**CHAPTER 12 / 3 MINUTE BRIEF**

# Databases and Data-Bound Technologies

Read the full chapter → / Read this brief online →

## Plan the work as well as the space

I find it helpful to follow four questions through an investigation. What has to exist, including physical representations, copies, and retained history? What has to happen to it—reads and writes, maintenance, transfer, or recovery? Where can that work happen, given the nodes, partitions, volumes, and failure domains? Finally, when must it finish, and what service has to remain available while it does?

The questions resist being collapsed into one metric because an answer to one may tell us almost nothing about another. Knowing the retained footprint does not establish recovery time; a good aggregate throughput result may conceal the busiest partition. Free disk space cannot be spent as network throughput, however attractively a report combines their percentages.

Explore this section in the full chapter →

## Cassandra: the data model is part of the capacity plan

Cassandra's partition key determines how data is grouped into partitions. Rows sharing that key stay together on their replica nodes, so adding nodes does not subdivide the partition. The project's modeling guidance starts with the queries the application needs to support.[1] That makes the distribution of partition sizes and activity much more interesting to me than the result of dividing total bytes by node count.

Explore this section in the full chapter →

[1] Apache Cassandra documentation, "RDBMS design" and "Data Definition", checked October 3, 2026. Query-first modeling and partitions as undivided units of storage; the customer example is hypothetical.

## Kafka: retention is a promise to a future reader

A Kafka consumer tracks its position in a partition's log and can return to earlier positions to read retained records again. Consuming a record does not instruct Kafka to delete it. Under the delete policy, retention limits govern deletion, while log compaction preserves the latest value for a key rather than a complete event history.[2] Before estimating storage, I want to know which of those things the customer expects to have available when they return.

Explore this section in the full chapter →

## A deletion does not settle the storage question

PostgreSQL updates and deletes leave old row versions for later reclamation. Routine vacuuming makes that space reusable, and it also freezes old transaction IDs; if freezing falls far enough behind, PostgreSQL stops assigning new ones and accepts only read-only transactions until vacuum catches up. Ordinary `VACUUM` generally does not return space to the operating system. `VACUUM FULL` rewrites the table, so it needs room for a complete new copy and an exclusive lock that blocks all other use of the table until it finishes.[3]

Explore this section in the full chapter →

## Build a data-capacity decision record

We need a way to carry this work into the forecast without asking every reviewer to become an expert in the technology. The following record preserves the technical differences while exposing the obligations and choices that other teams need to understand.

[2]Apache Kafka 4.3 documentation, "Design", checked October 3, 2026; the downloads page still listed 4.3.1 (June 25, 2026) as the newest release when rechecked October 4, 2026. Replay of retained records; consumption versus deletion and key compaction.

[3]PostgreSQL 18 documentation, "Routine Vacuuming", checked October 3, 2026. Reuse versus release, the steady-state goal, `VACUUM FULL`'s copy and `ACCESS EXCLUSIVE` lock, and read-only operation near wraparound.

| Record | What the decision needs |
|---|---|
| Business obligation | Customer, useful outcome, retention purpose, latency requirement, recovery deadline, and who may change each commitment |
| Data footprint | Logical versus physical measurements, indexes and logs included, replica scope, growth and deletion assumptions, and placement distribution |
| Work and limits | Foreground demand, tested bottleneck, maintenance backlog and its deadline, effective transfer or replay rate, and competing work |
| Operating scenarios | Normal service, maintenance, expansion, a specified failure, recovery, and migration overlap; identify which coincide |
| Evidence boundary | Engine version, configuration, schema/query assumptions, measurement window, test conditions, and unresolved questions |
| Decision | Action, cost, resource location, lead time, accountable owner, review trigger, and the consequence if the assumption is wrong |

I would not turn the record into a universal utilization score. Cassandra repair, Kafka replay, and PostgreSQL vacuuming need different measurements; what we can share is a way to explain how their results affect a decision. The specialists retain their methods, and the business can see which commitments a proposed change preserves or puts at risk.

Explore this section in the full chapter →