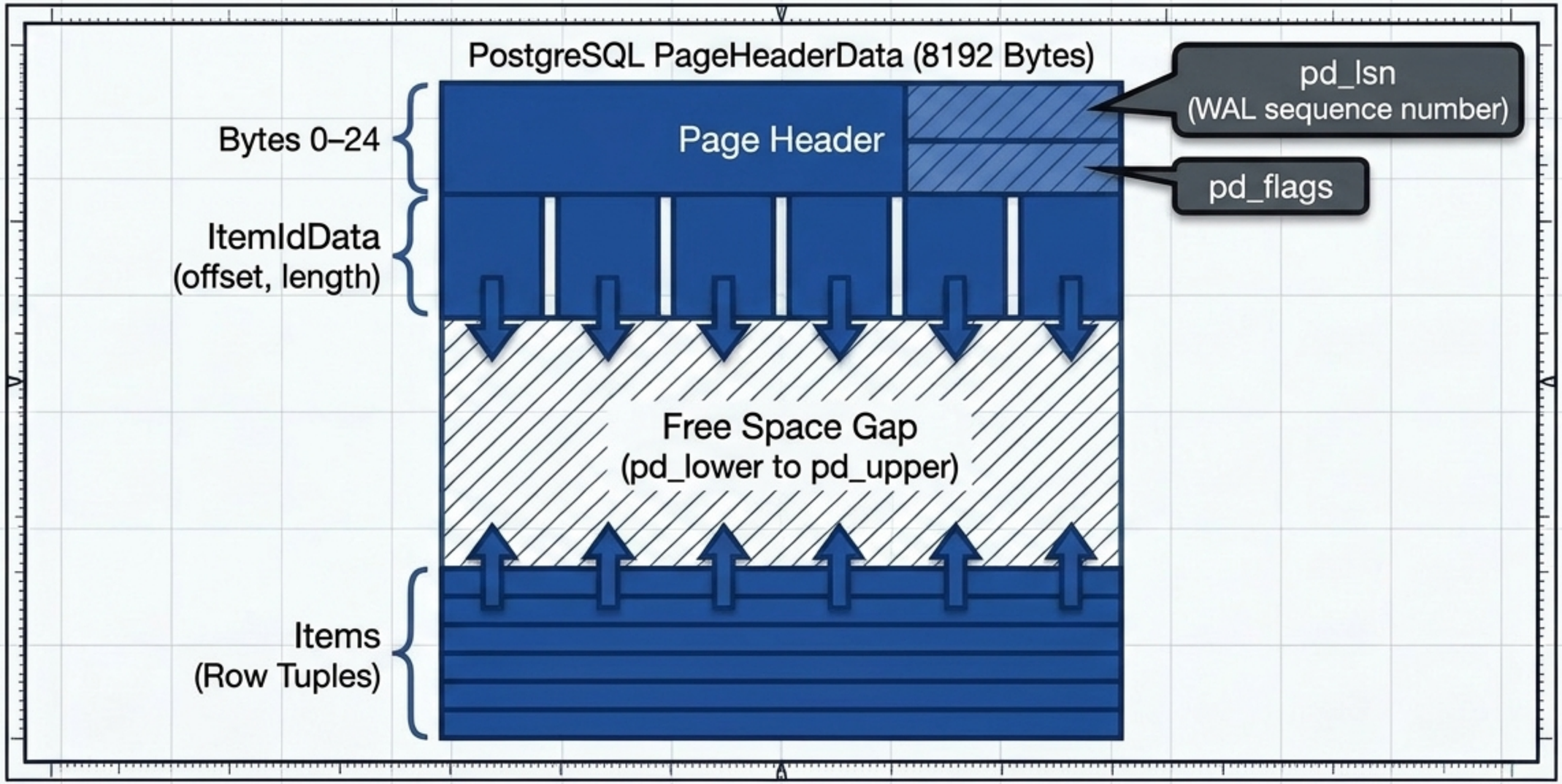
OLAP: Wide-Table Analytics



Gemini Notebook

Inside the OLTP Row Engine: The 8 KB Heap Page

The fundamental unit of operational storage. PostgreSQL physically packs row data by growing pointers forward and data backward.



Gemini Notebook

Sequential Durability and the Write-Ahead Log

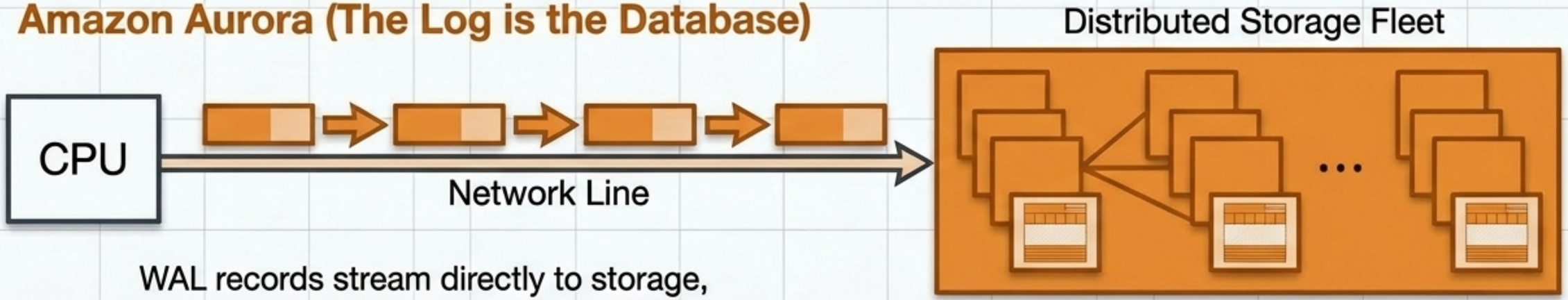
Durability ensures survival without forcing random I/O write penalties on the heap.

PostgreSQL / SQLite (Traditional Architectures)



Data is durably written to the sequential WAL, while the actual heap page update is deferred.

Amazon Aurora (The Log is the Database)

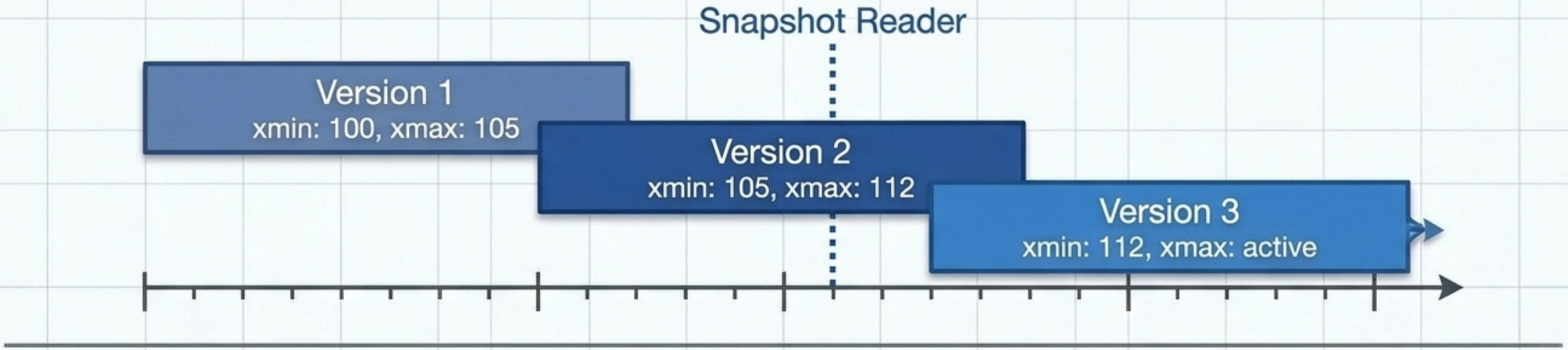


WAL records stream directly to storage, bypassing the primary compute disk. Storage nodes continuously materialize heap pages.

Gemini Notebook

MVCC and ACID Isolation Anomalies

How databases read the past while writing the future without locking rows.

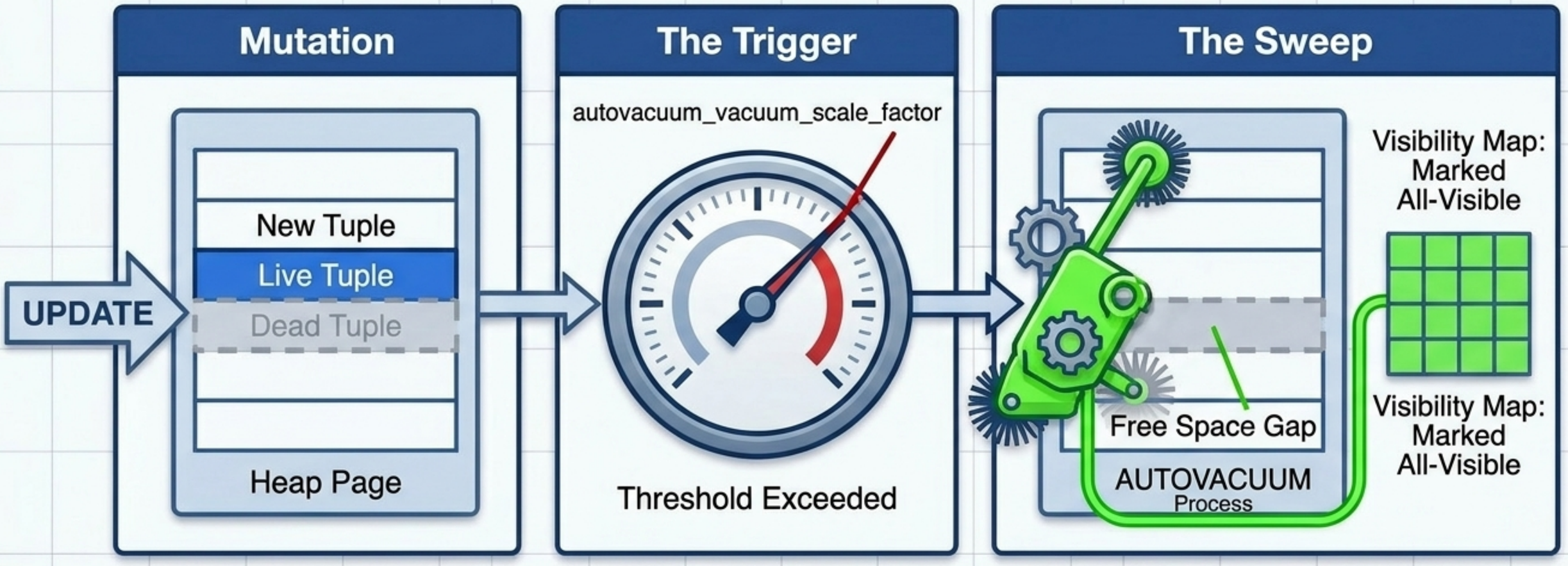


Isolation Level	Anomaly Blocking Capability
Read Committed	Blocks Dirty Reads
Repeatable Read	Blocks Non-Repeatable Reads
Serializable	Uses Predicate Locking to block Phantom Reads & Serialization Anomalies

Gemini Notebook

The Cost of MVCC: Dead Tuples and Autovacuum

The unavoidable tax of non-blocking writes: ghost rows, bloat, and visibility map maintenance.



Mutation

Threshold Exceeded

Reclaiming Dead Space & Updating Visibility Map

Gemini Notebook

The Great Engine Comparison Matrix

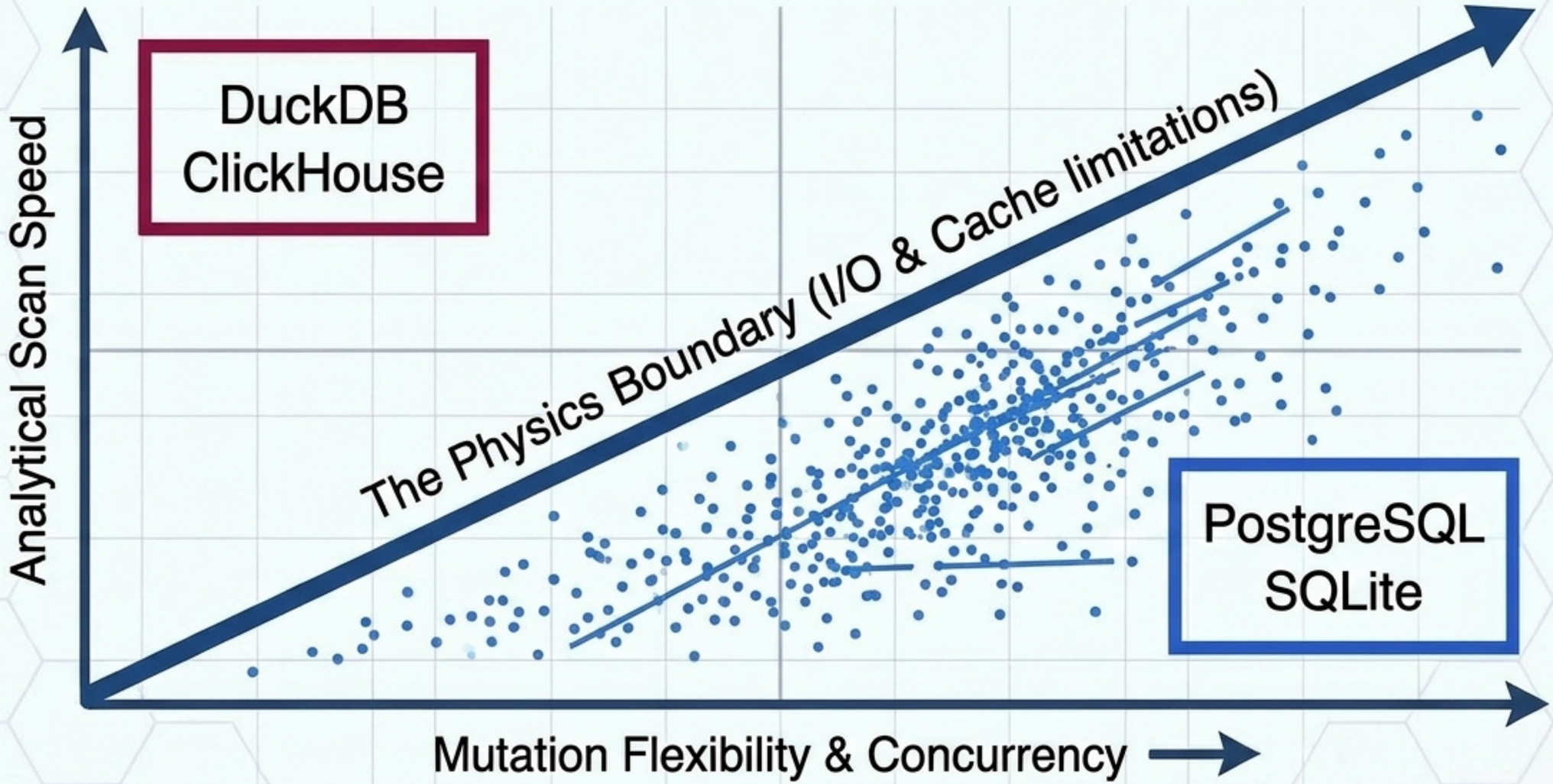
Categorical synthesis of storage architectures and their operational physics.

Engine	Storage Layout	Concurrency	Point R/W Latency	Billion-Row Scan Speed	Mutation Cost
SQLite	**Row**	B-Tree	Sub-millisecond	Slow	**Low**
PostgreSQL	**Row**	MVCC B-Tree	**Millisecond**	Slow	Medium
Aurora	**Row**	Log-structured distributed	**Millisecond**	Moderate	**Low**
DuckDB	**Column**	Vectorized MVCC	Moderate	**Sub-second**	High
ClickHouse	**Column**	MergeTree parts	High/Batched	**Milliseconds**	Extreme (Hostile to single-row updates)

Gemini Notebook

The Immutable Law of Storage Layouts

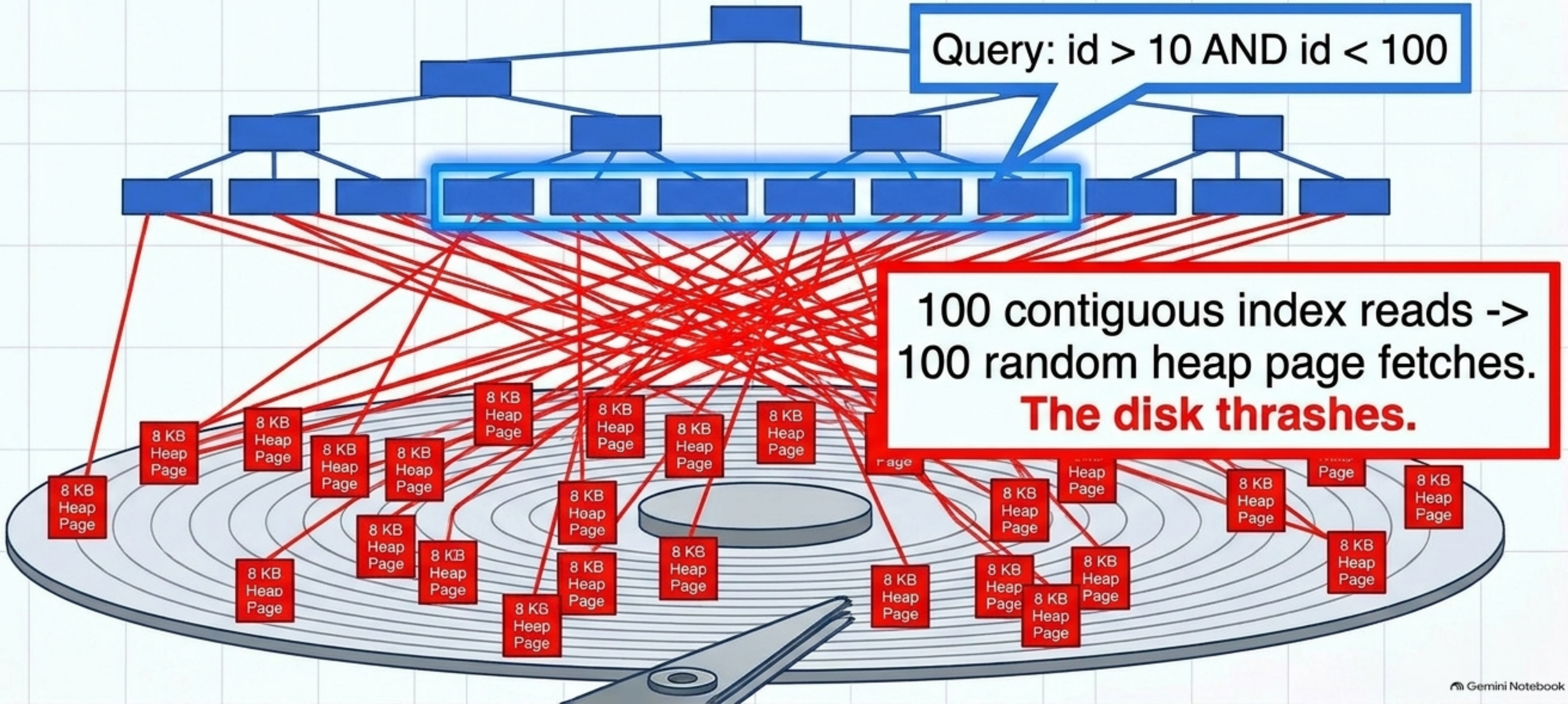
High-speed analytics requires rigid, heavily compressed data structures that are fundamentally hostile to random mutation.



Gemini Notebook

Dense B-Tree Anatomy and the Random I/O Trap

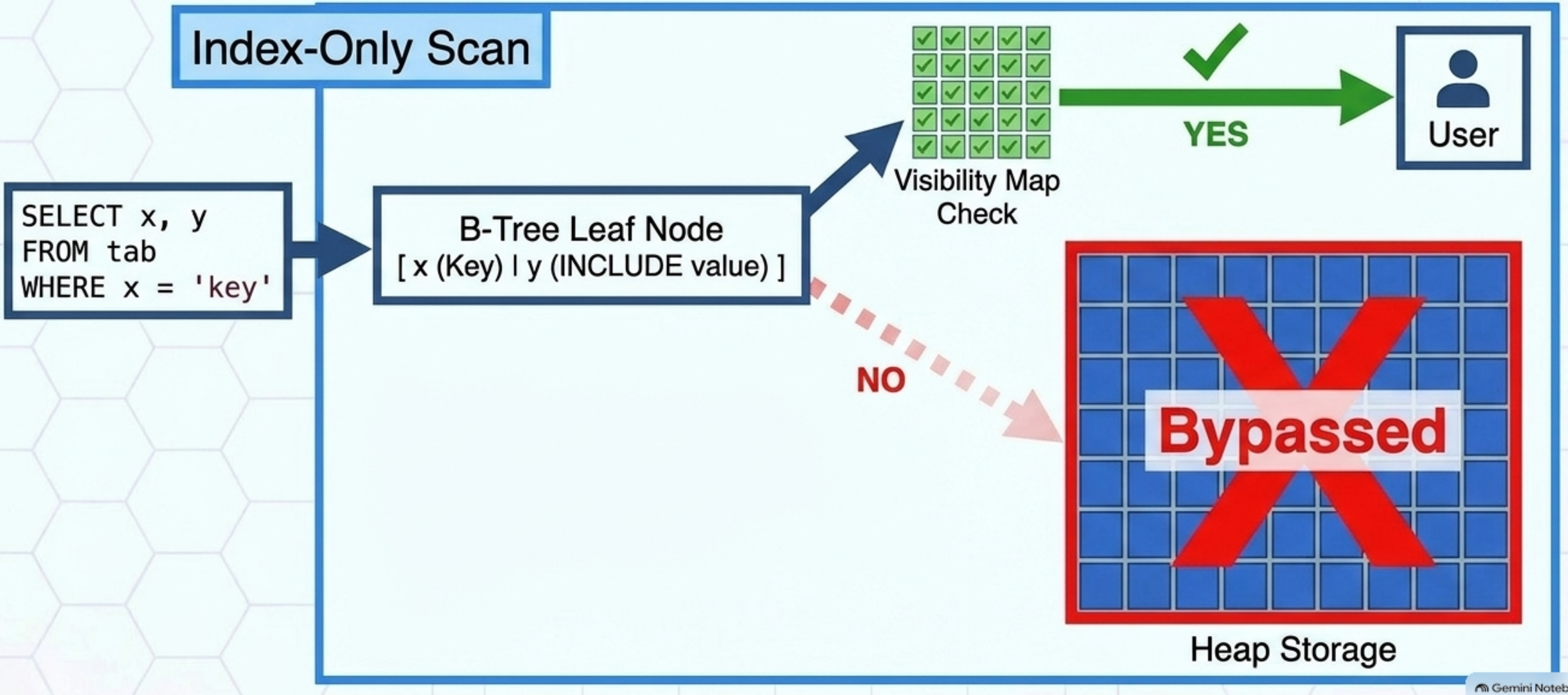
B-Trees excel at pinpoint lookups, but broad range scans on unclustered secondary indexes destroy I/O budgets.



Gemini Notebook

Covering Indexes and the Visibility Map

Saving the trip to the heap. PostgreSQL's INCLUDE clause allows the index to answer the query alone.

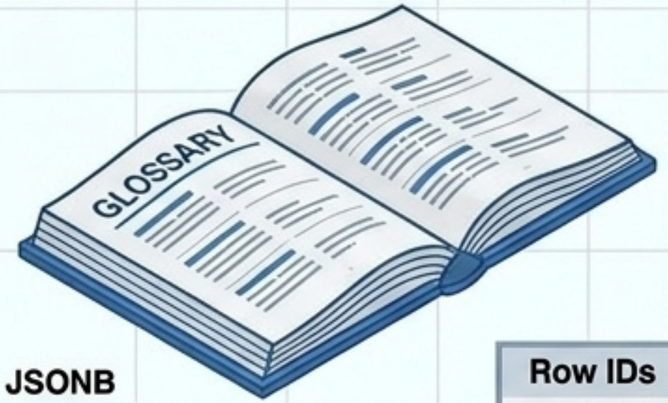


Gemini Notebook

Specialized OLTP Indexing: GIN, GiST, and BRIN

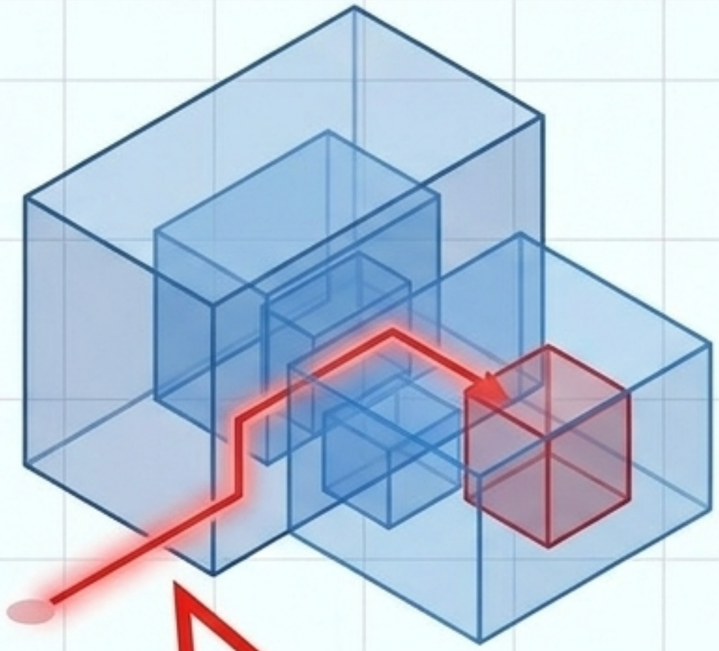
Matching the internal index data structure to the mathematical topology of the data.

GIN (Generalized Inverted Index)



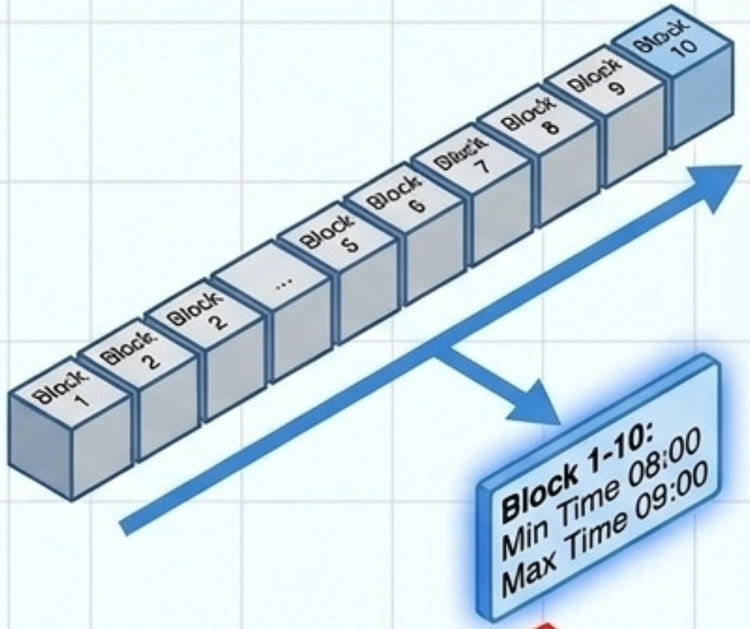
For jsonb / arrays

GiST (Generalized Search Tree)



For geo-spatial / nearest-neighbor

BRIN (Block Range Index)



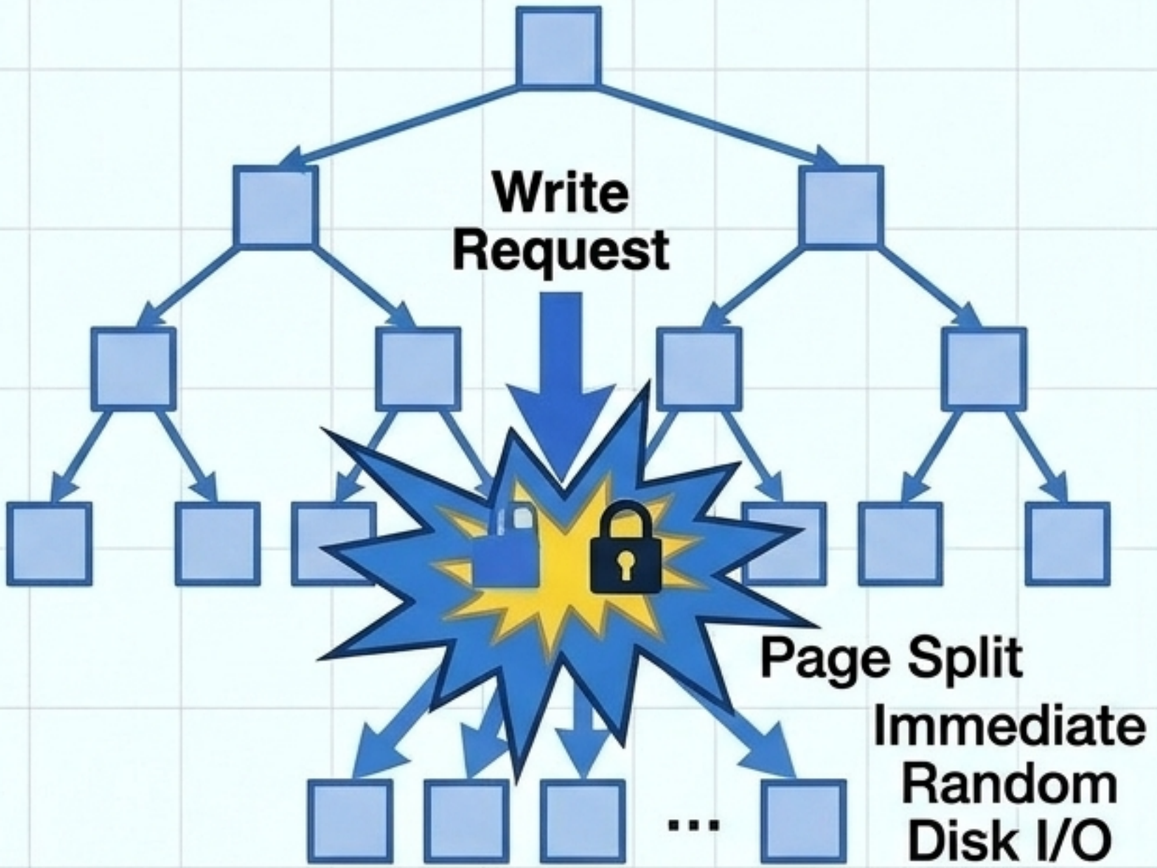
For correlated time-series

Gemini Notebook

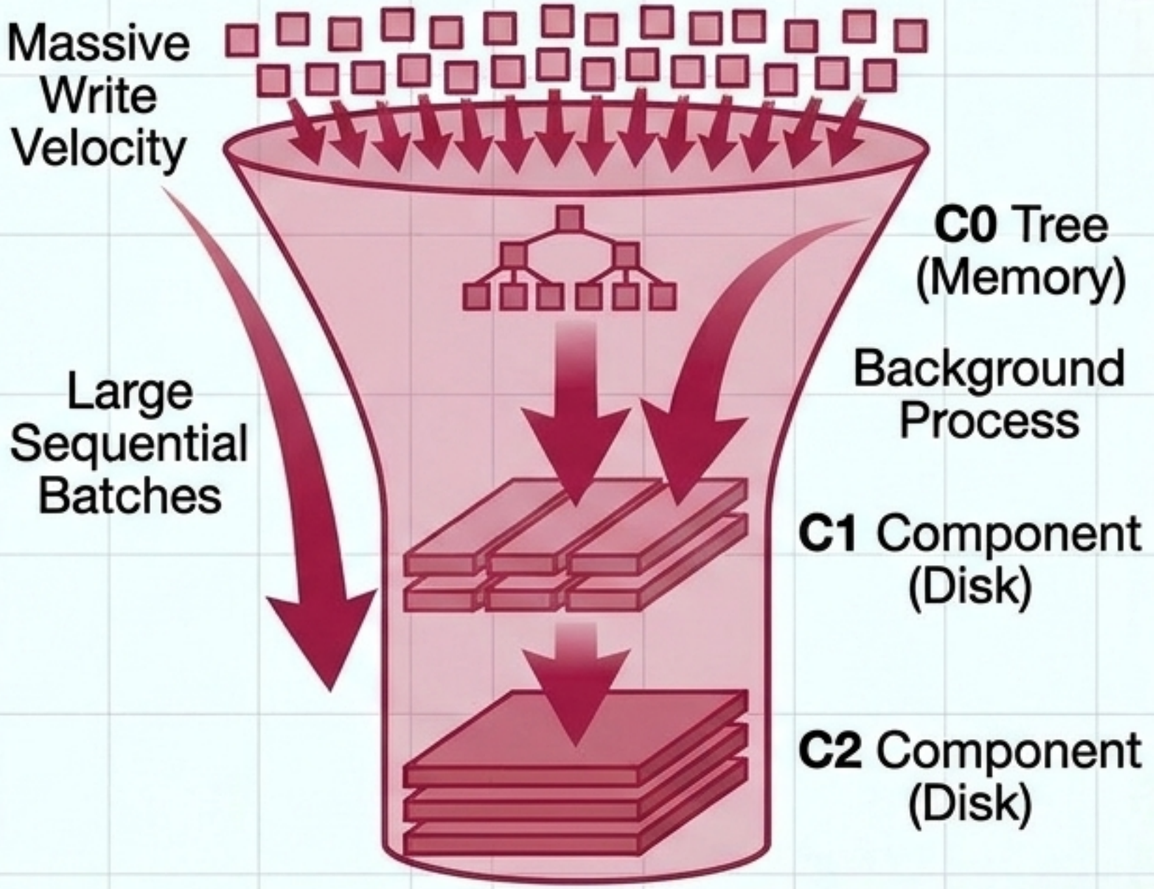
LSM-Trees vs B-Trees: The Write Optimization

Log-Structured Merge-trees absorb massive write velocity
by deferring and batching index changes.

In-Place Update



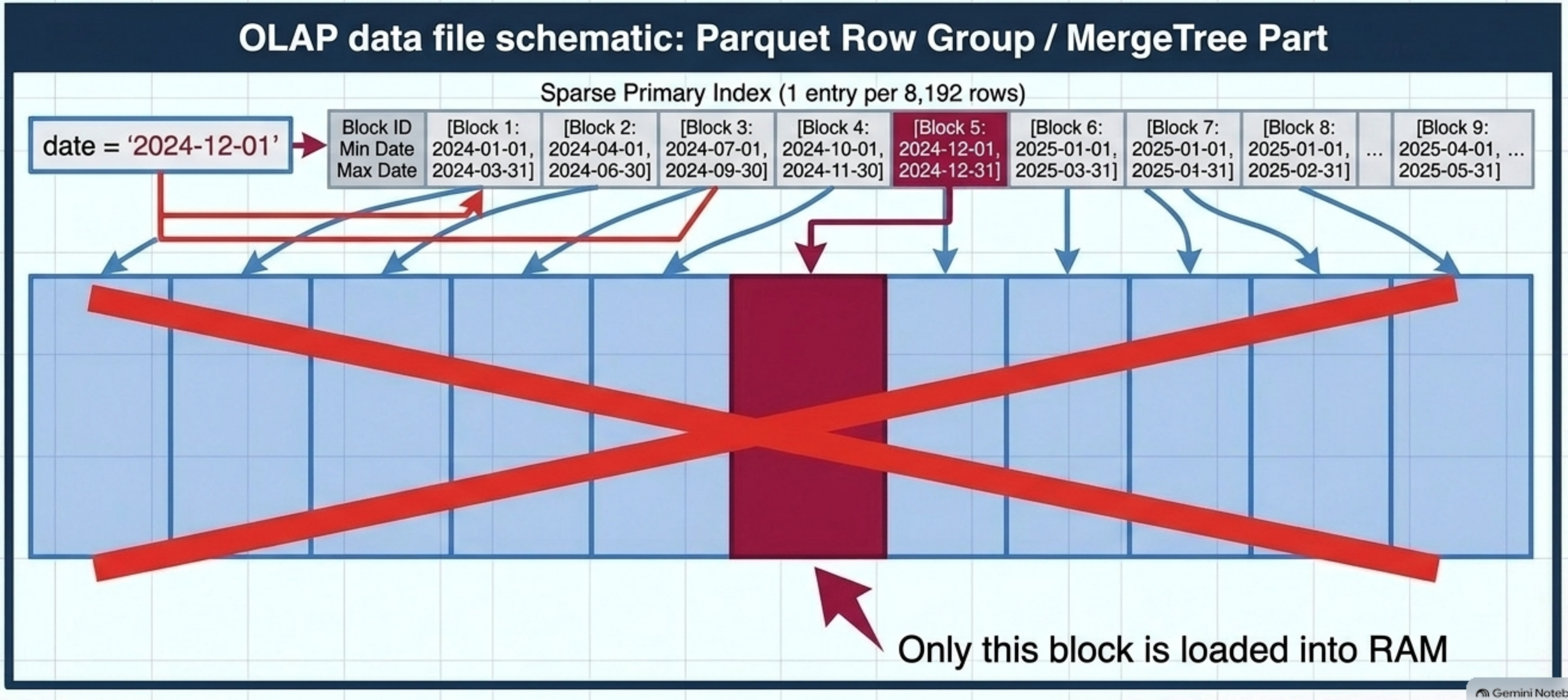
Deferred Cascade



Gemini Notebook

Analytical Pruning: ClickHouse & Parquet

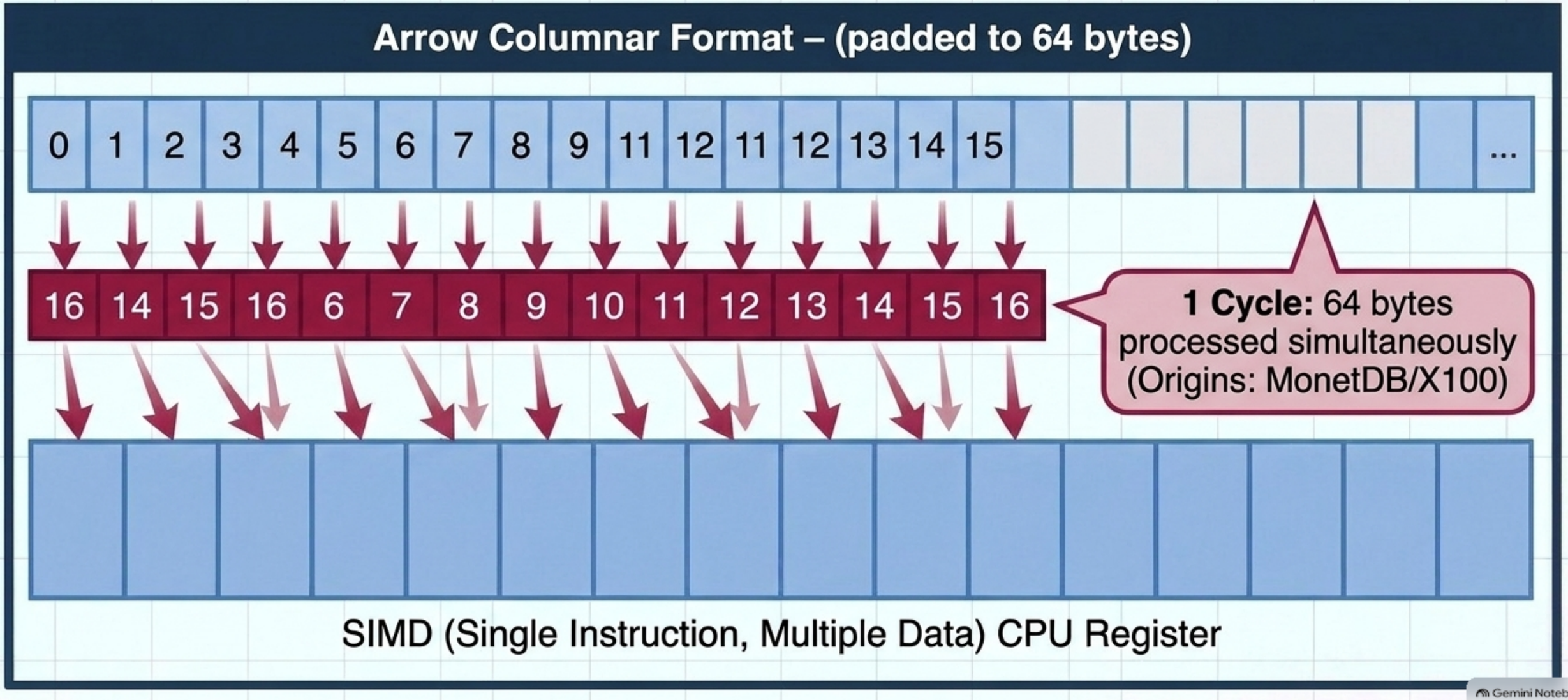
Analytics engines rely on smart sorting, block metadata,
and the brutal elimination of entire file chunks.



Gemini Notebook

In-Memory Execution: Arrow and SIMD

Interpreting rows one-by-one is dead. Modern analytical execution processes column vectors in tight, hardware-accelerated CPU loops.



Gemini Notebook

The OLTP I/O Budget: A PostgreSQL EXPLAIN

The mathematical **reality** of row-stores executing analytics: catastrophic I/O over-fetching.

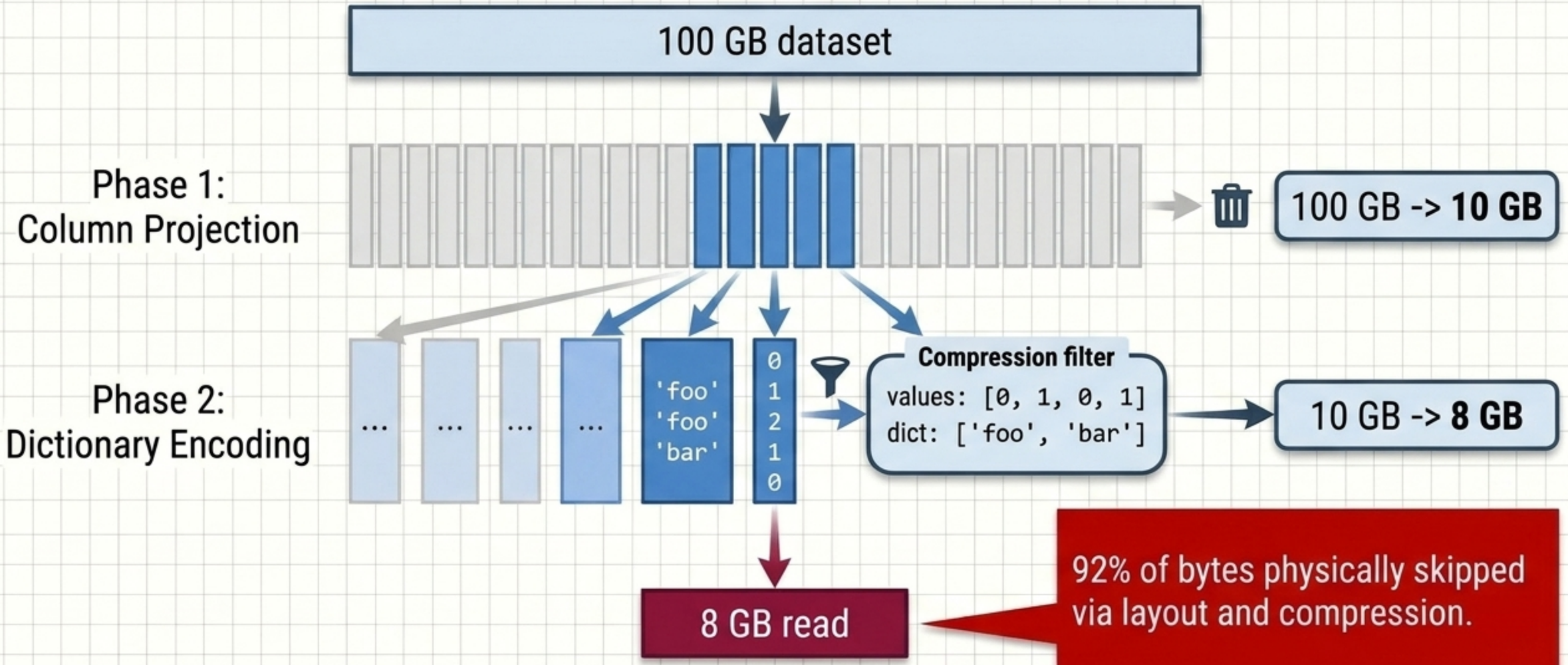
```
Seq Scan on measurements  (cost=0.00..445.00 rows=500000000 width=244)
  Buffers: shared hit=345 read=90000
```

To aggregate just 5 columns, the engine must pull the ENTIRE 100 GB table from disk into RAM. 95% of the bytes moved are instantly discarded.

Gemini Notebook

The OLAP I/O Budget: DuckDB and Parquet

Columnar formats mathematically slash the I/O budget before the CPU even wakes up.

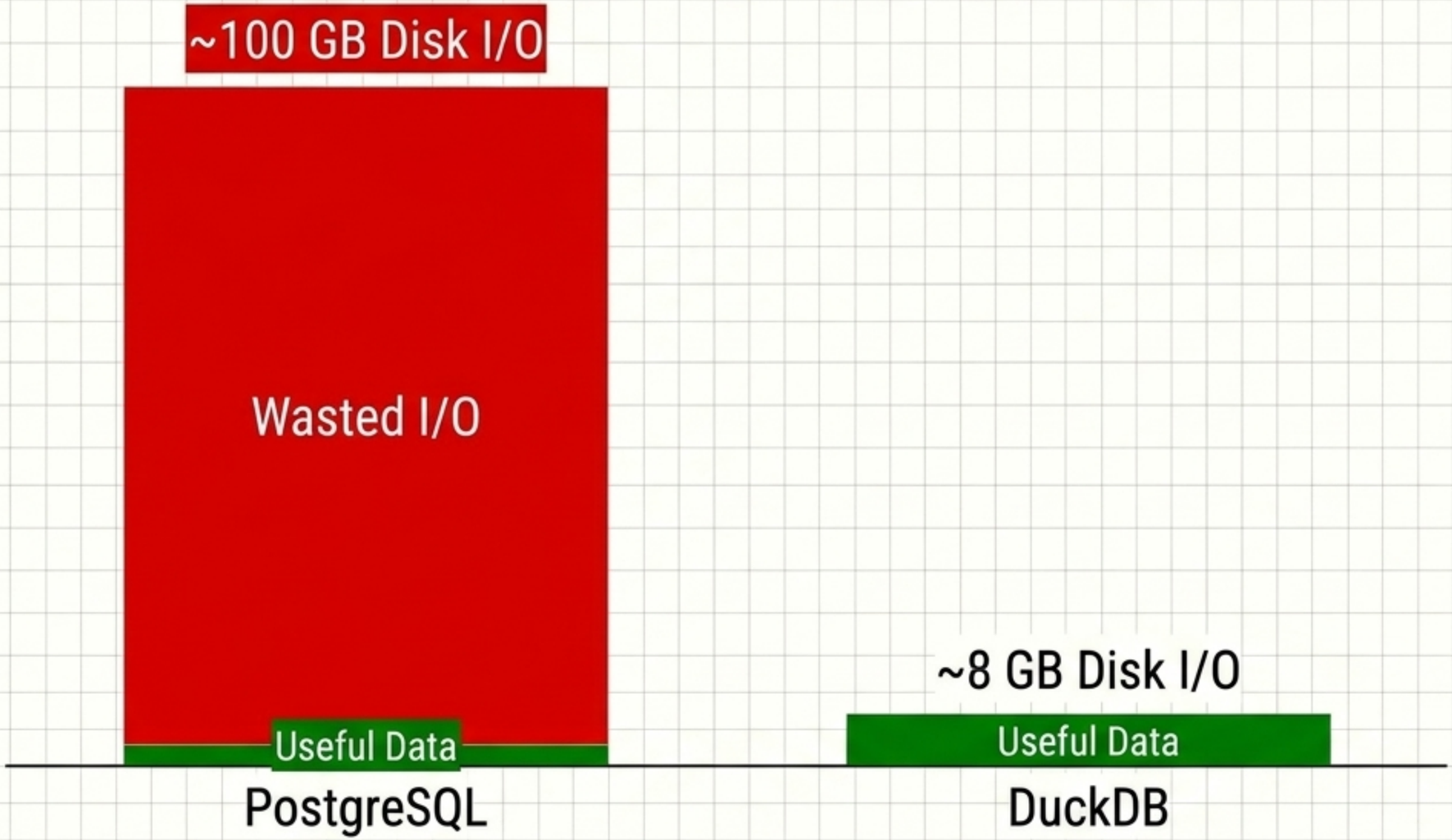


Gemini Notebook

You can tune Postgres memory buffers forever, but you cannot tune away the physical layout of the bytes.

Head-to-Head Math Synthesis

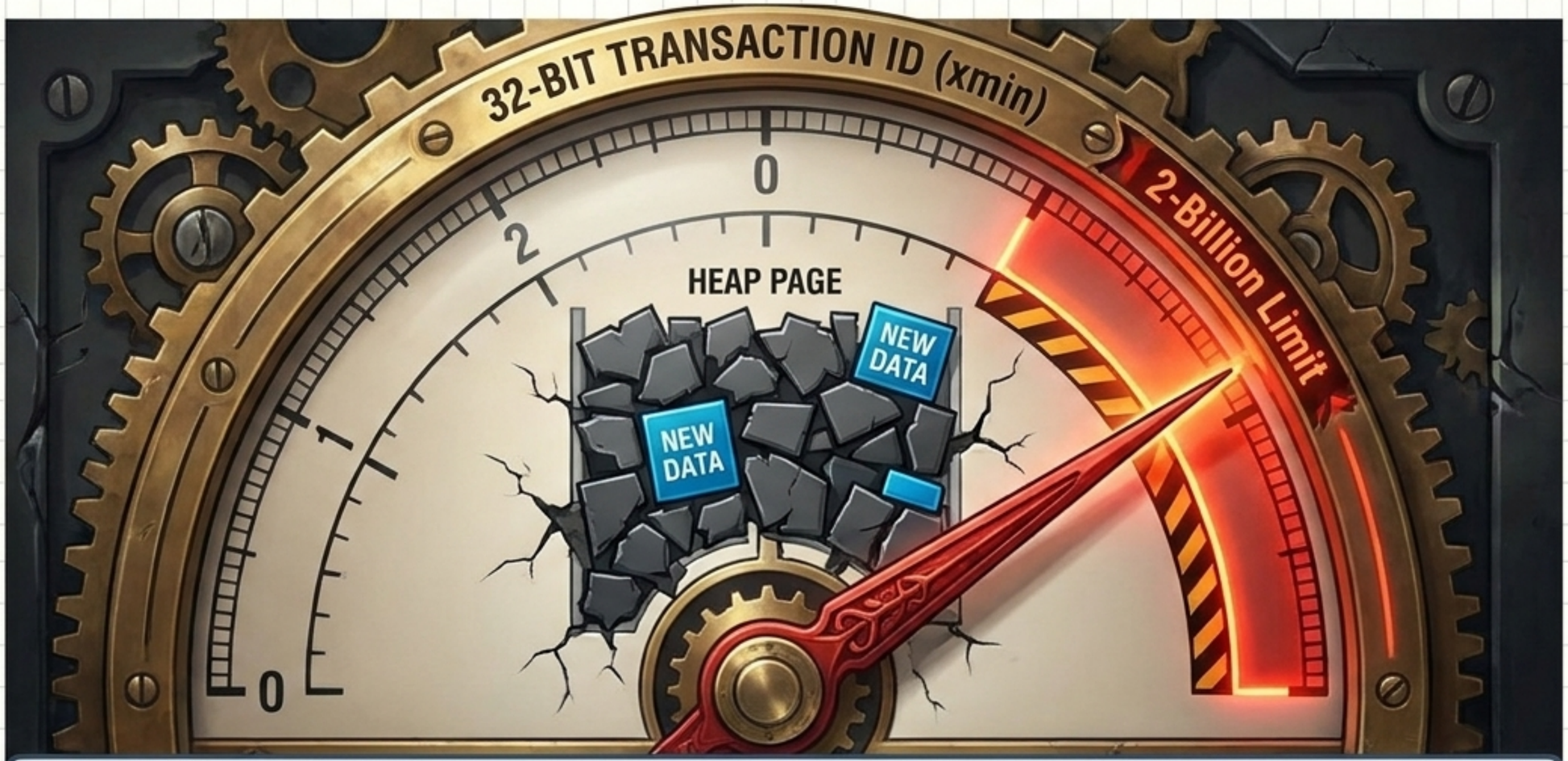
Why you do not use OLTP for wide aggregations. The raw disk math does not lie.



Gemini Notebook

Production Failure Mode: Transaction Wraparound

The catastrophic consequence of untuned **AUTOVACUUM** under heavy **UPDATE** churn.

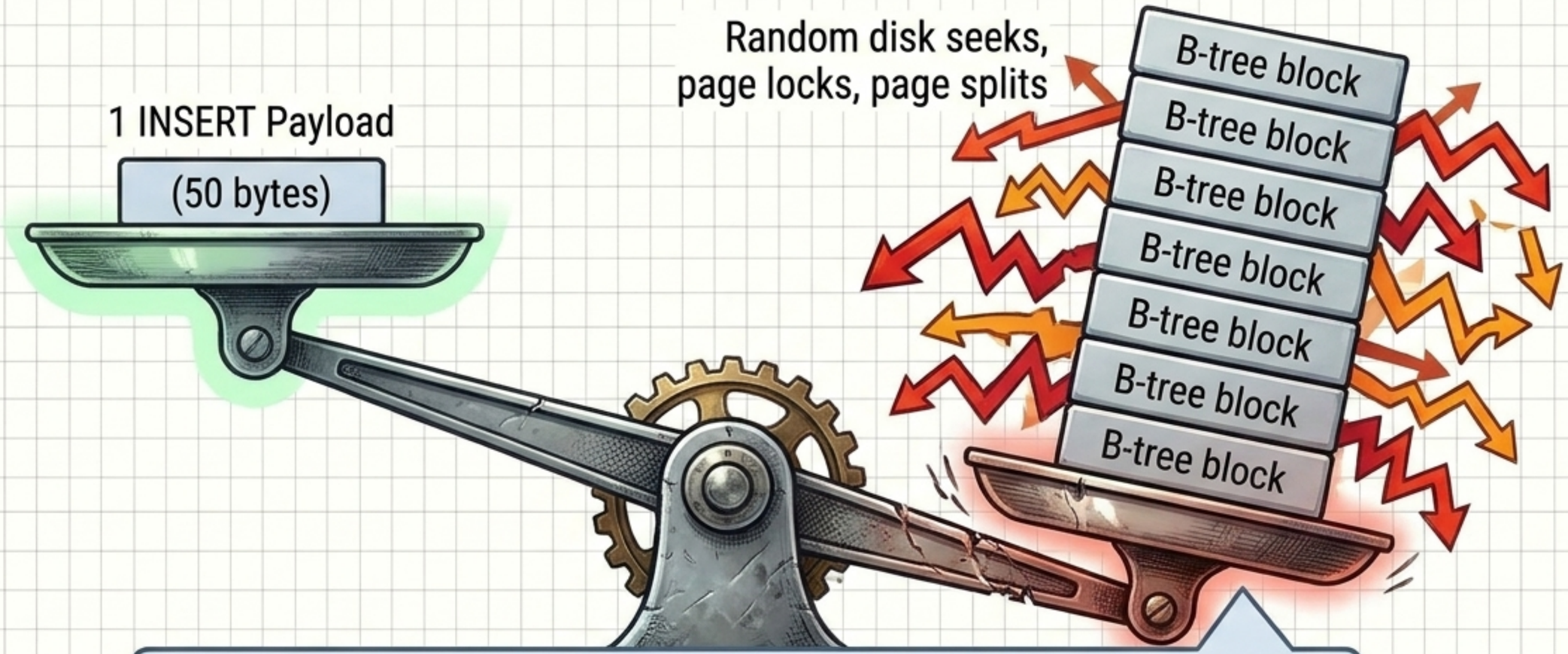



```
WARNING:  database must be vacuumed within 39985967 transactions
          FrozenTransactionId must be applied.
```

Gemini Notebook

Production Failure Mode: Over-Indexing Penalties

Every index you create is a permanent tax on every write you make.

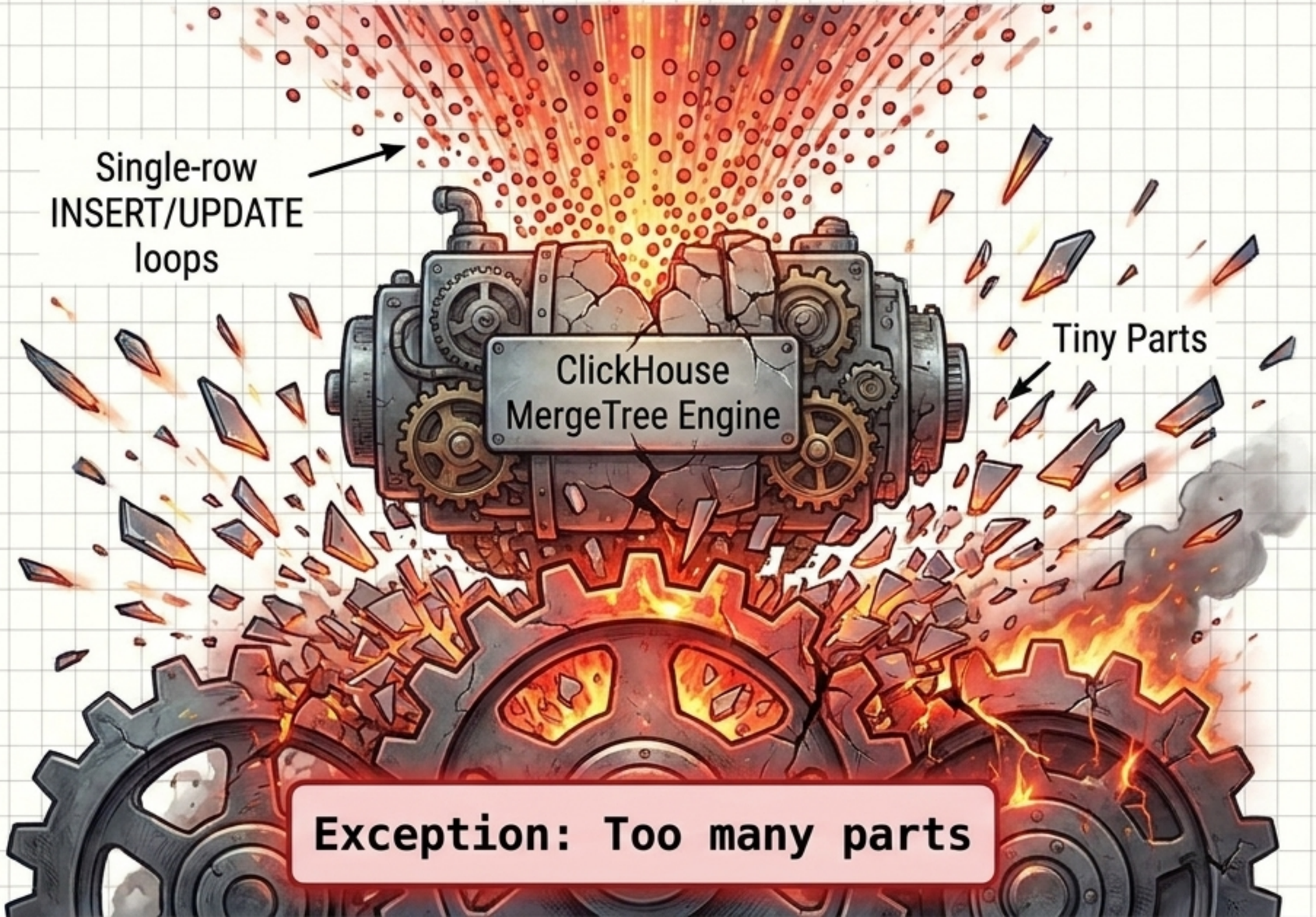


While GIN indexes use a fast-update pending list to batch writes, standard B-Trees force immediate write amplification. **Index only what you can mathematically prove you need.**

Gemini Notebook

Production Failure Mode: Unmerged Part Explosions

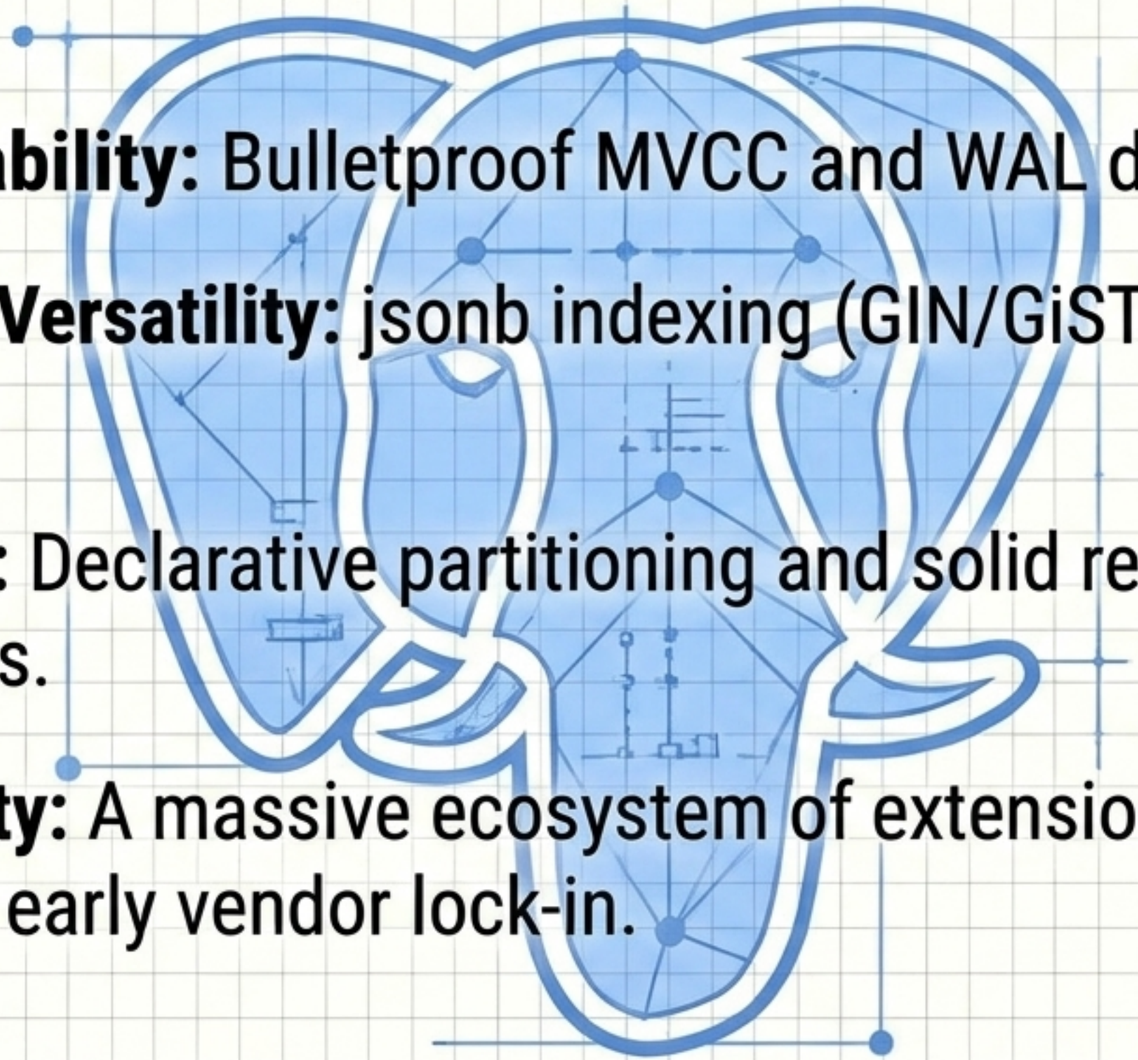
Firing row-by-row UPDATE loops into an OLAP engine will shatter it.



Gemini Notebook

Why Postgres Wins By Default

Until you hit the absolute limits of single-node hardware physics, PostgreSQL is the safest architectural bet in software engineering.

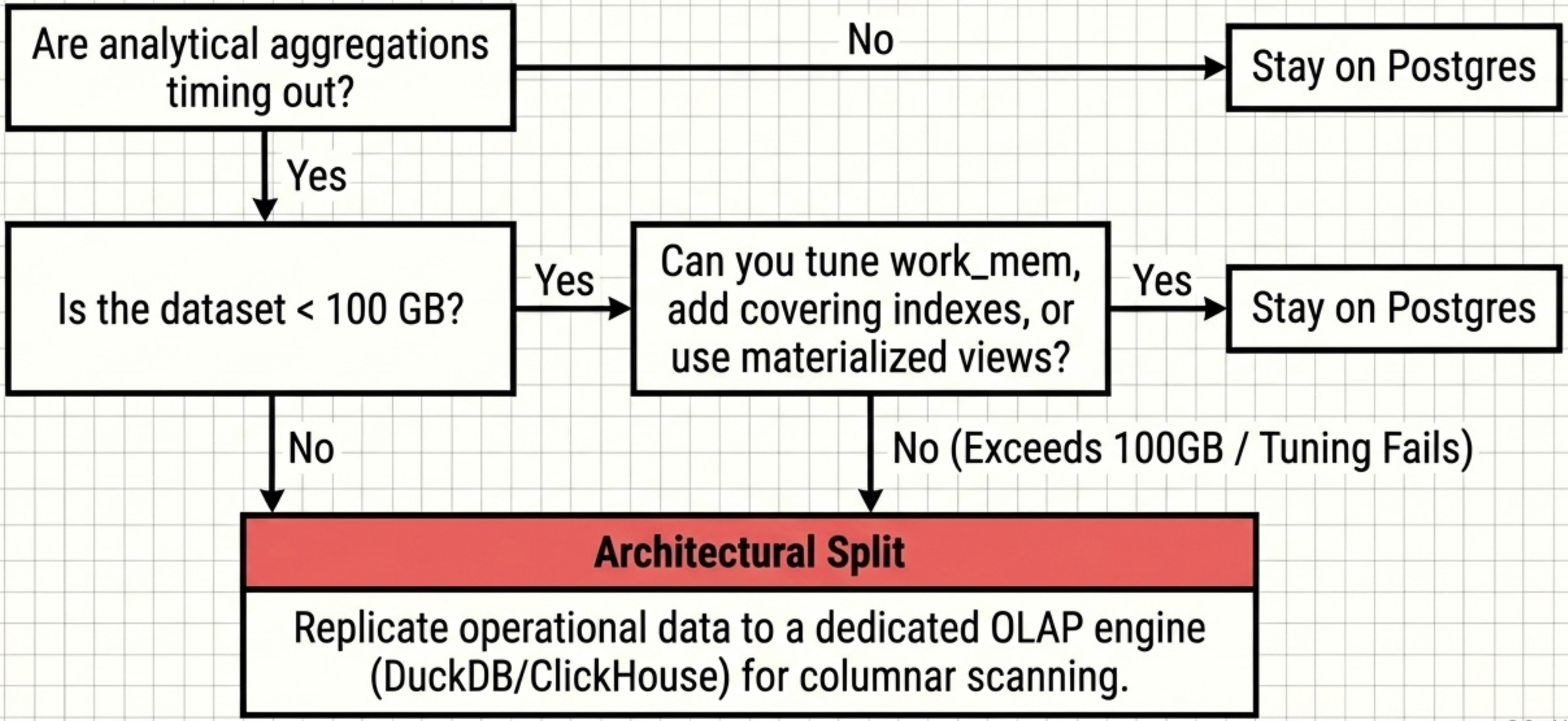


- ☒ **ACID Reliability:** Bulletproof MVCC and WAL durability.
- ☒ **Document Versatility:** jsonb indexing (GIN/GiST) absorbs NoSQL use-cases.
- ☒ **Scalability:** Declarative partitioning and solid read-replica ecosystems.
- ☒ **Extensibility:** A massive ecosystem of extensions (PostGIS, pgvector) preventing early vendor lock-in.

Gemini Notebook

The 100 GB Boundary: When to Offload

Do not introduce distributed analytical systems prematurely. Modern single-node Postgres can handle heavy workloads if tuned correctly.



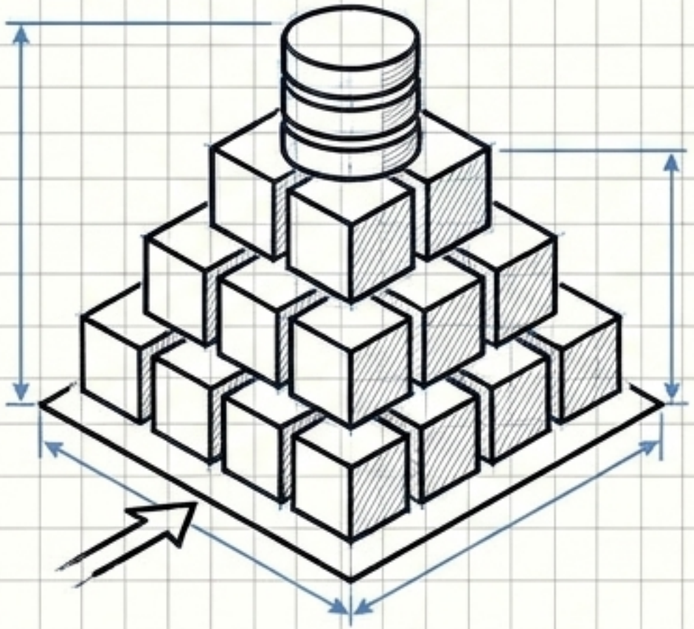
Gemini Notebook

Conclusion & Next Up

You now understand the physics of the engine. Next time, we scale the topology to infinity.

Data Warehouse

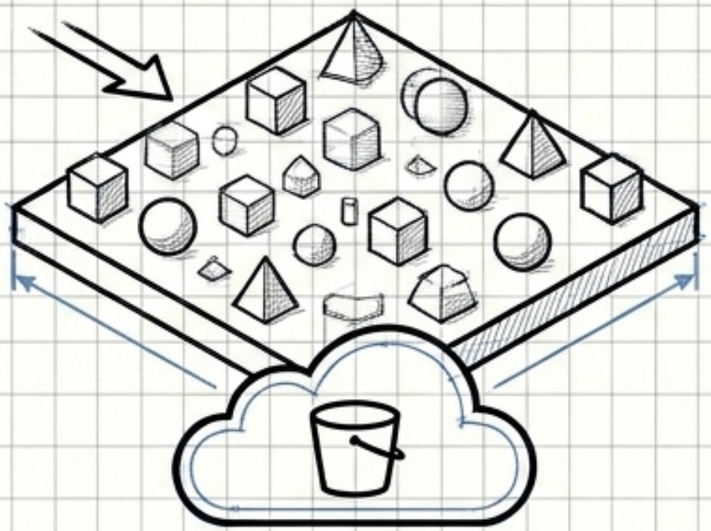
Snowflake/BigQuery



Governed, Structured Data

Data Lake

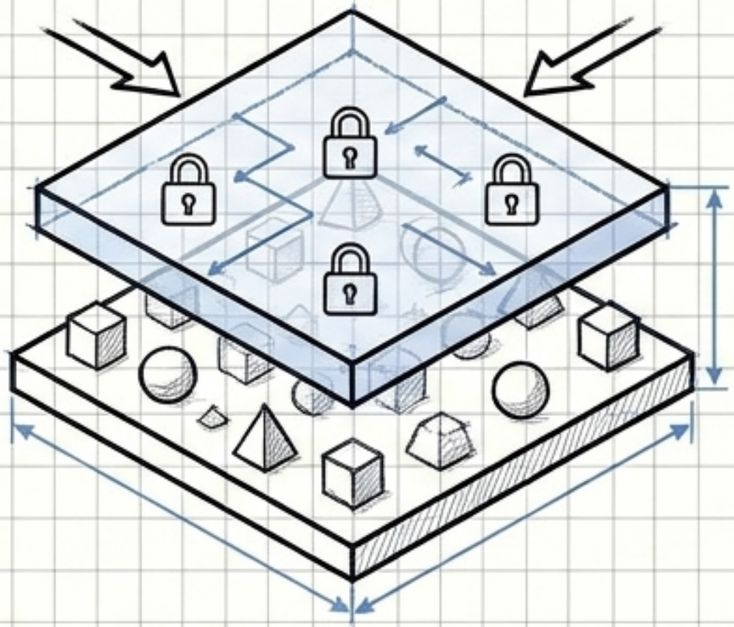
Parquet/Iceberg on S3



Raw, Unstructured Data

Lakehouse

Hybrid Structure



Structured Metadata over Lake Storage

S05E02: Warehouse vs. Lake vs. Lakehouse — The Distributed Topologies

Gemini Notebook