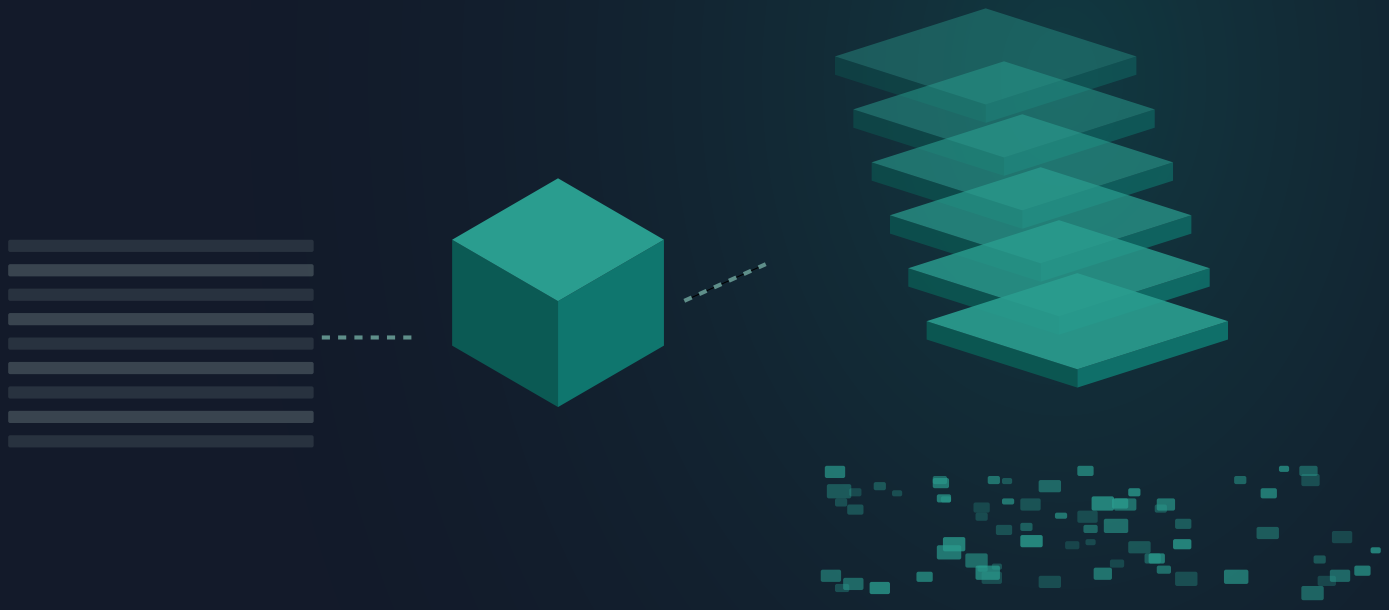


THE DATA SERIES, FROM TALLY TO TOKEN

From Warehouse to Lakehouse

Data on Every Platform

Analytical platforms have, step by step, separated what used to come in one box: storage, compute, table metadata, meaning and control. Files now move easily; the meaning of a number, its controls and the code around it do not. This book follows one board number, an insurer's quarterly loss ratio, from a 1987 printout to a 2026 lakehouse and an exit test, and shows a CTO how to judge a platform by one question: can an important result be reproduced elsewhere with its meaning and controls intact?

Juan Cardena

Enterprise AI & Agentic Systems Architect · blog.jcardena.com

CONTENTS

BEFORE YOU START

Reader's Brief: One Number, Six Questions

PROLOGUE

Two Numbers for One Question

PART I · THE QUESTION

01 The Workload and the Metric Contract

02 Why the Warehouse Existed

PART II · THE ENGINES

03 Cubes and the Frozen Answer

04 Parallel Engines and the Appliance

05 Files at Scale: Hadoop and the Lake

06 Columns on Disk: Parquet, ORC and Arrow

07 The Cloud Warehouse

08 Tables, Snapshots and the Lakehouse

PART III · THE PLATFORMS

09 Anatomy of a Platform

10 Catalog, Semantics and Policy

11 Security and Resilience on a Shared Platform

12 Seven Cards: Six Platforms and an Open Stack

PART IV · THE PRICE

13 What Work Costs

14 Location, Transfer and Jurisdiction

15 Migration and the Price of Leaving

PART V · THE WORK

16 Choosing per Workload

17 A Platform Ledger and Exit Test in 90 Days

CLOSING

Epilogue: The Format Outlives the Platform

APPENDICES

A Templates: Ledger, Metric Passport, Platform Card, Exit Inventory

B Timeline

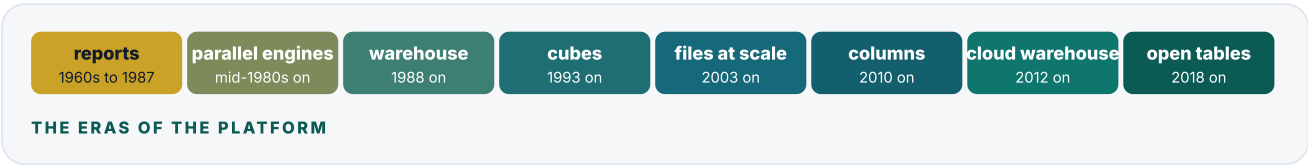
C Failure Library

D Glossary

E Sources

BEFORE YOU START

Reader's Brief: One Number, Six Questions



THE ERAS OF THE PLATFORM

This book follows one board number from a 1987 printout to a 2026 lakehouse and an exit test, and asks of every platform it passes through whether that number could be reproduced elsewhere with its meaning and controls intact. This brief tells you who is in the story, which questions the book asks, which three tools it adds to the Ledger, and where to start for your role.

Three companies, all illustrative

The companies are invented. The practices they use are real, and each chapter sources them.

- **The insurer:** the mutual fire insurer founded in 1898 from *From Tally to Token*, now writing property and casualty insurance in the United States. It owns the thread: its quarterly property loss ratio, reported to the board.
- **The health system:** regional hospitals and clinics under HIPAA, with medical images, FHIR records and a de-identified research copy.
- **The hotel and retail group:** about 60 hotels and 400 stores, with EU guests and a retail chain acquired on a second cloud. Its EU guest profiles get a second Ledger in Chapter Fourteen.

The six questions

Question	What a good answer names, on a platform
Where is it written?	Every copy: files, tables, snapshots, caches, shares, regions
What does it mean?	The definition, the code tables and the "as of", and which layer holds them
Who can change it?	People, service accounts and agents with write access, to the data and to the metric's SQL
What does it cost?	Store, move, query, keep and leave, with the people included
Who reads it now?	Dashboards, notebooks, models and assistants, and through which grants
What proves it is true and current?	Reconciliation, a recorded snapshot, and a rerun that returns the published figure

The metric passport

A **metric passport** is a box of eleven rows kept with one important number wherever it is published: metric id and version, numerator, denominator, population and exclusions, basis, effective date, knowledge cutoff, snapshot id or tag, transformation version, owner and approver, restatement status. Its core idea is that **every number has five times**: the date it describes, the date its data was cut off, when it was calculated, when it was approved and when it was published. Two times have rows of their own; calculation is written with the transformation version, approval with the owner and approver, publication with the restatement status. Chapter One fills in the passport for the insurer's Q4 2025 figure and finds three gaps: no snapshot, no SQL version, no restatement record.

The platform card

A **platform card** is one page of ten lines, the same lines in the same order for every platform: scope and date, storage of record, table formats, catalog and its interface, semantic layer, where policies are enforced, key custody, compute and SQL dialect, billing meters, and what travels when you leave (travels, travels with work, must be rebuilt). Chapter Nine explains the schema and Chapter Twelve prints seven cards, each dated November 2026. The book prints no vendor rankings and no vendor prices; prices live in a dated web companion.

The exit inventory

An **exit inventory** lists what one important number depends on, from files to grants, code and people, and whether each travels, travels with work or must be rebuilt. Chapter Fifteen builds it; Appendix A holds the template.

Reading paths by role

Role	Read first	The decision you leave with
CTO	Prologue, Chapters 1, 9, 12, 15, 16	Which platform and format per workload, and what an exit would cost
Chief data officer	Chapters 1, 8, 10, 17	Which numbers get a passport and a tagged snapshot, and which system holds definitions and grants
Data or AI architect	Chapters 6, 8, 9, 10, 11, Appendix A	Where meaning, policy and keys live, and what must be rebuilt elsewhere
CFO	Prologue, Chapters 1, 8, 13, 17	Which board figures can be reproduced on demand, and what each costs to keep that way

ERA CARD

Reader. You, with a number your board trusts and no record of which data it came from.

Security. Whoever can edit a metric's SQL or filter can change that number without touching a row.

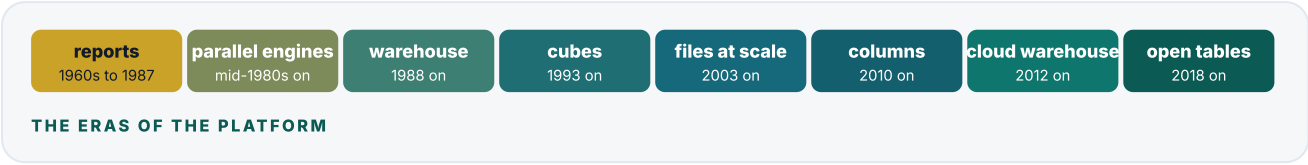
Cost. A passport takes about an hour per number to write (illustrative); Chapter Seventeen budgets the rest.

AI. An assistant quotes the number it finds; the passport is how it can find the right one.

Monday rule: pick one number your board reads every quarter and ask the six questions about it before you read further.

PROLOGUE

Two Numbers for One Question



THE ERAS OF THE PLATFORM

On Thursday 19 February 2026 the board of a mutual fire insurer met to review the fourth quarter of 2025. Page six of the board pack showed the quarterly property loss ratio: 61.8 percent, from the cloud warehouse dashboard. Twenty minutes into the item, a director asked about a different figure. On Tuesday the CFO's analyst had read the same quarter from the company's new lakehouse, in a notebook, and the notebook said 64.1 percent. Both numbers went up on the screen side by side. Each was labelled "Q4 2025 property loss ratio". Neither said when it had been computed or from which data. Nobody in the room could say, in the meeting, which one the board should rely on.

Two numbers on one screen

The insurer is invented, the mutual fire insurer founded in 1898 whose perpetual policy is the thread of *From Tally to Token*. By 2026 it writes property and casualty insurance in the United States. The meeting closed a bad week for its data: on 12 February its AI service agent told a building owner that a perpetual policy had lapsed, corrected the next morning, as *From Tally to Token* tells; on Monday 16 February an investigation began, which is covered in depth later in this series.

The **loss ratio** is losses incurred, paid or still owed, divided by premiums earned. Chapter One defines both from the statutory rules. A few points move the year's result, so a gap of 2.3 points in one quarter is not rounding: on this insurer's quarterly premium, each point is millions of dollars of losses.

The CFO said the pack figure was the approved one. The appointed actuary, who signs the statutory Statement of Actuarial Opinion and, under this insurer's close procedure, approves the board figure, confirmed approving it on 5 February; the pack matched. The analyst said the notebook read the same claims and premium tables through the same lakehouse catalog, and that a rerun that morning had returned 64.1 again. The head of data platform said both were right about their own query. The chair asked which number was the fourth quarter. There was no answer that fitted on a slide.

Here is what nobody could see on the screen. The pack figure was 61.8 percent (approved 5 February 2026, data to 23 January): computed by the warehouse close job on 26 January from claims and premium data received by Friday 23 January, approved by the appointed actuary and countersigned by the CFO on 5 February, published in the pack on 10 February. The notebook figure was 64.1 percent (latest snapshot, 13 February, read 17 February): computed when the analyst pressed run, from the quarterly loss table as last committed on Friday 13 February.

The board did not pick one. It asked management for a written reconciliation of the two figures, every tenth of a point explained, before the next meeting.

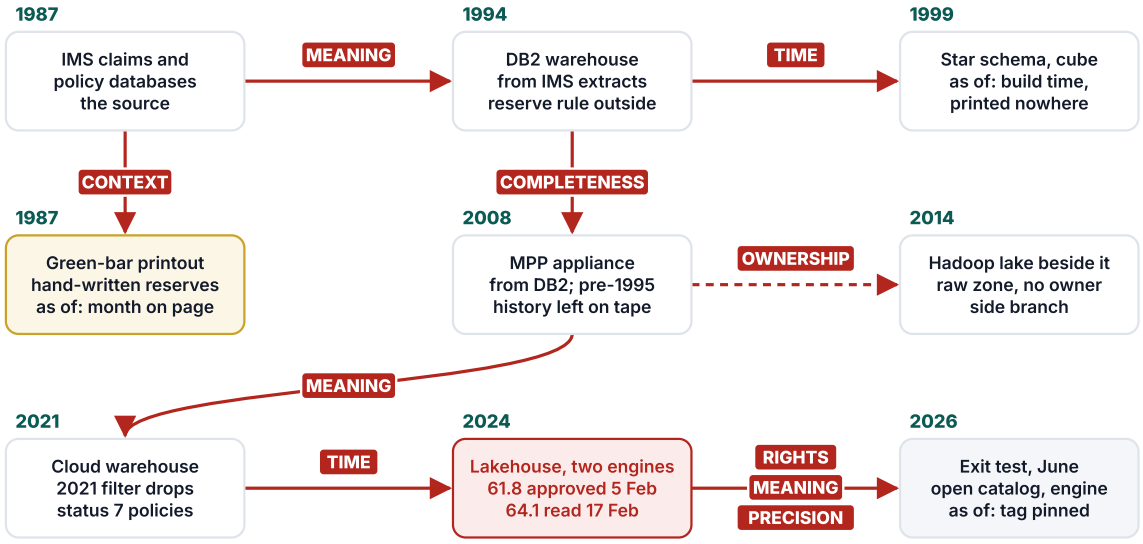
Neither number was a typo

It is tempting to treat this as an error to find and fix. Each could be rerun that week with the same result, the notebook's until the next Friday commit, though neither was tied to a recorded version of its data or its SQL. Both read real claims and real premium. They differ because they answer slightly different questions, and the screen did not say which question each one answered. Some of the gap comes from *when* each number knew what it knew. Some comes from *what* each one counts, and from a rule written into a query years ago by people who have since moved on.

Chapter Eight reconciles the two numbers, every tenth of the gap (2.3 points as printed, the difference of two rounded figures; 2.25 between the unrounded ones), beside the actuaries' own record of how the quarter's losses developed, with the unrounded parts printed so you can add them yourself. Chapter Fifteen copies the same tables to a second region, runs the board query on an open engine and gets a third number. Chapter Seventeen shows which figure the board's own written contract asks for, and closes the question. To see why the reconciliation was hard, start where the figure started.

FIGURE P.1

One Number, Since 1987: the hops of the board's loss ratio, and where its "as of" was kept



Red chips: loss codes (Appendix A). 61.8: data to 23 Jan, approved 5 Feb. 64.1: snapshot of 13 Feb, read 17 Feb. Neither kept its as of.

Hops 0 to 7 of the quarterly property loss ratio, each red chip a Ledger loss code: IMS fed both the printout and the warehouse, the warehouse fed the cube and the appliance, and hop 4 is a side branch (dashed), since no loss-ratio input came from the lake. The "as of" was on the 1987 page, hidden in the 1999 cube, lost at read time in the 2024 lakehouse and pinned again by a tag in the 2026 exit test.

Author's diagram based on the book's running example; the insurer, numbers and people are illustrative.

Walking back from the screen

A **hop** is any move or change of a dataset between people, systems or formats. Figure P.1 shows the hops of the board figure, numbered 0 to 7, with hop 4 on a side branch. Read it backwards, from the lakehouse to the printout.

2024, hop 6. The claims and premium tables were rewritten as Iceberg tables on object storage under a vendor-managed catalog, read by two engines: a SQL warehouse engine, which runs the close job and writes the dashboard's close table, and a notebook engine. Every write commits a **snapshot**, a version of the table. Claim events load nightly; the quarterly loss table both engines read is committed every Friday. No snapshot id, tag or "as of" was kept with the board figure, and nothing protected the quarter-close snapshot from the weekly job that expires old snapshots.

2021, hop 5. The appliance and the lake's curated tables moved to a cloud data warehouse (no loss-ratio input came from the lake), in the same programme that moved the policy domain (*From Tally to Token* tells that half). The loss-ratio SQL was rewritten in a new dialect. The STATUS_CODES table did not travel, so nobody knew that status 7 meant perpetual and in force, and a rewritten filter read "premium due = 0" as "not a live policy" and dropped those policies.

2014, hop 4. A Hadoop data lake was built beside the appliance for claim files: adjuster notes, photos, third-party reports. They landed in a raw zone with no accountable owner, and each team read them its own way. This hop is a side branch: the board figure itself did not pass through the lake.

2008, hop 3. The DB2 warehouse moved to a parallel warehouse appliance in one rack. History before 1995 was not migrated. It sits on tape in a load format no current job can read.

1999, hop 2. A claims star schema and a multidimensional cube were built for the board pack, rebuilt after each quarter close. The cube's build time was its only "as of", and it was printed nowhere. Corrections made later reached the warehouse but not the frozen cube.

1994, hop 1. Monthly extract programs copied claims and policy data from IMS into an enterprise data warehouse on DB2. "Incurred loss" was defined once, in the warehouse. The reserve adjustment rule stayed in the actuaries' spreadsheet, so the warehouse figure still needed a manual step.

1987, hop 0. A COBOL report program read the IMS claims database and the policy master after each month end and printed the monthly loss report.

Ahead lies hop 7, the June 2026 exit test: not a plan to leave, but a test of whether the number can be reproduced elsewhere with its meaning intact.

The green-bar printout

The 1987 report came off the line printer on continuous green-bar paper, 132 columns wide, totals by line of business, and went to the actuaries' room. The actuaries did not take its loss figures as printed. They knew which large claims were under-reserved and which reserve had been set before a court ruling, and they wrote their adjustments in pencil beside the printed lines. The marked copy went into a binder. The figure the controller used was the printed figure plus the pencil, and the two differed.

That is a loss too, and Figure P.1 marks it: the context lived in the binder, not in any system. But look at what the sheet kept. It named the month it described. The printed number and the used number sat on one page, and the difference between them was written beside it.

Anyone who asked why the board's figure was not the machine's figure could open the binder and see.

Thirty-nine years later the insurer had two numbers again, the machine's and the approved one, on a far better platform. This time the difference was written nowhere near the number.

> *The 1987 page knew which month it described. The 2026 screen knew more about claims and less about itself.*

What this book gives you

The book's claim is that analytical platforms have, step by step, pulled apart storage, compute, table metadata, meaning and control. Files became easier to move; the meaning of a number, starting with its "as of", did not. So judge a platform by one test: can an important result be reproduced elsewhere with its meaning and controls intact?

The tools are four: the **Data Lineage Ledger** from *From Tally to Token* (one row per hop, with what each lost), the **metric passport** (a number's definition and five times), the **platform card** (what a platform holds and what travels when you leave) and the **exit inventory** (what leaving costs). Every number in this book carries its time.

ERA CARD

Reader. A board reading two numbers, and behind it an actuary, a CFO, an analyst with a notebook and an assistant over the semantic layer.

Security. The view that filters the board figure can be changed by anyone with rights to its SQL; nobody reviewed that right as a control over the number.

Cost. The reconciliation the board asked for took the actuarial and data teams days of work; recording a snapshot and a definition at the close would have taken minutes (both illustrative).

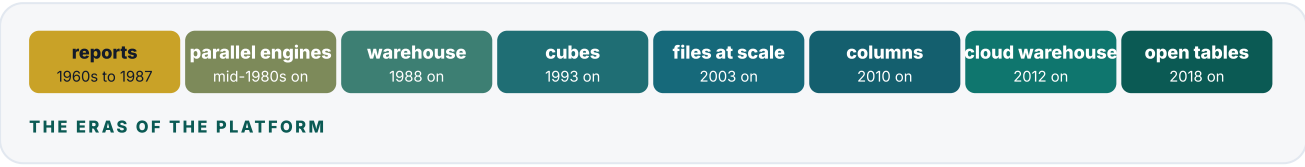
AI. An assistant asked for the Q4 loss ratio would have quoted whichever of the two numbers it reached first.

Monday rule: take the last board pack and, for each number on it, ask whether anyone can say from which data and as of which date it was computed.

PART I · THE QUESTION

CHAPTER ONE

The Workload and the Metric Contract



THE ERAS OF THE PLATFORM

In 1934 the Proceedings of the Casualty Actuarial Society printed a paper by John R. Lange on the loss reserve schedules in the annual statements of casualty insurers. One of its tables follows a company's liability business through Schedule P, policy year by policy year, under a heading that reads "developed loss ratio at end of" each later year. The same policy year can carry a different loss ratio at each year end, and the table prints each one under the year end at which it was measured. Actuaries have written the "as of" of a loss ratio into the number itself for that long. Most analytics platforms do not. The decision this chapter changes comes before any platform is chosen: **write down the workload, the definition of the number and its five times, and only then ask which platform can keep them**.

Workloads before platforms

A platform is bought for work, and the work comes in kinds. A **workload class** is a group of jobs with the same shape: how much data each one reads, how many run at once, how fresh the data must be and what happens when a result is late or wrong. Each class needs a **service objective**. Google's site reliability engineering book defines a service level objective as "a target value or range of values for a service level that is measured by" a service level indicator, which it calls "a carefully defined quantitative measure of some aspect of the level of service". The same discipline fits data: name the measure, set the target, write both down.

Workload class	What matters most	An objective worth writing down
Board and regulatory reporting	The same answer every time it is asked, with its times	Rerun at the recorded snapshot returns the published figure
Interactive BI	Response time for an analyst who is exploring	Most queries answer within seconds on curated tables
Ad hoc analysis	Freedom to join anything, at a cost someone sees	Cost per query visible to the person who ran it
High-concurrency dashboards	Hundreds of readers at once at month end	Peak readers served without queueing the close job
Large transformations	Finishing inside the batch window	Close tables ready by a stated hour
Streaming ingest	Freshness and order of events	Events queryable within minutes; late events handled by rule
Data science	Wide reads of history, repeatable training sets	Every training set tied to a snapshot
Sharing	Another party reads without a copy you lose track of	Every share listed with its recipient and purpose
Retention	Keeping what the law and the business need, and no more	Retention and deletion rule per dataset, tested yearly

The first row is this book's subject. A board figure is a small query: a few tables, a few million rows. What makes it hard is not speed. It is that the number must mean the same thing next quarter, on the next platform and in the hands of the next reader, and that someone must be able to say which data it was computed from. That is a service objective too, and it is usually the one nobody wrote down.

The classes also pull a platform in different directions. Dashboards want many small, fast reads; transformations want large scans in a window; data science wants wide history; sharing wants another party inside your controls. A company often ends up with more than one engine, sometimes more than one platform, reading the same tables. That is reasonable. What it requires is one authoritative definition of each number, with every engine's implementation tested against it, and that each engine records which version of the data it read. Chapter Sixteen turns these classes into a decision table; this chapter fixes what any choice must preserve.

The metric contract

A **metric contract** is the written definition of one number: what is counted, over which population, on which accounting basis, rounded how, owned by whom. The insurer's board figure is a **loss ratio**, and the words in it come from statutory accounting, the rules state insurance regulators apply to insurers' financial statements.

- **Loss ratio.** The National Association of Insurance Commissioners (NAIC) glossary defines it as "the percentage of incurred losses to earned premiums."
- **Incurred losses.** In the same glossary, "sustained losses, paid or not, during a specified time period", typically found from the losses paid in the period plus the unpaid losses at its end, minus the unpaid losses carried in from the period before.
- **Unpaid losses.** SSAP No. 55, the NAIC statement on unpaid claims and losses, names the parts: reported losses (expected payments on claims reported but not paid, held as **case reserves**) and **IBNR**, losses "incurred but not reported", for insured events that "have occurred but have not been reported" as of the statement date.
- **Net premiums earned.** The glossary's words: premiums "that will not have to be returned to the policyholder if the policy is cancelled". Net means after reinsurance; the contract below says which business counts.

Two independent choices sit inside those words, and a contract has to make each one. The first is **loss adjustment expense** (LAE), the cost of investigating and settling claims. SSAP No. 55 lists it as its own kind of future cost, separate from losses, and the glossary's loss ratio does not mention it. Yet the word is used both ways. In the NAIC's property and casualty industry snapshot for the year ended 31 December 2025, the line labelled "Loss Ratio", 66.5 percent, equals net losses incurred plus loss expenses incurred, divided by net premiums earned. Two analysts can both say "loss ratio" and differ by the whole expense line. The second choice is **anticipated salvage and subrogation**: what the insurer expects to recover by selling damaged property or by claiming from a third party who caused the loss. SSAP No. 55 makes this the insurer's choice: "If a reporting entity chooses to anticipate salvage and subrogation recoverables", it estimates them and deducts them from the liability for unpaid losses. Netting is permitted, not required.

The insurer's audit committee approved its contract in 2019. In plain words: calendar quarter; losses incurred (paid losses plus the change in case reserves plus the change in IBNR), net of reinsurance, excluding loss adjustment expense and net of anticipated salvage and subrogation; divided by net premiums earned; built from statutory accounting amounts as a management metric; the insurer's direct business, net of ceded reinsurance, with business assumed from its affiliate under the pooling agreement excluded and reported separately; status 7 policies, perpetual with the deposit held, in force and included; percent

to one decimal, rounded half to even; measured as booked at the board's knowledge cutoff for the quarter. Every clause answers a question two honest analysts could answer differently. The contract is a page. Most companies have the page somewhere. Few keep it beside the number.

Take the last clause, which looks like a formality. **Rounding half to even** means that a value exactly halfway between two tenths goes to the one whose last digit is even: 61.85 becomes 61.8, while 61.95 becomes 62.0. Rounding half up sends both upward. The Q4 2025 board figure is exactly 61.850 before rounding, so the rule decides whether the pack says 61.8 or 61.9. Two engines that compute the same exact value can print different board figures if one of them was never told the rule, or if one computes in binary floating point: there 61.85 is stored as a value a hair above the half, so even a half-to-even rule prints 61.9. The contract should require decimal arithmetic as well as the rounding rule. The insurer's close job printed 61.8 only because its engine's rounding function rounds decimals half to even: the rule lived in the engine's dialect, not in the metric, and Chapter Fifteen shows what that cost. A related habit belongs in the contract too: when a figure is built from parts, add the unrounded parts and round once at the end. Rounded parts can add up to a total that differs from the rounded total by a tenth, and a tenth is enough to start a reconciliation.

Every number has five times

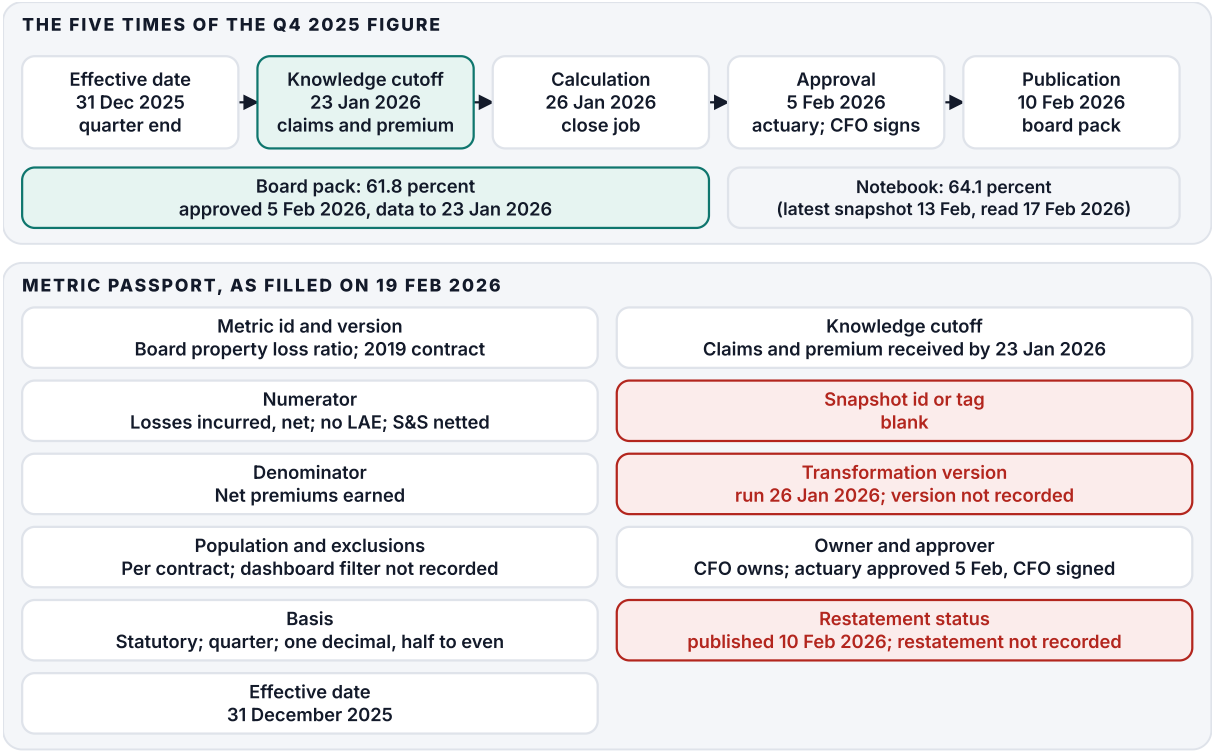
A definition says what a number counts. It does not say when. The Q4 2025 board figure, 61.8 percent (approved 5 February 2026, data to 23 January), carries five dates, and each answers a different question.

Time	Q4 2025 board figure	The question it answers
Effective date	31 December 2025, quarter end	Which period is measured?
Knowledge cutoff	Claims and premium data received by Friday 23 January 2026	Which facts were known when it was computed?
Calculation	Monday 26 January 2026, warehouse close job	When did the system compute it?
Approval	Thursday 5 February 2026, appointed actuary, countersigned by the CFO	When did accountable people accept it?
Publication	Tuesday 10 February 2026, board pack	When did readers receive it?

The effective date and the knowledge cutoff do the most work. Take a storm loss on 28 December, reported to the insurer in February. The Q4 figure computed from data to 23 January has no claim row for it; the loss is covered only by the IBNR reserve held at the close for losses not yet reported. A Q4 figure read later, from February data, sees a case reserve set for that claim. The first has the contract's time: calendar quarter, as booked at the knowledge cutoff. The second is a re-evaluation of Q4 with later knowledge, which statutory accounting books as a change in estimate in the period the change becomes known; the board figure is a management metric with its own cutoff, written into the contract. Both can be reproduced; only the first has the time the contract asks for. That does not make the approved 61.8 the contract figure: it used the contract's time, definition and rounding, but a 2021 population filter, and Chapter Seventeen gives the figure the contract asks for. And a read that adds case reserves set after the close to the IBNR held at the close, without releasing the IBNR they replace, counts the same loss twice. A number shown without its knowledge cutoff hides which one it is.

FIGURE 1.1

The metric passport holds all five times; on 19 February three of its rows had gaps



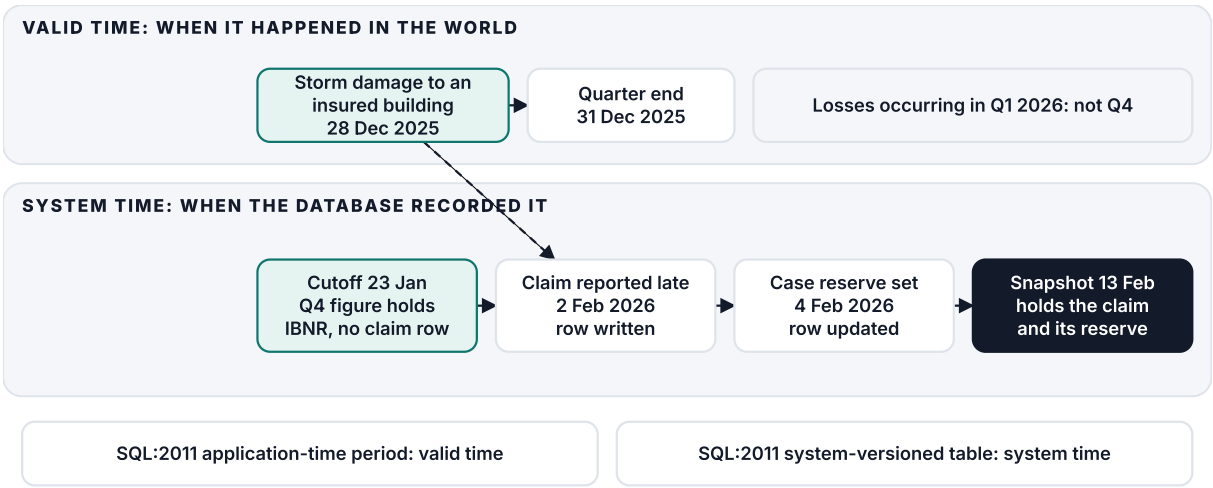
The five times of the Q4 2025 board figure above, its eleven-row passport below: effective date and knowledge cutoff have rows of their own, the calculation run sits in the transformation version row, approval in the owner and approver row, publication in the restatement status row; the red rows had gaps where the platform and the close process recorded nothing.

Author's diagram based on the book's running example; the insurer and its numbers are illustrative.

Database research has names for this. Krishna Kulkarni and Jan-Eike Michels of IBM, describing the temporal features of the SQL standard, write that the literature "recognizes two dimensions of time": **valid time**, "the time period during which a row is regarded as correctly reflecting reality by the user of the database", and **transaction time**, "the time period during which a row is committed to or recorded in the database". Their example is an insurance policy entered long before it takes effect. They cite Richard Snodgrass's book *Developing Time-Oriented Database Applications in SQL* (Morgan Kaufmann, 1999) for the two dimensions. ISO and IEC published SQL:2011 in December 2011, and it gives both a home: tables with an **application-time period** carry valid time, and **system-versioned tables** carry transaction time, which the standard calls system time. A query can ask for a table `FOR SYSTEM_TIME AS OF` a timestamp and get the rows the database held at that moment. A table with both periods is usually called **bitemporal**.

FIGURE 1.2

Valid time and system time: one loss belongs to Q4, but the database learns of it in February



The loss happened in the quarter (valid time) and was recorded after the knowledge cutoff (system time), so a Q4 figure read from the 13 February snapshot knows more than the one computed from data to 23 January; the dashed line follows the same loss, reported late.

Author's diagram based on Kulkarni and Michels, "Temporal features in SQL:2011", SIGMOD Record 41(3), 2012; the claim and its dates are illustrative.

In those terms, the effective date corresponds roughly to valid time (which period the figure describes) and the knowledge cutoff to system time (what the data held when the figure was computed). The match is close, not exact: valid time belongs to each row, not to a quarter, and the cutoff is when data reached the insurer, which a database may record a little later. A table format with snapshots (Chapter Eight) records when each table version was created, a table-level form of system time that lasts as long as the snapshots are kept, and an Iceberg tag can give one snapshot a name and a retention of its own. The facility exists. What a team has to add is the habit of recording, beside the board figure, which snapshot it came from.

The actuaries' "as of"

Property and casualty insurers have printed the knowledge time of their loss figures for a long time, because their regulators ask for it. Schedule P of the statutory annual statement shows losses by the year in which they were incurred, and the NAIC instructions state that "Schedule P displays ten years of historical data for all lines of business." That has been true only since the 2024 annual statement; before it, seven lines, the property lines among them, showed two years. Its history parts show incurred losses and reserves as of 31 December of

each year in the column heading, so the table is a **loss development triangle**: each row is one accident year, measured again at every later year end. An **evaluation date**, the date as of which losses are valued, is the nearest actuarial term for a knowledge cutoff.

> **WHAT HASN'T CHANGED**
>
> Lange's 1934 paper tested a company's Schedule P liability reserves "as of December 31, 1930" against what was later paid. The test needed two times on every figure: the year the losses belonged to and the year end at which they were estimated. A board figure in 2026 needs the same two, plus three more.

The insurer's actuaries never stopped working this way. Their reserving models and Schedule P carry evaluation dates. What lost the date was the path from their figures to the board. Each platform hop, from the 1987 printout to the 2024 lakehouse, carried the value and dropped some of its times, as the prologue's figure shows. The board figure is the most watched number in the company and the one with the least recorded history of when it was known.

A metric passport with three gaps

The tool for keeping a number's meaning and times together is the **metric passport**: one box per important number, eleven rows, kept with the number wherever it is published. Here is the passport for the Q4 2025 figure as the insurer could fill it on the day of the board meeting, Thursday 19 February 2026.

METRIC PASSPORT

Metric id and version	Quarterly property loss ratio, board figure; contract approved by the audit committee in 2019
Numerator	Losses incurred, net of reinsurance, excluding loss adjustment expense, net of anticipated salvage and subrogation
Denominator	Net premiums earned
Population and exclusions	As written in the contract: direct business net of ceded reinsurance; assumed pooling business excluded; status 7 policies included. Which filter the dashboard applies: not recorded
Basis	Management metric built from statutory amounts; calendar quarter; percent to one decimal, rounded half to even
Effective date	31 December 2025
Knowledge cutoff	Claims and premium data received by Friday 23 January 2026
Snapshot id or tag	
Transformation version	Run: warehouse close job, 26 January 2026. Version: not recorded
Owner and approver	Owner: chief financial officer. Approved 5 February 2026 by the appointed actuary, countersigned by the CFO
Restatement status	Published 10 February 2026, board pack. Restatement: not recorded

Eight rows came from documents: the contract, the close calendar, the approval memo. One of them is thinner than it looks: the population row says what the contract asks, not what the dashboard's filter does. The other three rows hold what the platform and the close process should have recorded, and the parts that matter are blank. The five times all have a place: effective date and knowledge cutoff in their own rows, the calculation run in the transformation version row, approval in the owner and approver row, publication in the

restatement status row. No snapshot id or tag ties 61.8 to the data it was computed from. No version names the SQL that computed it, rewritten in 2021 in a new dialect. Nobody can say whether the figure has ever been restated, because nothing records what "the figure" was; restating is a decision for finance and the actuaries, but only a recorded snapshot and version can show what was restated. The 64.1 percent on the analyst's screen (latest snapshot, 13 February, read 17 February) has no passport at all. Chapter Eight fills in the blanks for both numbers, and Chapter Seventeen makes the passport routine.

> **ERA CARD**
>
> **Reader.** The board, through a pack; behind it the actuaries, the CFO and, by 2026, an assistant over the semantic layer.
>
> **Security.** Whoever can edit the metric's SQL or a filter on its view can change the board figure without changing a single row of data.
>
> **Cost.** A passport takes about an hour per metric to write (illustrative); the missing rows cost a reconciliation each time two numbers meet.
>
> **AI.** An assistant reads the definition it finds; if the contract is not stored with the metric, it answers from whichever table it reaches.

Monday rule: pick the five numbers your board reads every quarter and, for each, write the metric contract and the five times on one page before anyone discusses platforms.

CHAPTER TWO

Why the Warehouse Existed



In February 1971 two researchers at MIT's Sloan School of Management, G. Anthony Gorry and Michael S. Scott Morton, circulated a working paper called "A Framework for Management Information Systems". Its sharpest line was about the systems companies already had. What had been called management information systems, they wrote, "has had almost nothing to do with real managers or information but has been largely routine data processing". Sixteen years later the insurer was a fair example. Its claims and policies lived in IMS databases built to pay claims and send bills, and the actuaries who set the reserves read a printed report once a month, then corrected it by hand. The warehouse was the industry's answer to that gap: a second copy of the data, built for the people who ask questions rather than the programs that write records. The decision this chapter changes is one many teams do not make on purpose: **where the definition of a board number lives, and who may change it**.

Reports for programs, questions for people

Gorry and Scott Morton sorted management decisions on two axes. One was the level of the decision: operational control, management control, strategic planning. The other was how well the decision was understood. A **structured** decision follows a known routine, such as reordering stock when it falls below a level. An **unstructured** decision has no routine: what reserve to hold for a storm whose claims are still arriving, or whether to keep writing fire policies in a county. They named systems for the first kind "structured decision systems" and systems for the second kind "Management Decision Systems", and they placed the greatest potential of interactive terminal systems in the area between, the semi-structured one.

The distinction matters for this book because it explains why analytics became a separate workload. A claims system is built for the structured side: record a claim, set a case reserve, issue a payment, all in small, fast, correct steps. A board loss ratio is an analytical question asked of records built for operations. Once its definition is written, computing it is structured work; the reserve judgment inside it is not. It totals a quarter's losses and the

premium earned by the policies in force, and it needs history, not just the current row. In the 1970s and 1980s that question was answered the way most questions were answered on a mainframe: a programmer wrote a batch report, it ran at night, and the answer arrived on paper.

The insurer's version of that report is where the thread of this book starts. In 1987 the monthly loss report ran after month end, a COBOL program reading the IMS claims database and the IMS policy master and printing totals by line of business on 132-column green-bar paper. The actuaries read it in their room. They did not take its totals as final: they wrote reserve adjustments by hand on the printout, and the adjusted pages went into a binder. The figure the controller reported upward was the adjusted one, and the quarter's ratio was built from three such monthly reports. So the printed figure and the used figure differed, and the difference lived in pencil, in one binder, read by the few people who knew where it was. The Ledger records this as a **CONTEXT** loss: the value travelled, the knowledge that corrected it stayed in the room.

THE HOP · 0 · 1987 · IMS CLAIMS DATABASE AND THE IMS POLICY MASTER TO MONTHLY LOSS REPORT, PRINTED ON GREEN-BAR PAPER BY A COBOL REPORT PROGRAM

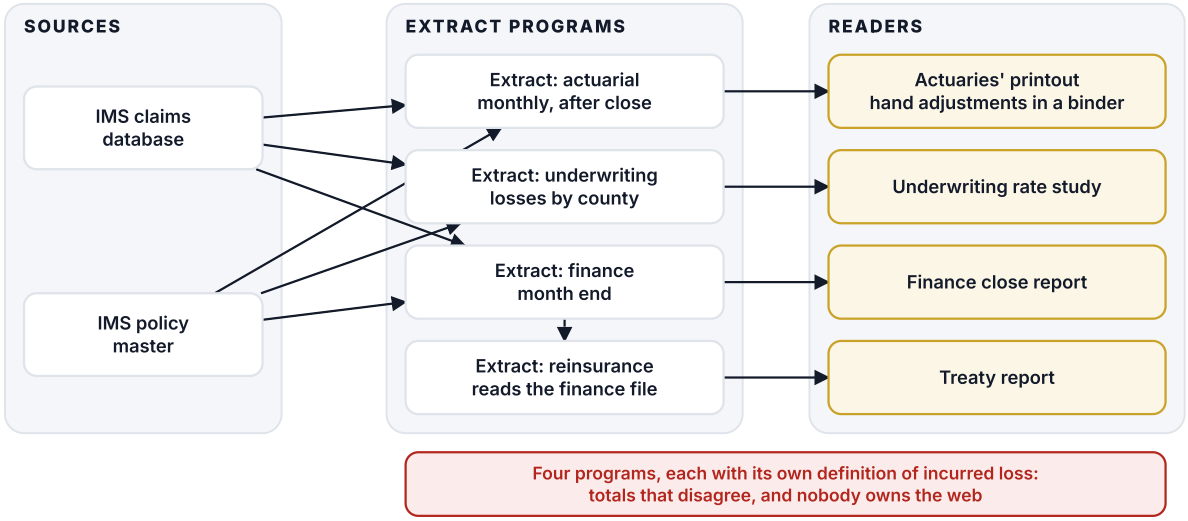
Record	Property loss ratio, Q4 board figure (dataset), hop 0, 1987
From	IMS claims database and the IMS policy master
To	Monthly loss report, printed on green-bar paper by a COBOL report program
Mechanism and schedule	Monthly batch job after month end; the report is read in the actuaries' room
Format change; what was lost or assumed	CONTEXT Fixed-width print lines, 132 columns; totals by line of business. **Reserve adjustments written by hand on the printout and kept in the actuaries' binder; the printed figure and the used figure differ**
Accountable owner	Controller
Who can change it	Claims systems programmers (the report program); actuaries (the hand adjustments)
Classification; retention or erasure duty	Financial record; kept with the statutory working papers
Read by (people, apps, models, agents)	Actuaries, the controller, the board (a typed summary)
Cost per month; basis	Machine time on the shared mainframe plus about two actuary days a month for the hand adjustments (illustrative)

The spider web of extracts

A single report is manageable. The trouble began when every department wanted its own. Underwriting wanted losses by county to price policies. Finance wanted losses by month to close the books. Reinsurance wanted losses by treaty. Each request became another **extract program**: a batch job that read the operational database, selected and summarized some records, and wrote a file for one reader. Each extract ran on its own schedule, applied its own filters and used its own idea of what "incurred loss" meant. Some read the claims database directly; some read another department's extract, because it was already summarized and cheaper to run.

FIGURE 2.1

Before the warehouse: one extract program per question, each with its own definition



Extract programs read the operational databases, and sometimes each other's files, to serve one reader each; by the late 1980s the insurer's loss totals came from several of them.

Author's diagram of a typical extract environment; the insurer and its programs are illustrative.

Figure 2.1 shows the shape that results: lines crossing from a few sources to many readers, with copies of copies in between. Each line is a small, reasonable decision. Together they produce three problems a CTO still meets today. First, the totals disagree, and nobody can say which one is right, because the definitions are buried in program code. Second, the operational systems carry the load of every extract, and the night batch window fills up. Third, nobody owns the whole picture: each extract has an author, but the web has none. Replace "extract program" with "pipeline" or "notebook" and the figure describes many cloud data estates in 2026.

A second store, built for questions

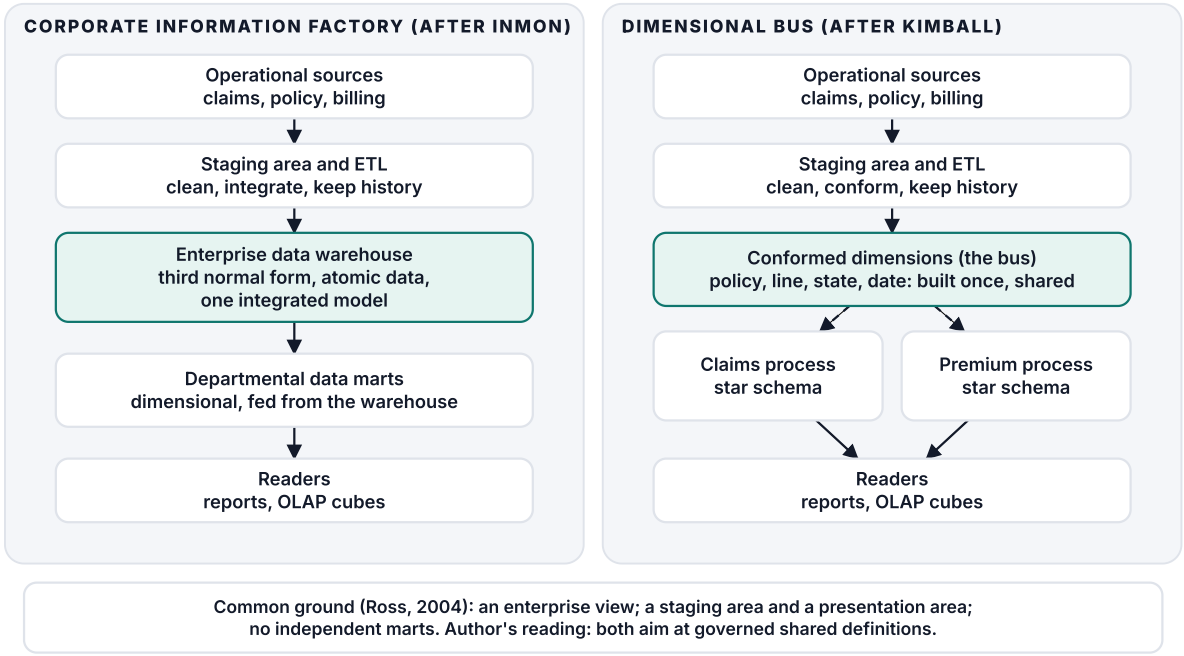
The warehouse idea was a deliberate break with the web. In 1988 the IBM Systems Journal printed "An Architecture for a Business and Information System" by Barry Devlin and Paul Murphy of IBM. Devlin later wrote that he and colleagues in IBM Europe had defined the architecture in 1986 for internal use. The move was simple to state: copy the business data out of the operational systems once, into a separate store designed for reading, keep its history there, and let the readers query that store instead of commissioning more extracts. One copy, built well, replaces many copies built in a hurry.

Two books then turned the idea into practice, and they disagreed on how to build it. W. H. Inmon's *Building the Data Warehouse* (QED, Boston, 1992) argued for an integrated enterprise warehouse fed from all sources. Ralph Kimball's *The Data Warehouse Toolkit* (Wiley, 1996), subtitled "practical techniques for building dimensional data warehouses", argued for building around **dimensional models**: **fact tables** that hold measurements, such as a claim payment or a premium, and **dimension tables** that hold the descriptions a reader filters and groups by, such as the policy, the line of business, the state or the date.

In 2004 Margy Ross of the Kimball Group summarized the two architectures side by side. In the approach that grew from Inmon's work, later called the Corporate Information Factory and "once known as the EDW approach", a third normal form relational database "containing atomic data is loaded" first, and that normalized warehouse "is used to populate additional presentation data repositories": data marts "tailored by business department/function". In the Kimball approach, the presentation area is dimensional, models are "built by business process", and integration comes from "the enterprise data warehouse bus architecture with common, conformed dimensions". A **conformed dimension** is one shared definition of, say, the policy or the calendar, managed once and reused by every fact table that needs it. The Kimball Group dates this bus architecture to the 1990s and draws it as a matrix, with business processes as rows and conformed dimensions as columns; Kimball wrote a column about that matrix in 1999.

FIGURE 2.2

Two ways to build a warehouse: integrate first in a normalized core, or integrate through shared dimensions



The architecture that grew from Inmon's work loads a normalized enterprise warehouse and feeds marts from it; Kimball's builds dimensional models by business process, which readers query, and joins them through conformed dimensions they share (dashed).

Author's diagram based on M. Ross, "Differences of Opinion", Kimball Group, 2004; W. H. Inmon, Building the Data Warehouse, 1992; R. Kimball, The Data Warehouse Toolkit, 1996.

Figure 2.2 puts the two side by side. The argument between the schools ran for years and was about the order of the work and where integration happens: in a normalized core before any mart is built, or in shared dimensions that let marts built one process at a time still add up. It was not settled, and this book does not settle it. What matters here is what Ross listed as common ground: both approaches "embrace the enterprise vantage point", both warn against independent marts that produce inconsistent results, and both separate a staging area, where data is cleaned and conformed, from a presentation area, where people query it. Both, in my reading, aim to give shared definitions one governed home instead of one per extract, though neither architecture by itself guarantees that a metric's formula lives there. The jobs that fill a warehouse are now usually called the **ETL** system (extract, transform, load); in the early 1990s most teams simply had extract programs, file transfers and load jobs.

The hop: 1994, the insurer builds its warehouse

In 1994 the insurer's director of management information built an enterprise data warehouse on DB2, a separate analytic copy of the claims and policy data. Monthly extract programs read the IMS databases and wrote fixed-width files; transfer and load jobs put them into normalized DB2 tables that kept history, organized as subject areas for claims and premium. Two warehouse developers owned the load jobs. Finance analysts, actuaries and report writers read the tables. The claims records carried personal data about claimants, so the copy took on the retention duties of statutory records.

The warehouse did what it was meant to do for one word. "Incurred loss" was defined once, in the load logic and the warehouse documentation, instead of in a dozen extract programs. Finance and underwriting could finally reconcile their totals, because they read the same table.

It did not carry the rule that mattered most to the board figure. The reserve adjustments the actuaries had pencilled on the 1987 printout had moved, by 1994, into a spreadsheet on an actuary's desk. Nobody put that rule into the warehouse. So the warehouse figure was still not the figure the controller reported: someone had to export it, apply the spreadsheet, and type the result into the board summary. The definition of the inputs had a home; the definition of the output did not. The Ledger records a **MEANING** loss: a rule that changes the value did not travel with it.

THE HOP · 1 · 1994 · IMS EXTRACTS (CLAIMS, POLICY) TO ENTERPRISE DATA WAREHOUSE ON DB2 (A SEPARATE ANALYTIC COPY)

Record	Property loss ratio, Q4 board figure (dataset), hop 1, 1994
From	IMS extracts (claims, policy)
To	Enterprise data warehouse on DB2 (a separate analytic copy)
Mechanism and schedule	Monthly extract programs, transfer and load jobs; subject areas for claims and premium
Format change; what was lost or assumed	MEANING Fixed-width extracts loaded to normalized DB2 tables with history. **Incurred loss is defined once in the warehouse, but the reserve adjustment rule stays in the actuaries' spreadsheet, so the warehouse figure still needs a manual step**
Accountable owner	Director of management information
Who can change it	Warehouse load jobs; two warehouse developers
Classification; retention or erasure duty	Financial and personal data (claimants); retention per statutory records rules
Read by (people, apps, models, agents)	Finance analysts, actuaries, report writers
Cost per month; basis	Share of mainframe DB2 capacity and two developers (illustrative)

One version of the truth, and what a governed history is for

Warehouse programs are often justified with a promise of "one version of the truth". The promise rarely holds, for reasons that are visible in the insurer's 1994 hop. A warehouse can hold one definition of each input and still leave the final rule outside, in a spreadsheet or a report tool. Departments that disliked the warehouse number built their own marts, and the spider web grew back one level higher. And even a perfectly governed warehouse answers "what do we know now", unless it also records what it knew at each point in time.

That last point is the one this book follows. Chapter One introduced the five times of a board number: the effective date, the knowledge cutoff, the calculation, the approval and the publication. The 1994 warehouse kept history of the data, which is a large gain: its monthly history tables could show the reserve loaded for a claim at any past month end. It did not keep the five times of the figure built from that data. The knowledge cutoff lived in the load

schedule; the approval lived in a memo; the hand adjustment lived in a spreadsheet. Each later platform in this book improved how history is stored, and each one, at its hop, dropped some of those times.

Chapter One closed with the metric passport for the Q4 2025 figure, three rows blank. The same passport, filled in as the 1994 warehouse could have filled it, shows how much of a board number's meaning lived outside the store even in a well-run warehouse. There was no written metric contract yet; the board's audit committee approved one in 2019.

METRIC PASSPORT

Metric id and version	None. "The loss ratio" was a line in the board summary.
Numerator	Incurred loss from the warehouse claims subject area, then the actuaries' reserve adjustment from a spreadsheet.
Denominator	Earned premium from the premium subject area.
Population and exclusions	Whatever the load jobs selected; not written down.
Basis	Statutory, by practice rather than by a written rule.
Effective date	The quarter label on the report.
Knowledge cutoff	Implied by the monthly extract schedule; not recorded with the figure.
Snapshot id or tag	None. History tables, but no marker for "the data behind this figure".
Transformation version	The load jobs as they stood that month; no version kept.
Owner and approver	Director of management information (the inputs); an actuary (the adjustment); neither named on the figure.
Restatement status	Not tracked. A corrected figure replaced the old one in the next summary.

So the case for a governed analytical store does not rest on that promise. It rests on three things a CTO can check. A definition lives in one place that has an owner. History is kept, so the data behind a past figure can be rebuilt. And the path from source to board figure is written down, including the steps that happen outside the store. The warehouse delivered the first two for many companies. The third is the one the insurer's thread shows going missing at almost every hop.

WHAT HASN'T CHANGED

Devlin and Murphy's 1988 architecture and a 2026 lakehouse make the same move: data is copied from the systems that write it into a store built for people who ask new questions. Gorry and Scott Morton's 1971 line about routine data processing still describes many dashboards: they report what the systems record, not what the manager has to decide.

ERA CARD

Reader. Actuaries and controllers reading printed reports, then finance analysts and report writers querying a separate warehouse.

Security. Each extract and each warehouse load was a new copy of claimant data with its own access; the binder and the spreadsheet were controls nobody audited.

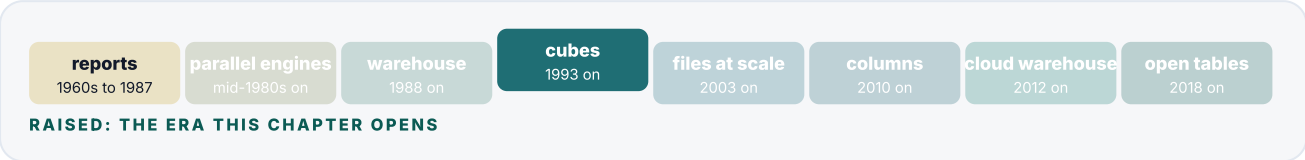
Cost. Mainframe batch time and people: the extract programmers, the warehouse developers and the actuary days spent on hand adjustments.

AI. None in this era; but an assistant reading today's tables inherits the same gap, because a rule kept outside the store is invisible to anything that reads the store.

Monday rule: for each figure your board reads, find every manual step between the governed store and the board pack, and give each one an owner and a written rule.

CHAPTER THREE

Cubes and the Frozen Answer



In 1993 a white paper by E. F. Codd, S. B. Codd and C. T. Salley, "Providing OLAP to User-Analysts: An IT Mandate", gave a name to a way of working with data. In its own words, "This paper defines a category of database processing: Online Analytical Processing (abbreviated OLAP)." The paper did not claim to have invented the way of working. It described a requirement already "called 'multidimensional data analysis'", and it closed by evaluating one product already on the market, Essbase. What the name gave buyers was a category and a checklist. What the products in that category gave the insurer, six years later, was a board number that was fast to read, easy to slice, and silently frozen at the moment it was built. The decision this chapter changes: **a precomputed answer must carry its build time beside it, everywhere it is shown**.

A name for a way of asking

Part I described the warehouse as a store built for questions. Part II, The Engines, looks at the machinery that answered them and stored their data, and at what each kept or dropped of a number's meaning. The cube comes first because it is the engine that put the board number in front of the board.

The Codd paper framed the analyst's need as a way of thinking. A user-analyst, it argued, sees the business in many dimensions at once: losses by line, by state, by quarter, compared with plan. Query and report tools of the time could fetch rows from a relational database; they were weak at consolidating, comparing and drilling across those dimensions. The paper set out twelve rules for OLAP tools, starting with a "multidimensional conceptual view" and including consistent reporting performance and unlimited dimensions and aggregation levels. The rules read today as a buyer's checklist, and that is how the market used them.

Two words from this period are worth keeping because they still describe engines in 2026. **Multidimensional OLAP (MOLAP)** stores precomputed totals, and usually a copy of the detail, in a structure built for the cube. **Relational OLAP (ROLAP)** keeps the data in

relational tables and computes or looks up totals there when asked. A hybrid keeps totals in the cube and detail in the relational store. Microsoft's current documentation for Analysis Services still describes all three storage modes, and it states the trade plainly: MOLAP answers queries without touching the source, while ROLAP "enables users to view data in real time" at the cost of slower queries.

The star schema and the cube

A cube is usually built from a star schema, the dimensional model of Kimball's 1996 book. At the center sits a **fact table**: for the insurer's claims, one row per claim transaction, with amounts for paid losses and reserve changes. Around it sit **dimension tables**: line of business, state, accident period, reporting period. A query joins the fact table to the dimensions it needs and sums the amounts.

The cube takes this one step further. Instead of summing at query time, a build job computes totals in advance for the combinations of dimension values its designers chose, and stores them as **cells**; products differ in how many they store and how many they compute when asked. A cell is a number at a coordinate: incurred loss, property line, Florida, fourth quarter. Reading a cell is fast because nothing is summed while the reader waits. Drilling from a national total to a state is a move from one stored cell to a group of others. For a CFO in 1999, with a board pack to prepare and a slow warehouse, that speed was the point.

The price of that speed is written in the same Microsoft page: "The data in the partition's MOLAP structure is only as current as the most recent processing of the partition." Between one processing and the next there is "a latency period during which data in OLAP objects may not match the source data." Nothing about this is a defect. A cube is a snapshot of answers. But a snapshot has a time, and the question is whether anyone can see it.

MDX and the business intelligence tool

Cubes invite questions that are long to write in plain SQL: "this quarter against the same quarter last year, by state, as a share of the parent total". On 9 September 1997 Microsoft announced public availability of the beta specification of **OLE DB for OLAP**, a set of interfaces for reaching OLAP data, previously code-named "Tensor"; eighteen software vendors and application developers announced support. Microsoft's documentation states that OLE DB for OLAP "defines multidimensional expressions (MDX)", a syntax for specifying a multidimensional dataset. A reporting tool that supported the specification could ask any compliant cube for cells in MDX, instead of learning each vendor's interface.

This is where a quiet change happened to the meaning of a board number. In the warehouse of Chapter Two, the definition of incurred loss lived in load logic and table design. In a cube, a team could also define **calculated measures**: a loss ratio written once, in MDX, inside

the cube, and reused by every report. That is convenient and, for a while, it is good governance: one formula, many readers. It also means the definition now lives in a second place, written in a second dialect, maintained by a business intelligence team rather than the warehouse team. When the warehouse definition changes, the cube definition has to change with it, and nothing forces that. The semantic layers of Chapter Ten are descendants of these cube formulas, with the same strength and the same risk.

The tools built on cubes, then called business intelligence tools, put the answer in front of executives as a grid or a chart. The reader saw a number and a title. The reader did not see when the cube was built, which warehouse load it read, or which formula produced the ratio. Those facts existed, in the build job's log and the cube's design. They were not on the screen.

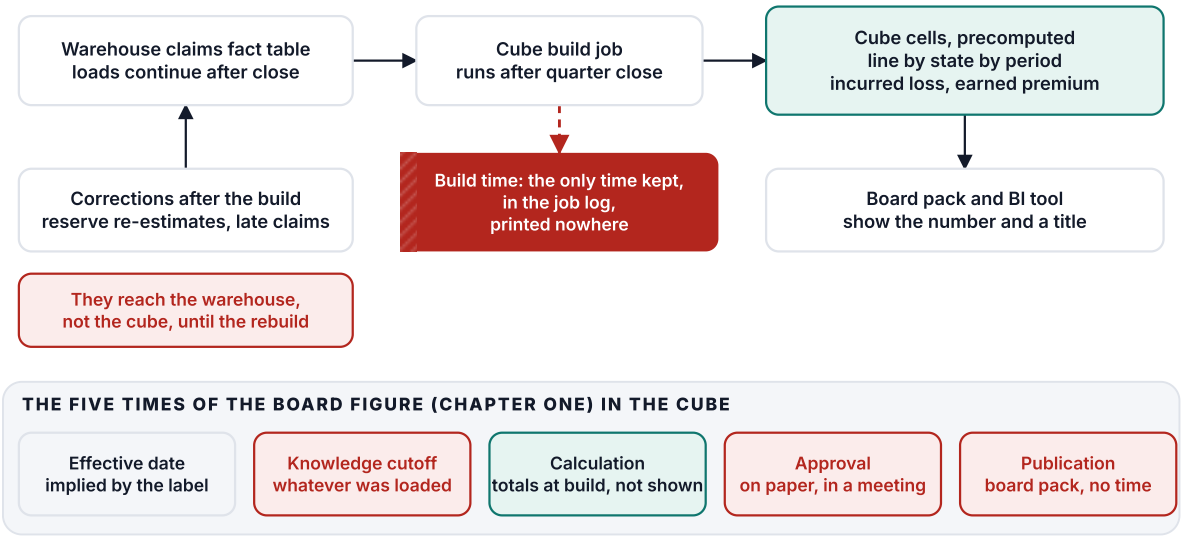
The hop: 1999, the board pack reads a frozen cube

In 1999 the insurer's director of management information gave the board pack its own model. The BI team built a claims star schema from the warehouse, with a fact table of claim transactions and dimensions for line, state and accident period, and on top of it a multidimensional cube of precomputed cells. Earned premium by line, state and quarter was loaded into the cube from the warehouse premium tables, so the cube's loss ratio measure could divide one by the other. After each quarter close a job rebuilt the cube; the board pack read the cube. The CFO, the board and the actuaries could drill to line and state, not to a single claim, and the cube held no claimant names, which kept personal data out of the most widely read copy.

Then corrections kept arriving, as they do in insurance. A reserve re-estimated in the week after close, a late-reported claim, a coding fix: each one reached the warehouse with the next load. None reached the cube until the next quarterly rebuild. The board figure reflected the data and rules present at the build and was silent about everything after. The build time was the only time the number carried, and it was printed nowhere: not on the board pack, not in the BI tool, not beside the ratio. The Ledger records a **TIME** loss.

FIGURE 3.1

The cube answers fast because it answers in advance; its build time is the hidden "as of"



The insurer's 1999 board cube (hop 2): rebuilt after each quarter close, read by the board pack, and silent about corrections that reached the warehouse after the build.

Author's diagram; MOLAP latency as described in Microsoft Learn, "Partition Storage Modes and Processing"; hop 2 from the thread (illustrative).

Figure 3.1 shows the anatomy. Of the five times from Chapter One, the cube kept one, its build time in the job log, and showed none. The build fixed the stored totals; the ratio, a calculated measure, divided them when a report asked, so the build time still decided its value. The effective date was implied by the quarter label. The knowledge cutoff was whatever the warehouse had loaded when the job ran. The approval happened in a meeting, on paper. A reader comparing the board pack with a warehouse query a month later would find two different numbers and no way to tell which was older. That is, in miniature, the problem the board faces in February 2026: the warehouse close table plays the frozen cube, and the lakehouse notebook plays the live warehouse query.

THE HOP · 2 · 1999 · ENTERPRISE DATA WAREHOUSE TO CLAIMS STAR SCHEMA AND A MULTIDIMENSIONAL CUBE FOR THE BOARD PACK

Record	Property loss ratio, Q4 board figure (dataset), hop 2, 1999
From	Enterprise data warehouse
To	Claims star schema and a multidimensional cube for the board pack
Mechanism and schedule	Cube rebuilt after each quarter close; the board pack reads the cube
Format change; what was lost or assumed	TIME Fact table of claim transactions, dimensions for line, state, accident period; pre-aggregated cube cells. **The cube's build time is the only 'as of', and it is printed nowhere; later corrections reach the warehouse but not the frozen cube**
Accountable owner	Director of management information
Who can change it	BI team (cube design and rebuild job)
Classification; retention or erasure duty	Aggregated financial data; no claimant names in the cube
Read by (people, apps, models, agents)	CFO, board, actuaries (drill to line and state, not to claim)
Cost per month; basis	OLAP server licence and a small BI team (illustrative)

WHAT HASN'T CHANGED

Precomputed answers are still everywhere: materialized views, extract files behind dashboards, cached query results, semantic model refreshes. Each one is, like the 1999 cube, only as current as its last refresh, and many dashboards still do not print that refresh time beside the number.

ERA CARD

Reader. The CFO and the board, through a BI tool: fast drill by line and state, with the number's build time out of sight.

Security. The cube kept claimant names out of the most widely read copy; aggregation was a privacy control as well as a speed trick.

Cost. An OLAP server licence, a small BI team and a build window after each quarter close; queries were cheap because the work was done in advance.

> **AI.** An assistant that answers from a cached semantic model inherits the cube's problem: it quotes the last refresh as if it were now, unless the refresh time travels with the answer.

Monday rule: for every dashboard the board or the executive team reads, make the refresh time of each number visible on the screen, next to the number.

CHAPTER FOUR

Parallel Engines and the Appliance



In September 2008 Oracle introduced Exadata, a rack that put database servers and new storage servers together and pushed part of each query's filtering down into the storage. Oracle wrote the software; Hewlett-Packard built the hardware. Six years earlier, on 23 September 2002, a company founded in 2000 called Netezza had announced the NPS 8000, which it described as a "purpose-built, tera-scale data appliance". The same year Oracle shipped Exadata, the insurer bought an appliance of its own and moved its warehouse off DB2. The machine was faster than anything the insurer had owned, and it came as one box: hardware, storage, database software, loader, support contract and the know-how of vendor-trained administrators. The decision this chapter changes: **everything a platform bundles is something an exit has to rebuild, so list the contents before you sign**.

Many processors, and what they share

An analytical query reads a lot of data and returns a little. The straightforward way to make it faster is to split the data across many processors and disks, let each scan its part, and combine the results. The design question is what the processors share. In March 1986 Michael Stonebraker set out the three choices in a short paper, "The Case for Shared Nothing", in the IEEE bulletin *Database Engineering*:

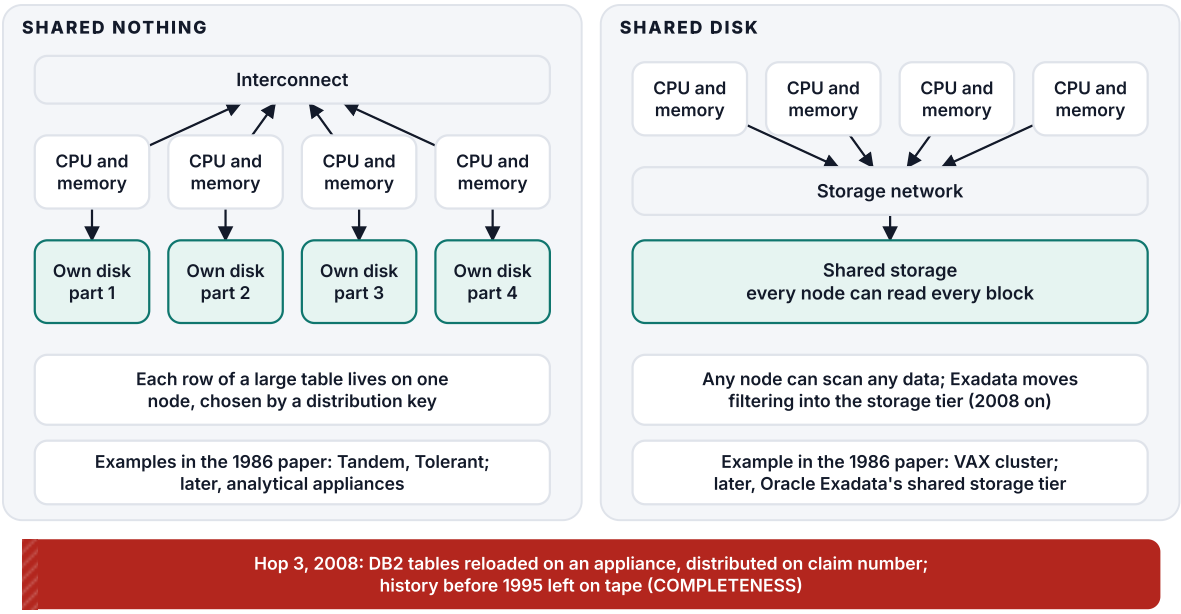
- **Shared memory**: "multiple processors shared a common central memory".
- **Shared disk**: "multiple processors each with private memory share a common collection of disks".
- **Shared nothing**: "neither memory nor peripheral storage is shared among processors".

Stonebraker wrote about high transaction rate systems, and he argued that shared nothing was the preferred approach. His comparison table rated it best at scaling to a large number of machines and poorest at database design, load balancing and number of messages, problems he expected tuning aids to make manageable. Analytical engines took up the same idea; engines that spread one query over many such nodes are called massively parallel

processing (MPP) systems. In a shared-nothing warehouse each row of a large table belongs to one node (small tables are often copied to every node), chosen by a **distribution key**, often a hash of a column such as the claim number. A query runs on every node at once against its own rows, and the nodes exchange data only to join or combine. Choose the key well and the work spreads evenly. Choose it badly and one node holds far more rows than the others (data skew), and every query waits for that node.

FIGURE 4.1

Shared nothing splits the data among nodes; shared disk lets every node reach all of it



Two of the three architectures Stonebraker compared in 1986, as analytical engines used them; the insurer's 2008 appliance was shared nothing, distributed on claim number (hop 3).

Author's diagram based on M. Stonebraker, "The Case for Shared Nothing", Database Engineering 9(1), 1986; Oracle, "Oracle Exadata: A guide for decision makers"; hop 3 from the thread (illustrative).

Figure 4.1 shows two of the three designs as analytical engines used them. Teradata built a parallel database computer, the DBC/1012, which Jack Shemer and Philip Neches described in IEEE *Computer* in November 1984; the company's own publicity spoke of "over 1,000 microprocessors operating in parallel" and of databases up to a trillion bytes. By March 1986 another paper in the same issue of *Database Engineering* could list "commercial database machines, such as Britton Lee's IDM-500 and Teradata" among those that "have begun to appear". Parallel database machines were on the market by the mid-1980s.

Columns before C-Store

A second idea changed what each node reads. A conventional database stores a row's values together, which suits programs that write and read one record at a time. A query that sums one amount over millions of claims needs one column, yet reads every column of every row. Storing each column separately lets the engine read only what the query names.

The idea is older than its fame. At ACM SIGMOD in May 1985, George Copeland and Setrag Khoshafian of MCC presented "A Decomposition Storage Model": each table split into binary relations, each "containing a surrogate and one attribute". They were careful about their claim: "The purpose of this report is not to claim that decomposition is better", only that the consensus against it was not well founded. They also cited fully transposed systems from 1971 onward. Twenty years later, at VLDB 2005, Stonebraker and colleagues presented C-Store, "a read-optimized relational DBMS" that stored data by column, compressed it, and kept overlapping sorted projections. The C-Store paper itself noted that column storage had already been "demonstrated in the warehouse marketplace by products like Sybase IQ, Addamark, and KDB". No one paper invented the column store; each made the case again for its own time. Chapter Six follows columns into files, where Parquet and ORC made them portable.

The appliance: easy to buy, hard to leave

By the 2000s, vendors packaged parallel engines as appliances. Netezza's 2007 prospectus to the US Securities and Exchange Commission described its product as one that "integrates database, server and storage platforms in a purpose-built unit". Each of its processing units combined a commodity processor, memory, a disk and a field programmable gate array that "filters records as they stream off the disk", so less data reached the processor. The company said it "first began shipping products in February 2003" and had shipped over 200 appliances to 87 customers by 31 January 2007. Oracle's Exadata took the shared-disk road: by Oracle's description, its storage servers run database processing "in the shared storage tier", and its first version focused on data warehousing by "moving database filtering operations into storage", a feature Oracle calls Smart Scan.

Why buyers liked the bundle is easy to see from the insurer's side of the table. One purchase replaced a dozen decisions. The hardware, storage layout and database were tuned together by people who knew them. One support contract covered all of it. The appliance arrived and was fast in weeks, not after a year of tuning. For a data team that had spent years fighting a shared mainframe for batch time, that was worth a great deal.

What made it hard to leave came in the same box. The Lakehouse paper (Armbrust and colleagues, CIDR 2021) describes that generation as systems that "typically coupled compute and storage into an on-premises appliance", which "forced enterprises to provision and pay

for the peak of user load and data under management". Capacity came in racks; growth meant a bigger purchase. The less visible coupling was in the work built on top: stored procedures in the vendor's dialect, load scripts written for the vendor's loader, tables designed around its distribution keys, monitoring tied to its tools, and administrators trained on its habits. None of that is data, and none of it moves when the data moves. It is the earliest version of what this book calls **code that does not travel**.

The hop: 2008, the warehouse moves to the appliance

In 2008 the insurer's head of data warehousing moved the enterprise warehouse from DB2 to a shared-nothing appliance in one rack. It was a one-time migration: unload the DB2 tables, convert them, reload them as appliance tables distributed on claim number, and rewrite the load jobs and stored procedures for the new engine. Vendor-trained administrators and the ETL team took over the platform. Finance, the actuaries, the board cube of Chapter Three and the pricing analysts read from it.

The project moved history from 1995 onward. Claims history before 1995 stayed behind on tape, in the old load format, with a record layout no current job could read. Nobody decided to delete it; it simply did not make the list. For a loss ratio that compares this quarter with the same quarter last year, the gap did not show. For an actuary who wants long development history, or an auditor who asks how a 1993 reserve was set, it did. The Ledger records a **COMPLETENESS** loss: rows that existed were not carried.

THE HOP · 3 · 2008 · DB2 ENTERPRISE DATA WAREHOUSE TO MPP WAREHOUSE APPLIANCE (SHARED-NOTHING, ONE RACK)

Record	Property loss ratio, Q4 board figure (dataset), hop 3, 2008
From	DB2 enterprise data warehouse
To	MPP warehouse appliance (shared-nothing, one rack)
Mechanism and schedule	One-time migration: unload, convert, reload; load jobs and stored procedures rewritten for the appliance
Format change; what was lost or assumed	COMPLETENESS DB2 tables to appliance tables distributed on claim number. **History before 1995 not migrated; kept on tape in the old load format, with a record layout no current job can read**
Accountable owner	Head of data warehousing
Who can change it	Appliance DBAs (vendor-trained); ETL team
Classification; retention or erasure duty	Financial and personal data; retention per statutory records rules
Read by (people, apps, models, agents)	Finance, actuaries, the cube, pricing analysts
Cost per month; basis	Appliance purchase and annual support contract (illustrative)

The incumbents are still running

This chapter is history, but its engines are not. Teradata offers its platform today on AWS, Microsoft Azure and Google Cloud and for hybrid deployments. Oracle introduced its Exadata generation X11M in January 2025, and Exadata has been available in Oracle's public cloud since October 2015. IBM offers Db2 Warehouse as a managed service and as software. An enterprise may run one of these beside a newer platform, and a board figure may still come from one. This book does not rank them against each other or against the cloud platforms of Part III. It asks the same question of each: what would have to be rebuilt to reproduce an important number somewhere else.

WHAT HASN'T CHANGED

Stonebraker's 1986 choice between sharing nothing, sharing disks and sharing memory is still a useful first question to ask of any engine. The same choice returns in Chapter Seven, where cloud warehouses keep data in shared storage and give each workload its own compute.

ERA CARD

Reader. Finance, actuaries, pricing analysts and the board cube, all reading one fast appliance.

Security. One rack in the company's data center held every claim and every claimant; access was managed inside the appliance, by its administrators, in its tools.

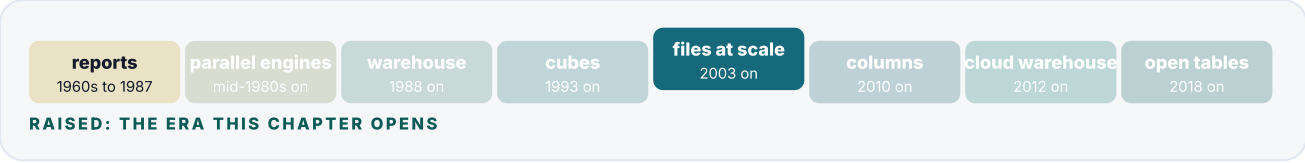
Cost. A capital purchase sized for the peak, plus an annual support contract; growth came in racks.

AI. Models built in this era trained on whatever history the appliance held; the years left on tape were invisible to them, and still are to anything that reads only the warehouse.

Monday rule: for each analytical platform you run, write down what the vendor bundles beyond the data (procedures, loaders, policies, schedulers, skills) and how long each would take to rebuild elsewhere.

CHAPTER FIVE

Files at Scale: Hadoop and the Lake



In October 2003, at the ACM Symposium on Operating Systems Principles in Bolton Landing, New York, three Google engineers, Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung, described the file system that held Google's data. One line in their list of design assumptions reads, in full: "Small files must be supported, but we need not optimize for them." It was a sensible choice for a system built to hold a few million very large files. Eleven years later the insurer built a data lake on an open source system modelled on that design and began to drop adjuster notes, photos and claim extracts into it every day, many of them small, most of them owned by nobody. The appliance of Chapter Four had a vendor, a DBA team and a budget line. The lake had a platform team that kept the cluster running and no one who answered for what was in it. The decision this chapter changes: **no folder in a lake receives a file until it has a named owner and a retention rule**.

Two papers from Google

The Google File System paper (Ghemawat, Gobioff and Leung, SOSP 2003) starts from cheap machines that fail. It describes a system that "provides fault tolerance while running on inexpensive commodity hardware". Files are cut into fixed-size **chunks** of 64 MB, "much larger than typical file system block sizes", and each chunk is copied to several machines: "By default, we store three replicas." One **master** server keeps all the metadata (the names of files, which chunks make up each file, where the copies live) in memory, and the paper notes that the master keeps less than 64 bytes of metadata for each 64 MB chunk. That works when files are big. The paper expected "a few million files, each typically 100 MB or larger in size", and called multi-GB files "the common case".

A year later Jeffrey Dean and Sanjay Ghemawat published "MapReduce: Simplified Data Processing on Large Clusters" (OSDI 2004). In its words, "MapReduce is a programming model and an associated implementation for processing and generating large data sets." A programmer writes two functions. The **map** function reads records and emits key and value

pairs; the **reduce** function receives every value for one key and combines them. The framework splits the input, runs the pieces on many machines and reruns the work of machines that fail. The paper's point for a CTO is about labor, not speed: a team that could not write a distributed program could now write two small functions and let the framework handle the cluster.

Together the two papers moved storage and computation onto the same racks of ordinary servers. A warehouse appliance sold a balanced box; this design sold nothing at all. It was a published design, and anyone could implement it.

Hadoop at Apache

Someone did. On 1 February 2006 Doug Cutting opened the first issue in the new Hadoop project's tracker at the Apache Software Foundation, HADOOP-1, titled "initial import of code from Nutch", with the note "The initial code for Hadoop will be copied from Nutch." Nutch was an Apache web search engine project, and the Hadoop documentation still says that its file system, HDFS, "was originally built as infrastructure for the Apache Nutch web search engine project". Hadoop began as a subproject of Apache Lucene. On 16 January 2008 the ASF board approved the resolution to "Establish the Apache Hadoop Project" as a top-level project of the foundation, discharging the Lucene project of "all responsibilities pertaining to the Apache Lucene Hadoop sub-project".

Yahoo's part in the early years shows in Apache's own records rather than in legend. The Hadoop project's news page for July 2008 reports that a Yahoo Hadoop cluster sorted one terabyte in 209 seconds and won the annual terabyte sort benchmark, adding: "This is the first time that either a Java or an open source program has won." When the foundation announced Hadoop 1.0 on 4 January 2012, its announcement quoted a Forrester analyst describing the framework as "originating with technologies developed by Yahoo, Google, and other Web 2.0 pioneers in the mid-2000s".

HDFS kept the shape of the Google design. Its architecture guide gives "a typical block size used by HDFS" as 128 MB, lets each file carry its own replication factor (its examples use three), and states that "the NameNode keeps an image of the entire file system namespace and file Blockmap in memory". Files are written once: "A file once created, written, and closed need not be changed except for appends and truncates." That last rule matters later in this book. A lake built on HDFS does not update a claim row in place; it writes a new file, and some job or some reader must decide which file holds the current truth.

The project is not a museum piece. The Hadoop site lists release 3.4.3 on 24 February 2026 and 3.5.0, "the first stable release of Apache Hadoop 3.5 line", on 2 April 2026. Many companies still run clusters, and many more run its ideas on object storage.

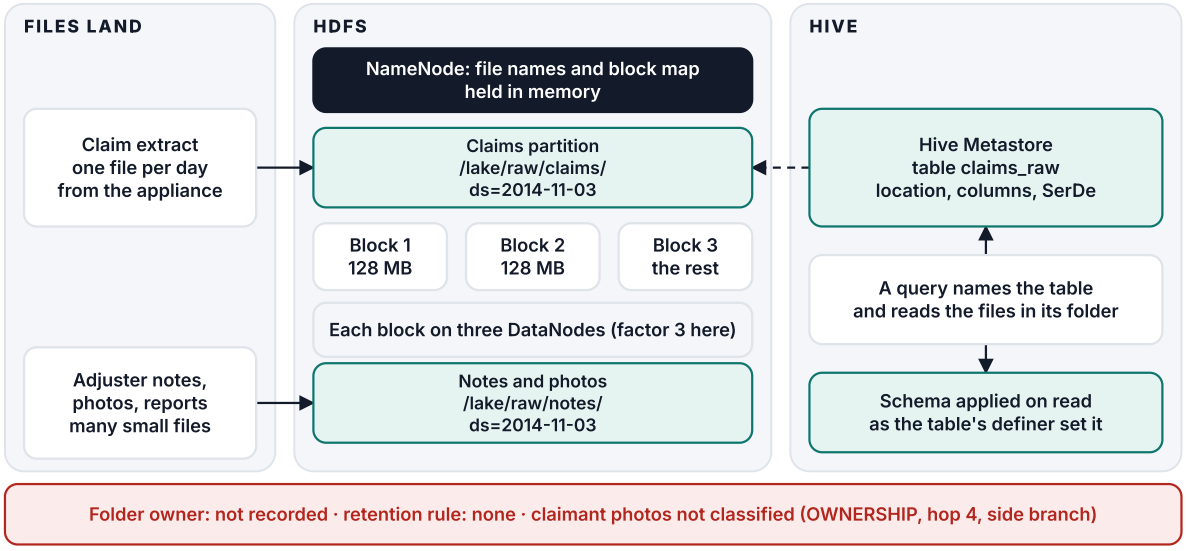
Hive: tables drawn over files

MapReduce asked for Java programs; analysts wrote SQL. Ashish Thusoo and colleagues on Facebook's data infrastructure team described Hive in "Hive: A Warehousing Solution Over a Map-Reduce Framework" (PVLDB, volume 2, 2009). Hive accepts queries in HiveQL, "a SQL-like declarative language", "which are compiled into map-reduce jobs executed on Hadoop". The paper reports that "the Hive warehouse contains several thousand tables with over 700 terabytes of data" at Facebook, used "for both reporting and ad-hoc analyses by more than 100 users". Hive graduated from Hadoop to its own Apache top-level project in September 2010.

The Hive data model is simple, and its simplicity explains a decade of lakes. "Each table has a corresponding HDFS directory," the paper says, and the table's data is "serialized and stored in files within that directory". A **partition** is a subdirectory: a table in `/wh/T` partitioned on date and country keeps the rows for one date and one country in a directory such as `/wh/T/ds=20090101/ctry=US`. How to turn bytes into columns is the job of a **SerDe** (serializer and deserializer), named per table. The table's schema, its partitions and its SerDe are kept in a separate catalog, the **Hive Metastore**, which the paper describes as "containing schemas and statistics". Hive could also define **external tables** over data that was already sitting in a directory.

FIGURE 5.1

Claim files in the lake: HDFS stores the blocks, Hive draws a table over the folder, and nobody owns the folder



The files are kept as they arrived; the Metastore points a table at a folder, and the schema registered for the table is applied only when a query reads the files.

Author's diagram based on Thusoo et al., PVLDB 2 (2009), section 2.1 (tables as HDFS directories, partitions as subdirectories, SerDe, Metastore), and the Apache Hadoop HDFS Architecture guide (128 MB blocks, NameNode namespace in memory). Folder name and table are illustrative.

Figure 5.1 follows one claim extract through that stack. The file lands in a date folder. HDFS cuts it into blocks and copies each block three times. The Metastore records that a table called `claims_raw` lives in that folder and that a given SerDe reads its lines. A query names the table; Hive looks up the folder, reads the files in it (or in the partitions the query selects), and applies the schema while it reads. That is **schema on read**: the file is stored as it arrived, and the schema registered for the table is applied only when a query reads it, so the file means what the table's definition makes of it, or fails to read. The warehouse of Chapter Two did the opposite, schema on write: nothing loaded until it fitted the model. The Lakehouse paper (Armbrust and colleagues, CIDR 2021) sums up the trade: the lake was "a schema-on-read architecture" that gave agility but "punted the problem of data quality and governance downstream".

Downstream is a place with people in it. When two teams define two tables over the same folder, with two SerDes, they have two meanings for one file and no rule for which one wins.

The lake, as Dixon explained it

On 14 October 2010, after Pentaho announced its first Hadoop release at Hadoop World in New York, James Dixon of Pentaho wrote a blog post titled "Pentaho, Hadoop, and Data Lakes". He used the term and explained it with an image. A data mart, he wrote, is like "a store of bottled water", "cleansed and packaged and structured for easy consumption", while "the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples."

Two details of the post are often lost. Dixon wrote from interviews with Hadoop users, and among the common traits he listed was that "the source of the data is typically a single application or system". His lake was fed by a source, singular, and held its detail because "there are many unknown questions that will arise in the future". The problem he named was the data mart's: only "a subset of the attributes are examined", and aggregation hides the lowest level. He did not describe a place where every department drops every file. Many companies built that place anyway and called it by his name.

What a lake of many sources needs is what Dixon's lake of one source could take for granted: someone who knows what each stream is. Without that, the lake fills with files whose producer has moved on, whose columns nobody can explain and whose personal data nobody has classified. Practitioners came to call such a lake a swamp. The condition has a simple test: pick a folder at random and ask who may delete it.

> **WHAT HASN'T CHANGED**
>
> The Google File System paper of 2003 kept every file name in one server's memory and said plainly that small files would be supported but not optimized. HDFS kept the same arrangement in its NameNode. Every file, however small, costs the name server a file entry and at least one block entry, so a million tiny files cost it far more memory than the same bytes in a few large ones. Object stores in 2026 show their users no such name server, yet a table made of a million tiny files is still slow to list, slow to plan and expensive to read, because every file is a separate request.

Small files, partitions and the cost of a folder

The two habits that made lakes easy to fill also made them slow to read. The first is the **small file**. A job that writes one file per hour, per source, per partition, produces tens of thousands of files a year, each far smaller than a block. Each one costs the name server memory, each one is a separate task or request for the engine, and each one has to be opened

before a query can decide that it did not need it. The usual cure is **compaction**, a job that rewrites many small files into a few large ones. It works, and it is one more job with its own schedule and its own chance to fail.

The second is the **partition**. Partitioning by date lets a query that asks for one month skip the folders of every other month, which is the cheapest speed-up a lake has. Partitioning by too many columns, or by a column with too many values, multiplies the folders and turns every partition into a small file. A partition scheme is also a decision about meaning: if claims are partitioned by the date the file arrived, a query for "claims in December" by loss date cannot prune by folder and reads all of them. The folder name is the nearest thing to an "as of" that a Hive table carried, and it was the time of the file, not the time of the loss. Both habits are reasons to treat a folder as a decision with an owner: who chose the partition key, who runs compaction, who may delete.

The hop: the insurer's lake, 2014

In 2014 the insurer built a Hadoop data lake beside the appliance. Daily file drops brought adjuster notes, claim photos and third-party reports into a raw zone, next to extracts from the appliance. Hive tables were defined over the files, nightly jobs refined some of them, and the tabular extracts were stored as ORC, a columnar file format that Chapter Six opens. Data scientists, fraud analysts and the pricing team read it.

The appliance's tables each had a DBA and a load job. The raw zone had neither. The head of analytics answered for the curated tables; the Hadoop platform team answered for the cluster, its disks and its uptime; and anyone with write access to the raw zone could add a folder. Nobody was recorded as the owner of the notes and photos themselves, nobody set a retention rule, and the claimant photos, personal data that may also be sensitive, were never classified. Schema on read meant each team read the notes its own way, and no team was recorded as their owner. This is the loss this book calls **OWNERSHIP**: a copy with no accountable owner.

THE HOP · 4 · 2014 · APPLIANCE EXTRACTS AND RAW CLAIM FILES (ADJUSTER NOTES, PHOTOS, THIRD-PARTY REPORTS) TO HADOOP DATA LAKE BESIDE THE APPLIANCE (HDFS, HIVE TABLES STORED AS ORC)

Record	Claim files beside the board figure's path (a side branch; the figure itself did not pass through the lake), hop 4, 2014
From	Appliance extracts and raw claim files (adjuster notes, photos, third-party reports)
To	Hadoop data lake beside the appliance (HDFS, Hive tables stored as ORC)
Mechanism and schedule	Daily file drops into a raw zone; Hive tables defined over the files; nightly jobs
Format change; what was lost or assumed	OWNERSHIP Text, images and PDFs as files; tabular extracts as ORC; schema applied when read. **Adjuster notes and photos land in a raw zone with no accountable owner and no retention rule; schema on read means each team reads them differently**
Accountable owner	None recorded for the raw zone; head of analytics for curated tables
Who can change it	Anyone with write access to the raw zone; the Hadoop platform team
Classification; retention or erasure duty	Personal and sensitive data (claimant photos, notes) not yet classified
Read by (people, apps, models, agents)	Data scientists, fraud analytics, the pricing team
Cost per month; basis	Cluster hardware, support subscription and a platform team (illustrative)

Why did a careful insurer, which had kept its books since 1898, build an unowned store? Because every incentive pointed that way. Storage on commodity disks was cheap enough that keeping everything cost less than deciding what to keep. Schema on read let a team land data before anyone had agreed what it meant, which was the point. The platform team was measured on uptime, not on contents, and the people who produced the files, adjusters and outside firms, did not know the lake existed. Ownership was nobody's job, so it was nobody's failure.

The lake did not go away when the warehouse moved. In 2021 the curated tables fed the cloud warehouse (Chapter Seven), though no loss-ratio input came from them, and the cluster kept running for the claim files. It ran until 2024, when those files moved into the

lakehouse of Chapter Eight. For ten years the raw zone held claimant photos with no owner and no retention rule, and each migration copied the problem forward rather than solving it.

The board number did not pass through the raw zone; in 2014 the loss ratio still came from the appliance. Hop 4 is a side branch of its path, and its OWNERSHIP loss belongs to the claim files, not to the board figure. The lake matters to it because every later platform inherited the lake's habit: land the data first, decide what it means later, and leave ownership to the next project.

> **ERA CARD**
>
> **Reader.** The data scientist and the batch job: programs that read whole folders and decide the schema while they read.
>
> **Security.** Access was granted on directories, so whoever could read a folder could read every column of every file in it, including photos and notes nobody had classified.
>
> **Cost.** Disks became cheap enough to keep everything, which moved the cost from storage to people: compaction jobs, a platform team and the hours spent working out what an old file means.
>
> **AI.** The lake's raw files are the material later models are trained and grounded on; a file with no owner arrives in a model's corpus with no one to say what it may be used for.

Monday rule: list the top-level folders in your lake or object store, write a named owner and a retention rule beside each one, and freeze writes to any folder that still has a blank after thirty days.

CHAPTER SIX

Columns on Disk: Parquet, ORC and Arrow



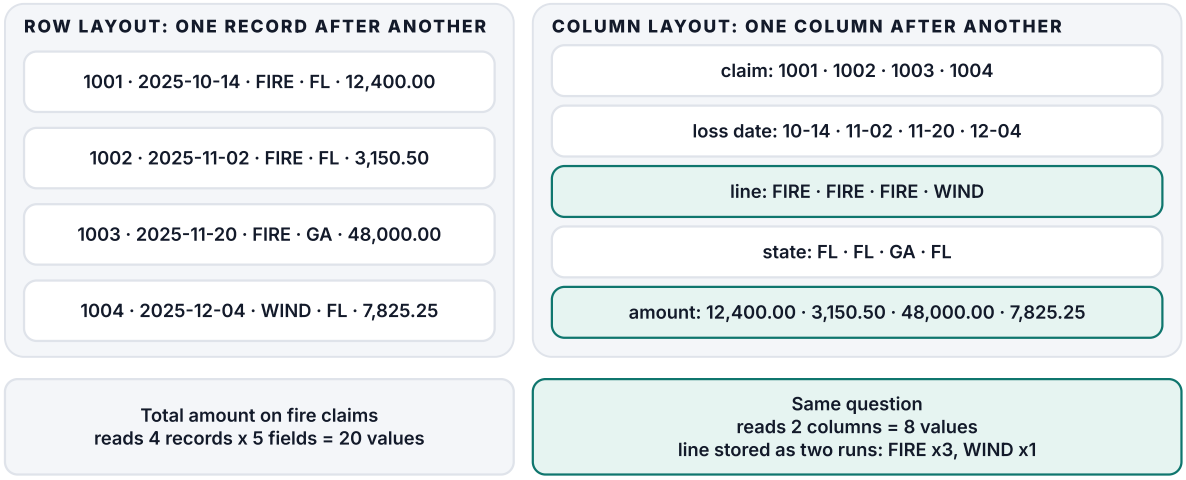
In September 2010, at the VLDB conference in Singapore, Sergey Melnik and six Google colleagues described Dremel, a query system that, in the words of their abstract, could run "aggregation queries over trillion-row tables in seconds". Part of the answer was how the data sat on disk: not one record after another, but one field after another, even for records nested several levels deep. Within three years that idea was in open file formats that many engines could read. The insurer used two of them. Its 2014 lake stored claim extracts as ORC, and its 2021 move to the cloud warehouse unloaded the appliance's tables to Parquet, with the amounts kept as exact decimals and the policy status as a bare integer. The files did what they promised, and the status code still lost its meaning. The decision this chapter changes: **treat a file format as a storage contract, and write down what it does not carry**.

Rows and columns: the same claims, two layouts

A table is a grid, but a disk is a line. Something has to decide the order in which the cells of the grid are written. A **row layout** writes one record, then the next: claim number, loss date, line of business, state and amount for the first claim, then the same fields for the second. A **column layout** writes every claim number, then every loss date, then every line of business, and so on. Chapter Four followed column orientation inside database engines; this chapter follows it into files that sit on shared storage and are read by many engines.

FIGURE 6.1

The same four claims, two layouts: a column layout reads two columns where a row layout reads every field



For the question "total amount on fire claims", the row layout passes over all 20 values; the column layout reads the line and amount columns, 8 values, and the repeating line of business compresses to two runs.

Author's diagram. Claims and amounts are illustrative; the layouts follow Melnik et al., PVLDB 3 (2010), figure 1 (record-wise vs columnar representation).

Figure 6.1 shows why analytical engines like the second layout. A question such as "total incurred on fire claims this quarter" needs two fields out of dozens. In a row layout the reader passes over every field of every record to reach them. In a column layout it reads the two columns and leaves the rest of the file untouched. The second gain is compression. Values of one column sit next to each other and look alike: the line of business repeats, dates cluster, states come from a short list. Alike values compress far better than a mix of names, dates and amounts.

The cost runs the other way. Changing one claim means rewriting the part of the file that holds it, in every column, and even an append writes a value into every column. A row layout suits a system that writes a record at a time, such as the claims system. A column layout suits a reader that scans millions of records and a few fields, such as the loss ratio. That is why analytical copies exist at all, and why the hop from one layout to the other is a hop in this book's sense: a job decides what travels.

From Dremel to three file formats

Dremel (Melnik and colleagues, "Dremel: Interactive Analysis of Web-Scale Datasets", Proceedings of the VLDB Endowment, volume 3, 2010) presented "a novel columnar storage representation for nested records". The hard part was nesting. A record may hold a list of names, each with a list of languages. If each field is stored as its own column, the reader needs a way to rebuild which values belonged to which record and which list. Dremel attaches two small numbers to every value. The **repetition level** "tells us at what repeated field in the field's path the value has repeated"; the **definition level** records how many of the optional or repeated fields above the value are actually present, so that a missing field can be told apart from a missing parent. Splitting records into columns this way is called **record shredding**; putting them back together is record assembly.

Hadoop's world arrived at columns by a different road. In 2011 a team from Facebook, Ohio State University and the Chinese Academy of Sciences presented **RCFile** (He and colleagues, ICDE 2011), a "Record Columnar File". Its rule is "first horizontally-partition, then vertically-partition": a table "is first horizontally partitioned into multiple row groups", and then "each row group is vertically partitioned so that each column is stored" together. The paper reports that RCFile "has been chosen in Facebook data warehouse system as the default option". The row group was the important idea. It keeps all the fields of a set of rows in one block of storage, so a record can be rebuilt without reading across the cluster, while each field is still stored by column inside the group.

ORC (Optimized Row Columnar) took the same shape further. Its project page says, "Back in January 2013, we created ORC files as part of the initiative to massively speed up Apache Hive." Hive's documentation states that ORC was "introduced in Hive version 0.11.0", and Hive's download page dates that release to 15 May 2013. An ORC file is divided into **stripes**, each "self contained", and inside a stripe the writer keeps an index for each set of rows, which "default to 10,000 rows", holding "the minimum and maximum values for each column". The insurer's hop 4 tables in Chapter Five were ORC: the claim extracts in the lake carried these statistics from 2014.

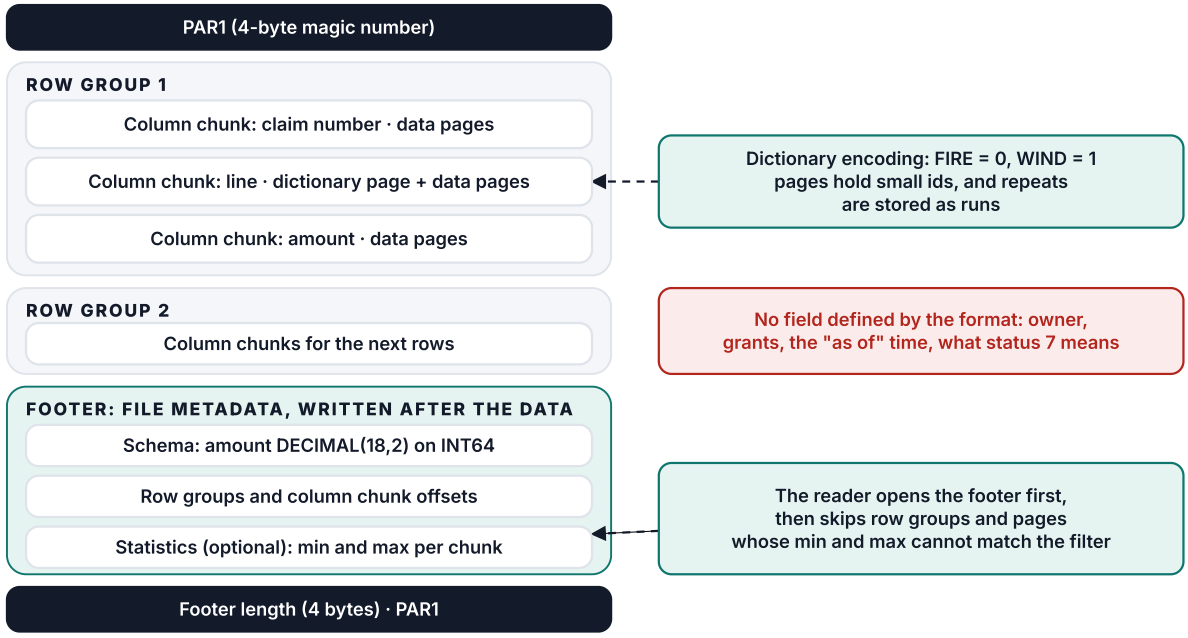
Parquet came from Twitter and Cloudera. Twitter announced Parquet 1.0 on 30 July 2013 as "an open-source columnar storage format" for Hadoop, "the result of a collaboration between Twitter and Cloudera". Its format description says it "uses the record shredding and assembly algorithm described in the Dremel paper". Parquet became an Apache top-level project in April 2015. The files of hop 5 in Chapter Seven, the appliance's tables unloaded for the cloud warehouse in 2021, were Parquet, and the lakehouse of Chapter Eight stores its data in Parquet files too.

Inside a Parquet file

The Parquet specification defines three nested units. A **row group** is "a logical horizontal partitioning of the data into rows". A **column chunk** is "a chunk of the data for a particular column"; column chunks "live in a particular row group and are guaranteed to be contiguous in the file". A **page** "is conceptually an indivisible unit (in terms of compression and encoding)". In the specification's summary: "a file consists of one or more row groups. A row group contains exactly one column chunk per column. Column chunks contain one or more pages."

FIGURE 6.2

A Parquet file decoded: row groups of column chunks, pages inside them, and a footer that describes it all



A reader opens the footer first, uses the schema and, when the writer recorded them, the minimum and maximum of each column chunk to skip what cannot match, and decodes only the pages it needs; the format defines no field for owner, grants or the "as of" time, though optional key and value metadata could carry them by convention. Dashed lines point to the part described.

Author's diagram based on the Apache Parquet format documentation (file format, concepts, encodings, page index) and LogicalTypes.md (DECIMAL). Column names are illustrative.

The file begins and ends with the four bytes `PAR1`. The **footer**, near the end, holds the file metadata: the schema with every column's type, the list of row groups, where each column chunk starts and, when the writer records them, statistics about the values. "File metadata is

written after the data to allow for single pass writing," the format says. A writer streams rows out and describes them at the end; a reader does the reverse, and reads the footer before it touches any data.

Two encodings do most of the work on columns like the insurer's. **Dictionary encoding** "builds a dictionary of values encountered in a given column": the line of business is stored once as FIRE, WIND and so on, and the pages hold small numbers that point into the list. **Run-length encoding** stores a value and how many times it repeats; Parquet combines it with bit-packing in a hybrid that, in its words, stores "repeated values" more efficiently. If the dictionary "grows too big, whether in size or number of distinct values", the writer falls back to plain encoding. A claim number, unique per row, gains little from a dictionary; a line of business or a state gains a great deal.

The statistics, optional in the format, are what make a file fast to skip. Parquet "stores min/max statistics at several levels (such as Column Chunk, Column Index, and Data Page)". A query that asks for losses dated in the fourth quarter of 2025 reads the footer, sees that a row group's loss dates run from March to June, and never reads that row group. Engines call this **predicate pushdown**: the filter is pushed down to the file reader, which uses the statistics to prune. It works best when the data is sorted or clustered on the filtered column. A file written in random order has every row group spanning every date, and its statistics prune nothing. Partition folders, from Chapter Five, prune whole directories before the reader opens a file; statistics prune inside it.

Arrow: the columns in memory

A file format solves storage. Once an engine has read the columns, it must hold them in memory, and every engine used to hold them its own way. Passing data from one system to another, say from a query engine to a data science library, meant converting it, value by value. On 17 February 2016 the Apache Software Foundation announced **Apache Arrow** as a top-level project: "a high-performance cross-system data layer for columnar in-memory analytics", "initially seeded by code from the Apache Drill project". The announcement claimed it "accelerates the performance of analytical workloads by more than 100x in some cases", a claim a team should test on its own workload.

The Arrow specification describes "a language-agnostic in-memory data structure specification", designed so that data is "relocatable without 'pointer swizzling', allowing for true zero-copy access in shared memory". The division of labor is simple to state. Parquet and ORC are how columns rest on disk, compressed and described by a footer. Arrow is how columns sit in memory while engines work on them and hand them to each other. A modern stack often reads Parquet into Arrow and never builds a row at all.

What a file promises, and what it does not

A columnar file makes real promises. It carries a schema, with names and types. Its types can be exact: Parquet's `DECIMAL` annotation "represents arbitrary-precision signed decimal numbers of the form unscaledValue * 10^(-scale)", with the precision and scale written in the footer, so a claim reserve of 48,000.00 is stored as the integer 4800000 with scale 2 and is read back exactly. It carries encodings that any conforming reader can decode, and statistics any reader can use. Those promises are why files travel well, and why the insurer's 2021 unload kept its amounts to the cent.

TYPE CONTRACT

Type, precision and rounding. Claim amount (paid or reserved) as DECIMAL(18,2): an unscaled integer times 10 to the power minus 2, exact to the cent. No rounding in the file; the loss ratio is rounded later, by the metric, half to even.

Encoding and serialization. Parquet logical type DECIMAL on the INT64 physical type (allowed up to precision 18); pages plain or dictionary encoded; footer minimum and maximum compared as signed values.

Unit or currency. US dollars. Not in the file: only the catalog entry and the column description say so.

Time basis. The amount as booked when the row was written: the 2021 migration unload, then nightly loads of claim events into raw tables, while the quarterly loss table the board figure reads is rebuilt every Friday (Chapter Eight). The file holds no valuation date; the folder date is the date of the unload, not of the loss or the reserve.

Identity scope. One row per claim transaction, keyed by claim number and transaction sequence within the insurer's claims system; business pooled from the affiliate is a separate population and must be flagged, not inferred.

Absence. Null means no amount recorded; zero means a recorded nil amount. The schema marks the column optional and says nothing about which is which.

Authority. The claims system of record sets the amount; the appointed actuary owns the reserve basis. The file is a copy and has no authority of its own.

Read the Type Contract for its gaps. Four of the seven lines are not in the file at all. And the format makes no promise about several things a CTO is tempted to assume:

- **No transactions.** Each specification describes one file. None of them says how several writers change a set of files together, so a reader of a folder can see half of a change. Transactions, where they exist, come from a layer above the files (Chapter Eight).
- **No catalog.** A file does not know which other files make up its table, which ones are current or which were replaced. A folder listing is not a table.
- **No grants.** Anyone who can read an unencrypted file can read every column in it. Parquet does define optional modular encryption, so that "each column and the footer can be encrypted with the same key, with a different key, or not encrypted at all", but the specification is "not limited to a single KMS, key generation/wrapping method, or authorization service". A key is not a grant; who may decrypt is decided elsewhere.
- **No meaning and no "as of".** The 2021 Parquet files typed the policy status correctly as an integer. Nothing in the format could say that 7 meant "perpetual, deposit held, no premium due", and the code table did not travel with them, which Chapter Seven follows into the board number. The format defines no field for any of the five times of a number either; its optional key and value metadata could carry them, but nothing requires a writer to fill it or a reader to honour it, and the insurer's files carried none.

Types, encodings and their failure modes deserve more than one section, and they are covered in depth later in this series. For this book the point is narrower: open formats made data easy to move, and they moved exactly what they describe.

OUR VIEW, AS OF NOVEMBER 2026

Parquet is likely to remain the common resting form of analytical data for years, because the major engines read it. That is a reason to rely on it for bytes and types, and a reason not to rely on it for anything else. The gaps listed above are where table formats, catalogs and policy engines compete, and that is where the next lock-in lives.

ERA CARD

Reader. The scan engine: software that reads a footer, prunes row groups and pulls a few columns from millions of rows, and, with Arrow, hands them to the next engine without conversion.

Security. Access is to files and folders, not columns; whoever reads the storage reads every column unless column keys are used and managed elsewhere.

Cost. Compression and pruning cut the bytes read, which on a platform billed by bytes scanned is the bill; a badly sorted or badly partitioned file pays for reading it all.

> **AI.** Training and feature pipelines read Parquet through Arrow at scale; the files bring exact types and no meaning, so a model learns from status 7 as a number.

Monday rule: pick the five Parquet or ORC tables your board numbers depend on and, for each money column, write down the unit, time basis and meaning of null that the file itself does not carry, in the catalog beside the column.

CHAPTER SEVEN

The Cloud Warehouse



On 1 May 2012 Google made BigQuery "publicly available", with the promise that users would "pay only for what you use". There was no machine to buy and no rack to fill. A team loaded its data, sent a query and received a bill that counted the work the query did. Behind the service ran Dremel, a query system Google had described at VLDB in 2010. By June 2015 Amazon Redshift and Snowflake had joined it, and the warehouse, which for twenty years had been a box on a capital budget, became a service on a monthly invoice. The insurer made that move in 2021, and its board number arrived in the cloud with one meaning missing. The decision this chapter changes: **when a migration rewrites the SQL behind a number, test that the number still means what its contract says, not only that the rows arrived**.

Three services and three bills

The three services that moved the warehouse into rented capacity arrived between 2012 and 2015.

- **Google BigQuery.** Publicly available on 1 May 2012. A 2020 paper by the Dremel team describes BigQuery as "a publicly available analytics service backed by Dremel" and as "a fully-managed, serverless data warehouse". The 2010 Dremel paper had shown "aggregation queries over trillion-row tables in seconds" by combining a columnar layout for nested data (Chapter Six) with multi-level execution trees.
- **Amazon Redshift.** A limited preview announced on 28 November 2012; available to all customers in AWS's What's New post of 14 February 2013 (the press release followed on 15 February), in AWS's words "a fully managed, petabyte-scale data warehouse service". At launch a customer chose a cluster of nodes, each with its own processors and disks.
- **Snowflake.** Introduced in October 2014 and generally available on 23 June 2015, by the company's announcement. Its design was published at SIGMOD 2016 as "The Snowflake Elastic Data Warehouse", by Benoit Dageville and colleagues.

Each of the three priced work in a different unit, and the unit decides who notices cost. In November 2026 the vendors' pages name these meters:

Service	Compute is metered in	Storage is metered in
BigQuery	On demand: the number of bytes processed by each query (per TiB). Capacity: slots, which Google's pricing page calls virtual CPUs (its slots page, virtual compute units), over time (per slot-hour)	Stored bytes over time (GiB-months): active or long-term, logical or physical
Redshift	Provisioned: nodes per hour. Serverless: Redshift Processing Units (RPU-hours), per second, with a 60-second minimum	Managed storage per GB-month
Snowflake	Credits consumed by a virtual warehouse while it runs, billed per second with a 60-second minimum each time it starts; on first-generation standard warehouses each size up doubles the credits per hour	Stored data, billed apart from compute and data transfer

This book prints no vendor prices. They change too often for paper and differ by region and contract; the dated web companion carries them. The cost figures in the Ledger rows are the invented insurer's own, labeled illustrative. Units last longer, and they matter more for design. A board query on BigQuery on demand costs what it scans, so a missing partition filter shows up on the invoice. The same query on a Snowflake warehouse costs the seconds the warehouse runs, so the expensive mistake is often a warehouse left running. Chapter Thirteen builds a cost model from these meters.

Storage apart from compute

The appliance of Chapter Four was **shared-nothing**: each node owned its disks and the rows on them. The Snowflake paper describes the design fairly, as one that "scales well for star-schema queries", and then names its cost: "it tightly couples compute resources and storage resources". A workload that needed more storage had to buy more processors, and a month-end peak of queries had to be sized into the hardware for the whole year. The paper adds a point about mixed work: a configuration "ideal for bulk loading (high I/O bandwidth,

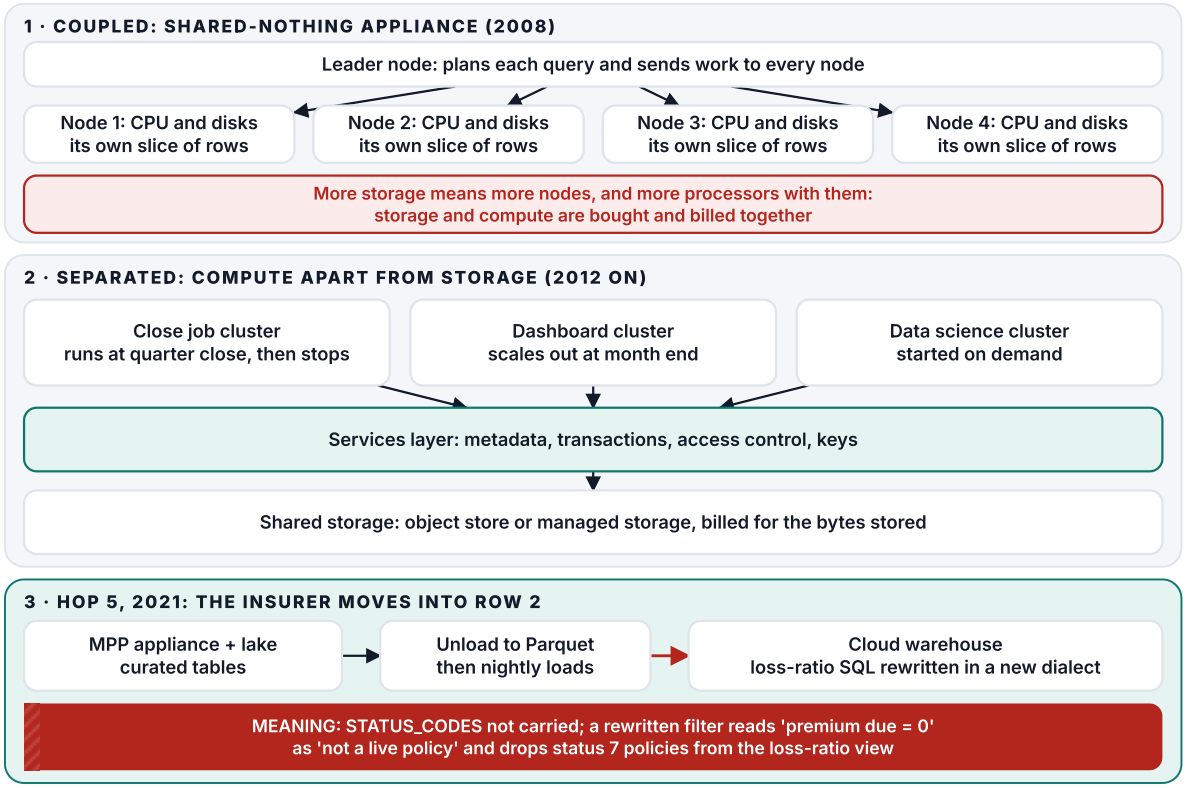
light compute) is a poor fit for complex queries (low I/O bandwidth, heavy compute) and vice versa". One rack had to serve the nightly load, the close job and the analysts, and each competed with the others for the same processors and disks.

Snowflake's answer, in the paper's words: "Snowflake separates storage and compute. The two aspects are handled by two loosely coupled, independently scalable services." In the system the paper describes, data sat in Amazon S3. Queries ran in "virtual warehouses", clusters of virtual machines a customer could start, stop and resize. A third layer, which the paper calls the system's "brain", held the metadata: "database schemas, access control information, encryption keys, usage statistics and so forth". Keep that third layer in mind. It is where meaning and control live, and it is the layer that does not travel when the files do.

BigQuery reached the same arrangement from inside Google. The 2020 Dremel paper recalls that Dremel "initially ran on a few hundred shared-nothing servers" and later moved to Google's shared file system, after which "Dremel on disaggregated storage outperformed the local-disk based system". Google's documentation now describes "a storage layer" and a compute layer that "operate independently of each other". Redshift made its move on 3 December 2019, when RA3 nodes with managed storage let customers, in AWS's words, "scale and pay for compute and storage independently". RA3 was Redshift's move, not the start of the idea.

FIGURE 7.1

Storage and compute: from one box to separate services, and where the insurer's hop 5 lost a meaning



In the appliance a leader node sends work to nodes that each own their disks; in the cloud warehouse pattern, drawn here in general form after Snowflake's published design and adopted by each service at a different date, compute runs apart from one storage layer and a services layer holds metadata, grants and keys; the insurer's 2021 move into that design rewrote the loss-ratio SQL.

Author's diagram based on Dageville et al., "The Snowflake Elastic Data Warehouse", SIGMOD 2016; Melnik et al., "Dremel: A Decade of Interactive SQL Analysis at Web Scale", PVLDB 2020; AWS, RA3 announcement, 3 December 2019; hop 5 from the thread (illustrative).

Separation changed three things for a CTO. Each workload can have its own compute: the close job, the month-end dashboards and the data scientists no longer queue behind each other. Keeping history no longer means keeping compute: storage is billed by the byte, apart from the engines that read it. And the provider can offer **serverless** compute, where the customer sets no cluster size at all. Google's documentation says BigQuery users "don't need to provision or manually scale resources"; Redshift Serverless, generally available on 12 July 2022, lets a customer "run and scale analytics without having to provision and manage data warehouse clusters". Serverless moves the sizing decision to the provider. It does not move the accountability for what a query means.

The 2020 Dremel paper is plain about where the idea came from: Dremel "is one of the pioneers in providing an elastic, multi-tenant, and on-demand service, now widely referred to in the industry as serverless", and the scheduling unit it introduced, the slot, continues "to be the core customer-visible concept of resource management in BigQuery". The unit a CFO reads on the invoice is a design decision made inside a query engine a decade earlier.

History inside the warehouse, for a while

Cheap storage brought one more change that matters to a board number: the warehouse began to keep its own past. The Snowflake paper lists "time travel and cloning" among the system's features, and the vendors now document the windows. Snowflake's standard retention "is 1 day (24 hours) and is automatically enabled for all Snowflake accounts"; on its Enterprise edition, permanent tables "can be set to any value from 0 up to 90 days". After the window, "the data is moved into Snowflake Fail-safe", where a customer can no longer query it. BigQuery keeps a time travel window "which covers the past seven days by default", configurable "from a minimum of two days to a maximum of seven days", and retains deleted data "for an additional seven days after the time travel window" as fail-safe. A query can read a table as it was, with Snowflake's `AT` or `BEFORE` clause or BigQuery's `FOR SYSTEM_TIME AS OF`.

Look at the length of those windows against the five times of the board number (Chapter One). The knowledge cutoff for Q4 2025 was 23 January 2026, the approval 5 February, the board meeting 19 February. A default window of one day, or seven, has closed long before anyone asks how a quarter's number was made. Default time travel in a cloud warehouse is sized for recovery and recent audit: to undo a bad load, or to compare today with last week. It is not a record of what the board saw. That record has to be made on purpose: a close table, a copy at the cutoff, or, in the lakehouse of the next chapter, a tagged snapshot with a stated retention.

On each platform. The windows, editions and syntax differ, and they change; the dated web companion lists the current values. What does not differ is that the default is short and that a longer window is a setting someone has to choose, on an edition that offers it, with the retained history kept as stored data.

> **WHAT HASN'T CHANGED**
>
> The 1999 cube of Chapter Three was rebuilt after each close and frozen, and its build time was the only "as of", printed nowhere. Two decades later the cloud warehouse could show a table as it was a few days ago, and still nobody printed the time beside the number. The tools for keeping time improved; the habit of writing it down did not come with them.

Data that does not move still has a hop

Once storage stood apart, the engine that reads it became a choice. Presto, built at Facebook, is described in its ICDE 2019 paper, "Presto: SQL on Everything" by Raghav Sethi and colleagues, as "an open source distributed query engine that supports much of the SQL analytics workload at Facebook". Its "Connector API allows plugins to provide a high performance I/O interface to dozens of data sources", and its "federated design" lets one query read "from many different data sources even within a single query". The paper notes that Amazon Athena is built on Presto. On 27 December 2020 the Presto creators' fork, PrestoSQL, was renamed Trino after a dispute over the Presto trademark, in their words "we are simply renaming".

An open engine over shared storage gives a company a second way to compute the same number, which is valuable at audit, in recovery and at exit (Chapter Fifteen). It also gives the company a second dialect. Two engines that read the same Parquet files can still disagree on how they round a value that falls exactly halfway (61.85 to one decimal, for example), how they cast a decimal, how they handle a decimal overflow or what a timestamp without a time zone means. The files travel; the semantics of the SQL over them belong to each engine. When a number moves to a new engine, its rounding rule and its filters have to move with it, written in the metric's definition rather than left to the engine's defaults.

BigQuery, Redshift and Snowflake each speak their own SQL dialect with its own functions and procedural extensions, and all three can now also read open files in object storage. What reads the files is a choice; what the SQL means is a contract each team has to write.

The latest vendor promise is that data need not be copied at all. On 7 November 2023 AWS made its Aurora MySQL zero-ETL integration with Redshift generally available: "Within seconds of transactional data being written into Aurora, the data is available in Amazon Redshift", and "you don't need to build and maintain complex data pipelines". BigQuery's federated queries let a team "send a query statement to an external database" with the `EXTERNAL_QUERY` function and "get the result back as a temporary table".

One paragraph further, the same documentation says that for a federated query BigQuery must "temporarily move data from the external data source to BigQuery", and that "the data from the external data source is converted to GoogleSQL types". So there is still a copy, still a type conversion and still a moment when the data was read. A zero-ETL pipeline is a pipeline the provider runs: it has a lag, a mapping of types, a list of tables included and an owner who can change it. None of that is bad. It is a hop, and it belongs in the Ledger like any other: what moved, when, in which types, and what was assumed. "No pipeline" means no pipeline *you* wrote, not no pipeline.

OUR VIEW, AS OF NOVEMBER 2026

Zero-ETL and federation are good engineering for freshness. The risk is in the word "zero": teams stop recording hops they did not build. A Ledger row for a provider-run integration takes ten minutes and names the lag and the type mapping a board number depends on.

The hop: the board number reaches the cloud

In 2021 the insurer ran one cloud programme. One stream of it moved the policy domain from DB2 for z/OS into the cloud warehouse, the move told in *From Tally to Token*, the first book of this series, where the STATUS_CODES lookup table was left off the list of tables to carry. Another stream moved the claims and premium tables behind the board number: from the MPP appliance of 2008 and from the curated tables of the Hadoop lake, unloaded to Parquet and loaded into the cloud warehouse, then kept current by nightly loads. Decimal types survived exactly. The appliance was retired. The lake kept running, because adjusters' notes and photos still landed there, and it would run until 2024.

The loss-ratio SQL had to be rewritten in the new dialect, and the rewrite is where the board number lost a meaning. The old appliance view joined the policy table, loaded from DB2, to its status codes. In the warehouse, status was a bare integer and the code table was not there, because the policy stream had left it behind. A developer looking at the policy rows saw some with no premium due and wrote a filter that read "premium due = 0" as "not a live policy". Status 7 policies, perpetual policies with the deposit held, owe no annual premium and are in force; the metric contract approved in 2019 includes them. The new filter dropped them entirely, and every loss on them, in every quarter since. The reconciliation compared row counts of the loaded base tables and total earned premium between the old system and the new; the base tables were complete, and status 7 policies have no earned premium, so both checks passed. The side-by-side run of the loss ratio itself showed a gap of a few tenths of a point, which the team put down to the two systems loading on different nights. Nobody tested the view against the contract's population clause, because the contract was a document in the audit committee's papers and the test was a script in the migration team's repository. The board view has carried that filter since 2021. In 2024 the view moved, filter and all, onto the lakehouse platform's SQL warehouse engine, which has computed the board figure since then; Chapter Eight measures the filter's effect.

THE HOP · 5 · 2021 · MPP APPLIANCE AND THE HADOOP LAKE'S CURATED TABLES TO CLOUD DATA WAREHOUSE (LOADED VIA PARQUET), IN THE SAME PROGRAMME THAT MOVED THE POLICY DOMAIN FROM DB2 FOR Z/OS

Record	Property loss ratio, Q4 board figure (dataset), hop 5, 2021
From	MPP appliance and the Hadoop lake's curated tables
To	Cloud data warehouse (loaded via Parquet), in the same programme that moved the policy domain from DB2 for z/OS
Mechanism and schedule	Bulk migration plus nightly loads; loss-ratio SQL rewritten in the new dialect; the appliance retired; the lake keeps running for claim files
Format change; what was lost or assumed	MEANING Appliance tables unloaded to Parquet and loaded to warehouse tables; DECIMAL types kept; status as a bare integer. **The STATUS_CODES lookup table is not carried (as in From Tally to Token); a rewritten filter reads 'premium due = 0' as 'not a live policy' and drops status 7 policies from the loss-ratio view**
Accountable owner	Head of data platform
Who can change it	Pipeline service account; data engineers; the warehouse load role
Classification; retention or erasure duty	Financial and personal data; US federal (GLBA) and state insurance privacy rules; retention duties
Read by (people, apps, models, agents)	Finance and actuarial dashboards; the board pack; data scientists
Cost per month; basis	Warehouse compute and storage for this domain, about 3,400 dollars a month (illustrative)

ERA CARD

Reader. Analysts and dashboards on rented compute, each workload with its own cluster or none at all; the engine became a choice and the dialect came with it.

Security. Grants, policies and keys live in the provider's services layer, beside the metadata, not in the files; the files can move without them.

Cost. Compute is metered by bytes scanned, slot-hours, credits or RPU-hours, storage by the byte; the unit decides which mistake is expensive.

AI. A model that writes SQL for a cloud warehouse inherits the dialect and the views as they are, filters included; it cannot see a meaning the migration left behind.

Monday rule: for every board number that a migration rewrote into a new dialect, run the new view against the written metric contract, clause by clause, and record in the Ledger which filters and rounding rules the rewrite introduced.

CHAPTER EIGHT

Tables, Snapshots and the Lakehouse



In 2020 the engineers who built Delta Lake at Databricks published a candid number. In the first years of their cloud service, 2014 to 2016, "around half the support escalations we received were due to data corruption, consistency or performance issues due to cloud storage strategies". Files in object storage were cheap and open, but a folder of files could not promise what a database table promises: that a write is all or nothing, and that a reader sees one consistent state. Table formats fixed that by keeping a record of every committed state of a table, a **snapshot**. A snapshot is also an exact "as of" for every number computed from it. The insurer adopted snapshots in 2024 and recorded none of them with its board figure. The decision this chapter changes: **a board number is recorded with the snapshot it was computed from, and that snapshot is tagged and kept**.

A folder is not a table

The Hive tables of Chapter Five were directories. The Delta Lake paper (Armbrust and colleagues, VLDB 2020) describes the pattern: a table stored "as a collection of objects", often split into directories by a column, such as `mytable/date=2020-01-01/`. The table is "just a bunch of objects" that any tool can read. The paper then lists what that costs on object storage. Listing objects is slow: one listing call "only returns up to 1000 keys", so "it can take minutes to list a dataset with millions of objects". And "because multi-object updates are not atomic, there is no isolation between queries": a reader can see half of an update, and a crashed job leaves the table in a state nobody chose. Hive's own transactional tables relied on the Hive metastore to track each table's state, which, the paper says, "can create a bottleneck in tables with millions of partitions".

Three projects answered with the same idea, a metadata layer that lists exactly which files make up the table at each commit:

- **Apache Iceberg** started at Netflix, entered the Apache Incubator on 16 November 2018 and graduated on 20 May 2020. Its specification calls it a format "designed to manage a large, slow-changing collection of files in a distributed file system or key-value store as a table".
- **Delta Lake** was offered to Databricks customers from 2017 and open sourced on 24 April 2019. It keeps a transaction log, "compacted into Apache Parquet format to provide ACID properties, time travel, and significantly faster metadata operations".
- **Apache Hudi** (Hadoop Upserts Deletes and Incrementals) was developed at Uber in 2016 and open sourced in 2017; Uber's engineering blog introduced it on 11 March 2017 as "an analytical, scan-optimized data storage abstraction" that applies changes to data in HDFS within minutes. It entered the Apache Incubator in January 2019 and was announced as a top-level project on 4 June 2020.

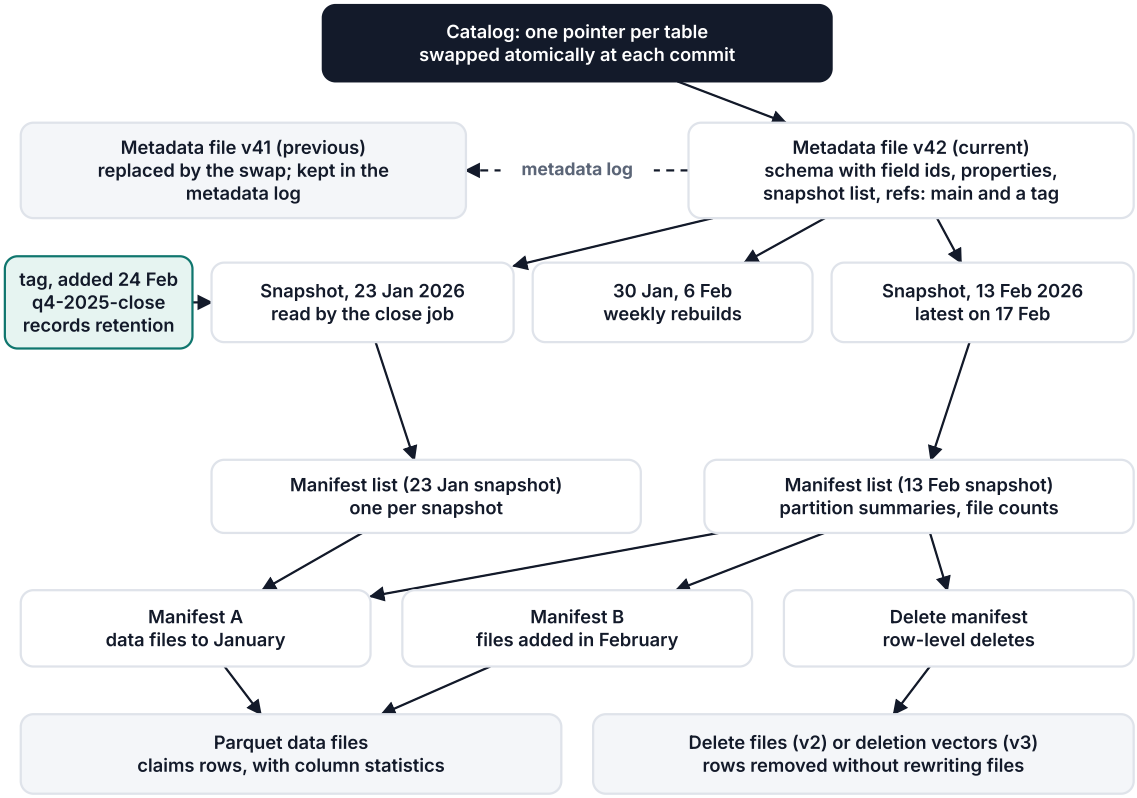
All three bring **ACID** transactions (atomic, consistent, isolated, durable) to files on object storage, one table at a time; their isolation levels and rules for concurrent writers differ in detail, by format and by engine. The rest of this chapter uses Iceberg's specification, because the insurer's lakehouse uses Iceberg and because the specification is public and precise; Delta Lake and Hudi make the same promises with different files.

Inside an Iceberg table

Figure 8.1 shows the layers. A **catalog** holds one pointer per table, to the current **metadata file**. The specification: "All changes to table state create a new metadata file and replace the old metadata with an atomic swap." The metadata file holds the schema, the partitioning, the table properties and the list of snapshots. "A snapshot represents the state of a table at some time." Each snapshot points to a **manifest list**, which points to **manifests**, which list the data files with their statistics. Since version 2 of the format, manifests can also list **delete files** that mark rows removed from existing data files "without rewriting the files"; version 3 adds binary deletion vectors. The specification states that versions 1 to 3 "are complete and adopted by the community" and that version 4, which restructures the metadata tree, "is under active development and has not been formally adopted" (read on 20 November 2026).

FIGURE 8.1

An Iceberg table: one catalog pointer, a tree of metadata, and snapshots that a tag can keep



The catalog points to the current metadata file, which logs the previous one; each weekly snapshot of the quarterly loss table reaches its data and delete files through a manifest list and manifests, reused by later snapshots where files did not change; the tag q4-2025-close, added on 24 February 2026 (on 17 February there was none), keeps the close snapshot from expiry.

Author's diagram based on the Apache Iceberg Table Spec (overview, snapshots, manifest lists, snapshot references, delete formats), read 20 November 2026; dates from the thread (illustrative).

Two details matter for a board number. First, columns are tracked by **field id**, not by name, so a renamed column keeps its identity; *From Tally to Token* showed what that does and does not protect. Second, a snapshot can carry a name. "Tags are labels for individual snapshots." Branches are named lines of snapshots that can move. Each reference has its own retention: a tag can be told to keep its snapshot for a stated time, or for ever. A tag called `q4-2025-close` on the snapshot a close job read is the whole difference between "the claims table" and "the claims table as the board saw it".

Time travel, late data and the cost of keeping history

Time travel means reading a table as it was. Iceberg's specification illustrates it with the query `SELECT * FROM prod.db.table TIMESTAMP AS OF '1986-10-26 01:21:00Z'`, which finds the snapshot current just before that instant; the SQL itself belongs to each engine (Iceberg's Spark documentation gives the same form), and engines also read by snapshot id or by tag. Note what the timestamp is: the time a snapshot was committed, the platform's own clock. It is system time in the language of Chapter One. It is not the knowledge cutoff of a number. Claims received by 23 January may be loaded on the 24th; a correction to December may be committed in February. The exact way back to the state a close job read is that job's snapshot id, recorded when it ran.

Late data makes this sharper. Claims arrive after the quarter, reserves are re-estimated, and change data capture (CDC) streams updates from a source system into a table as row-level changes, each commit a new snapshot. That is what a lakehouse is for: the table holds today's best estimate of every past quarter. It is also why "the latest snapshot" is a different question from "what was booked at the cutoff".

History has a cost, so tables need maintenance. Iceberg's maintenance documentation lists eleven operations; three matter here. **Compaction** combines small files "into larger files to reduce metadata overhead and runtime file open cost". **Snapshot expiry**: "Snapshots accumulate until they are expired by the expireSnapshots operation", and expired snapshots "are no longer available for time travel queries". **Orphan file removal** deletes files no snapshot references, with a default safety interval of three days. The expiry defaults are table properties, read on 20 November 2026 for Iceberg 1.12.0:

Table property	Default	Meaning when expiry runs
`history.expire.max-snapshot-age-ms`	432000000 (5 days)	Snapshots older than this may be expired
`history.expire.min-snapshots-to-keep`	1	At least this many snapshots stay on each branch
`history.expire.max-ref-age-ms`	Long.MAX_VALUE (forever)	How long tags and branches other than main are kept; "The main branch never expires"

Read the defaults together. On an untagged table, an expiry run keeps the current snapshot and anything younger than five days, and nothing else. Platforms often raise the age: a longer setting delays the loss of a quarter-close snapshot but does not prevent it. A snapshot held by a tag survives for the tag's retention, which by default is for ever. Expiry is not optional housekeeping either: until every snapshot that references a file has expired and the

file itself is removed, a deletion that must reach retained versions is not physically complete. Retention is therefore a decision with two owners, the CFO for the quarter-close snapshots and the privacy officer for the deletions, and it belongs in the metric passport. The two duties meet on the tag: a quarter-close tag kept for the statutory records period keeps the claimant rows in its files for that long, so the retention schedule has to record why that period is allowed and who approved it.

One copy, several formats

In 2021 Armbrust, Ghodsi, Xin and Zaharia defined the **lakehouse** at CIDR as "a data management system based on low-cost and directly-accessible storage that also provides traditional analytical DBMS management and performance features such as ACID transactions, data versioning, auditing, indexing, caching, and query optimization". The promise was one governed copy of each table, read by the warehouse engine, the notebooks and the models.

Then the formats themselves had to meet. Delta Lake 3.0, announced in June 2023 and released on 17 October 2023, introduced UniForm, which writes Iceberg metadata beside Delta's (Hudi followed, in preview, with Delta Lake 3.2 in May 2024), so that "a single copy of the data files supports both Delta and Iceberg clients", in the words of Databricks' current documentation. Apache XTable, which entered the Apache Incubator on 11 February 2024 and was still incubating when checked on 20 November 2026, "reads the existing metadata of your table and writes out metadata for one or more other table formats".

Interoperability makes the files readable. It does not make the readings equal. Databricks states that for UniForm "Iceberg client support is read-only" and that metadata is generated "asynchronously after a Delta Lake write transaction completes", so an Iceberg reader can be a commit behind. XTable's site lists among its current limitations that, when it commits the metadata of a secondary format, "the timestamp of the commit will not be the exact same timestamp as shown in the primary", and it does not support merge-on-read tables. For a board number, the timestamp warning matters most: the same commit can carry two different "as of" times in two formats. And no format carries the grants, the row policies, the masking rules or the metric definitions; those live in the catalog and the semantic layer (Chapter Ten).

The hop: two snapshots, two numbers

In 2024 the insurer moved the claims and premium tables, and the lake's claim files, into a lakehouse: Iceberg tables on object storage, a vendor-managed catalog with an Iceberg REST interface, and the vendor platform's two engines, a SQL warehouse engine and a notebook engine. From then on the cloud warehouse that computed the board figure was that SQL

warehouse engine, and the 2021 view moved onto it, filter and all. Claim events load nightly, and by stream, into raw claim-event tables. The quarterly loss table (claims and reserves) that both engines read is rebuilt by the reserving batch and committed every Friday, so it gains one snapshot a week. The insurer defines the knowledge cutoff as that commit on the cutoff day: anything loaded later is not known. The close job runs on the SQL warehouse engine, reads the Iceberg tables through the vendor catalog, so the catalog's row access policies apply to it, and writes the close table the dashboard shows. The actuaries' notebooks read the same tables through the same catalog, at the latest snapshot. Nobody recorded a snapshot id, a tag or an "as of" with the board figure, and nobody set a retention for quarter-close snapshots. The expiry run, every Sunday, kept 45 days of snapshots, a setting the platform team had raised from Iceberg's five-day default in 2024 for audit reads; snapshots of earlier quarter closes had already gone. The snapshot committed at the knowledge cutoff, Friday 23 January 2026, was the one the close job read on Monday 26 January. The job's log kept its run time but not the snapshot id; because the table commits once a week, the snapshot log showed one candidate before that run. In February it was still in the table only because it was younger than 45 days, and the Sunday run of 15 March would have removed it. Nothing protected it until Tuesday 24 February, five days after the board meeting, when the platform team created the tag `q4-2025-close` on it with retention set to the statutory records period.

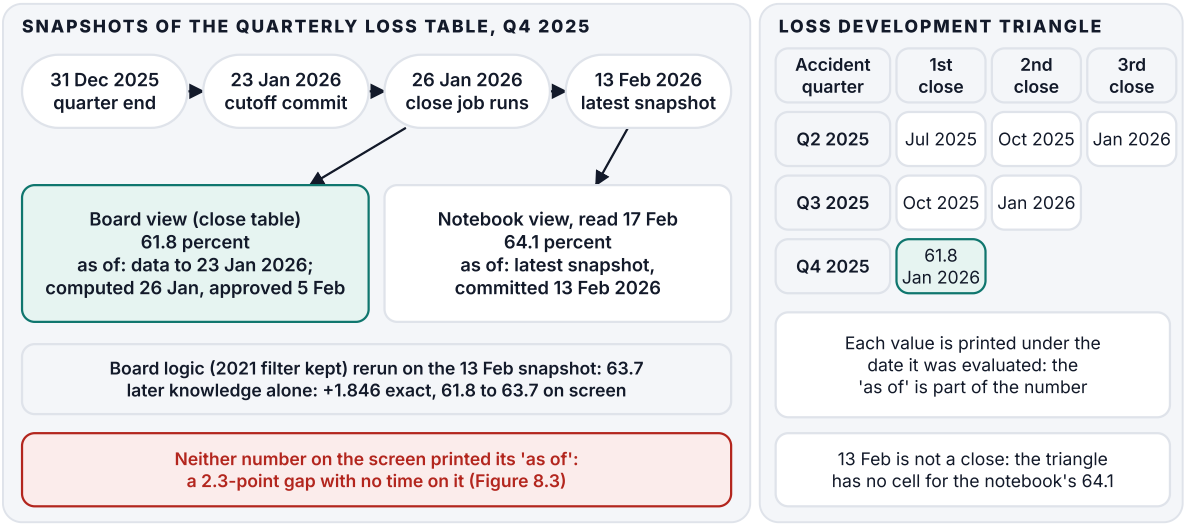
THE HOP · 6 · 2024 · CLOUD WAREHOUSE TABLES AND THE LAKE'S CLAIM FILES TO LAKEHOUSE: ICEBERG TABLES ON OBJECT STORAGE, A VENDOR-MANAGED CATALOG WITH AN ICEBERG REST INTERFACE, TWO ENGINES

Record	Property loss ratio, Q4 board figure (dataset), hop 6, 2024
From	Cloud warehouse tables and the lake's claim files
To	Lakehouse: Iceberg tables on object storage, a vendor-managed catalog with an Iceberg REST interface, two engines
Mechanism and schedule	Tables rewritten as Iceberg under a vendor-managed catalog; the platform's SQL warehouse engine runs the close job and writes the dashboard's close table; the notebook engine reads the latest snapshot; the reserving batch commits the quarterly loss table every Friday
Format change; what was lost or assumed	TIME Parquet data files, Iceberg metadata, manifests and snapshots; row and column policies held in the vendor catalog. **Numbers are computed at read time from the latest snapshot; no snapshot id, tag or 'as of' is kept with the board figure, and quarter-close snapshots are removed by the weekly expiry run once older than the platform's 45-day setting**
Accountable owner	Head of data platform (platform); CFO (the metric)
Who can change it	Platform team; write jobs; catalog administrators (policies and grants)
Classification; retention or erasure duty	Financial and personal data; GLBA and state privacy rules; policies enforced through the vendor catalog
Read by (people, apps, models, agents)	Finance dashboards, actuaries' notebooks, pricing models, an assistant over the semantic layer
Cost per month; basis	Storage, two engines and catalog for this domain, about 3,100 dollars a month (illustrative)

The Q4 2025 board pack showed 61.8 percent from the close table (computed 26 January on the SQL warehouse engine, approved 5 February 2026, data to 23 January). On Tuesday 17 February the CFO's analyst read the same quarter in a notebook from the latest snapshot of the quarterly loss table, the Friday rebuild committed on 13 February, and brought 64.1 percent. A rerun on Thursday 19 February, the day of the board meeting, would still have returned 64.1: the next commit came on Friday 20 February. Figure 8.2 puts the two readings on one time line, beside the actuaries' own way of keeping time: a loss development triangle, where every value is printed under the date at which it was evaluated.

FIGURE 8.2

Two snapshots, two numbers: the notebook read a later state of the table than the board, and printed no date



On 26 January the close job on the platform's SQL warehouse engine read the snapshot committed at the cutoff (23 January 2026) and wrote the board figure to the close table; the notebook read the latest weekly snapshot (13 February 2026); a loss development triangle, by contrast, prints every value under its evaluation date.

Author's diagram; numbers and dates from the thread (illustrative); triangle drawn after the accident-year layout of Schedule P described in Chapter One, here by accident quarter and as loss ratios (illustrative).

Figure 8.3 walks the gap one effect at a time: 2.3 points between the two figures on the screen, 2.250 points exact.

FIGURE 8.3

The 2.3-point bridge: from the board figure to the notebook figure, one effect at a time

Step	As of	Shown	Exact	Shown difference	Exact effect	Loss code
Board figure as approved (contract rounding; 2021 population filter applied)	Data to 23 Jan 2026; approved 5 Feb	61.8	61.850			
+ later knowledge	Latest snapshot, 13 Feb 2026	63.7	63.696	+1.9	+1.846	TIME
+ status 7 losses kept (population)	13 Feb 2026	64.0	64.004	+0.3	+0.308	MEANING
+ anticipated salvage and subrogation not netted (definition)	13 Feb 2026	64.1	64.100	+0.1	+0.096	MEANING
Lakehouse notebook figure (total gap)	13 Feb 2026, read 17 Feb	64.1	64.100	+2.3	+2.250	

Effects measured in this order: later knowledge, then population, then definition (they interact, so another order gives other sizes). The exact gap is 2.250; the 2.3 on the screen is 64.1 minus 61.8, the difference of two rounded figures, and the printed steps are differences between shown figures, not rounded exact effects. The status 7 losses (770) and the anticipated salvage and subrogation (240) did not develop between 23 January and 13 February (illustrative), so the same amounts serve this bridge and the restatement.

Source: the thread's illustrative Q4 2025 rows (thousands of dollars; net premiums earned 250,000); the reader can rebuild every line from the table in Appendix A.

- **Later knowledge, +1.846 exact (TIME); on screen, 61.8 becomes 63.7.** The 13 February snapshot holds development booked after the cutoff: the January reserve re-estimates, and case reserves on storm claims reported after 23 January, set above the IBNR allowance held for them. This is not an error. It answers "today's estimate for Q4", not "what was booked for Q4 at the board cutoff".
- **Population, +0.308 exact (MEANING).** The board view carries the 2021 filter of Chapter Seven, which drops status 7 policies. The notebook query, which the actuaries wrote against the lakehouse tables without the 2021 view, keeps them, so their Q4 fire losses enter the numerator while their earned premium is zero. The contract includes them. On this step the lakehouse is right, and the board view has been wrong since 2021.

- **Definition, +0.096 exact (MEANING).** The notebook query takes losses before anticipated salvage and subrogation; the contract nets them.

The verdict is uncomfortable for both sides. Both numbers are reproducible: the board figure from the close table and the 23 January snapshot, the notebook figure from the 13 February snapshot. Neither is the figure the board's own metric contract asks for. 64.1 uses the wrong knowledge time and the wrong definition. 61.8 uses the right time and the right definition and the wrong population. The insurer's correction, and the figure the contract gives when it is recomputed at the original cutoff, belong to Chapter Seventeen. Two small lessons travel with them: add the unrounded parts and round once, and keep the snapshot the number came from.

> *The table still held the state the close job had read. Nobody had written down which snapshot it was.*

ERA CARD

Reader. Warehouse engines, notebooks, models and assistants reading one copy of open tables, each at whatever snapshot it happens to load.

Security. Grants, row policies and masks sit in the catalog, not in the files or the table format; expiry is also how a required deletion becomes physical.

Cost. Storage, compute per engine and maintenance runs; a retained snapshot keeps the files it still references, so retention is a cost decision as well as an audit one.

AI. An assistant over the lakehouse reads the latest snapshot unless told otherwise, so it quotes today's estimate as if it were the booked figure.

Monday rule: for each board number, record the snapshot id its close job read, put a named tag with a stated retention on that snapshot, and check that your expiry job cannot remove it.

Part II ends here. The Data Lineage Ledger below gathers the first seven hops of the board number, from the 1987 printout to the 2024 lakehouse, with what each hop lost; the eighth, the exit test, belongs to Chapter Fifteen, and Appendix A holds all eight.

THE LEDGER, PART 1 OF 2

One Number, Since 1987: hops 0, 1, 2, 3, 4, 5, 6: movement and meaning

Hop, year	From	To	Mechanism and schedule	Format change; what was lost or assumed
0 1987	IMS claims database and the IMS policy master	Monthly loss report, printed on green-bar paper by a COBOL report program	Monthly batch job after month end; the report is read in the actuaries' room	CONTEXT Fixed-width print lines, 132 columns; totals by line of business. **Reserve adjustments written by hand on the printout and kept in the actuaries' binder; the printed figure and the used figure differ**
1 1994	IMS extracts (claims, policy)	Enterprise data warehouse on DB2 (a separate analytic copy)	Monthly extract programs, transfer and load jobs; subject areas for claims and premium	MEANING Fixed-width extracts loaded to normalized DB2 tables with history. **Incurred loss is defined once in the warehouse, but the reserve adjustment rule stays in the actuaries' spreadsheet, so the warehouse figure still needs a manual step**
2 1999	Enterprise data warehouse	Claims star schema and a multidimensional cube for the board pack	Cube rebuilt after each quarter close; the board pack reads the cube	TIME Fact table of claim transactions, dimensions for line, state, accident period; pre-aggregated cube cells. **The cube's build time is the only 'as of', and it is printed nowhere; later corrections reach the warehouse but not the frozen cube**
3 2008	DB2 enterprise data warehouse	MPP warehouse appliance (shared-nothing, one rack)	One-time migration: unload, convert, reload; load jobs and stored procedures rewritten for the appliance	COMPLETENESS DB2 tables to appliance tables distributed on claim number. **History before 1995 not migrated; kept on tape in the old load format, with a record layout no current job can read**
4 2014	Appliance extracts and raw claim files (adjuster notes, photos, third-party reports)	Hadoop data lake beside the appliance (HDFS, Hive tables stored as ORC)	Daily file drops into a raw zone; Hive tables defined over the files; nightly jobs	OWNERSHIP Text, images and PDFs as files; tabular extracts as ORC; schema applied when read. **Adjuster notes and photos land in a raw zone with no accountable owner and no retention rule; schema on read means each team reads them differently**
5 2021	MPP appliance and the Hadoop lake's curated tables	Cloud data warehouse (loaded via Parquet), in the same programme that moved the policy domain from DB2 for z/OS	Bulk migration plus nightly loads; loss-ratio SQL rewritten in the new dialect; the appliance retired; the lake keeps running for claim files	MEANING Appliance tables unloaded to Parquet and loaded to warehouse tables; DECIMAL types kept; status as a bare integer. **The STATUS_CODES lookup table is not carried (as in From Tally to Token); a rewritten filter reads 'premium due = 0' as 'not a live policy' and drops status 7 policies from the loss-ratio view**

Hop, year	From	To	Mechanism and schedule	Format change; what was lost or assumed
6 2024	Cloud warehouse tables and the lake's claim files	Lakehouse: Iceberg tables on object storage, a vendor-managed catalog with an Iceberg REST interface, two engines	Tables rewritten as Iceberg under a vendor-managed catalog; the platform's SQL warehouse engine runs the close job and writes the dashboard's close table; the notebook engine reads the latest snapshot; the reserving batch commits the quarterly loss table every Friday	TIME Parquet data files, Iceberg metadata, manifests and snapshots; row and column policies held in the vendor catalog. **Numbers are computed at read time from the latest snapshot; no snapshot id, tag or 'as of' is kept with the board figure, and quarter-close snapshots are removed by the weekly expiry run once older than the platform's 45-day setting**

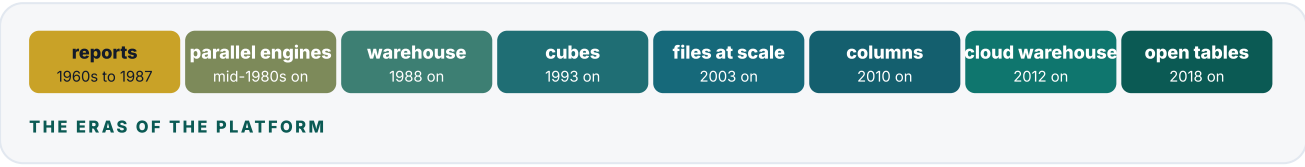
THE LEDGER, PART 2 OF 2

One Number, Since 1987: hops 0, 1, 2, 3, 4, 5, 6: control, readers and cost

Hop	Accountable owner	Who can change it	Classification; retention or erasure	Read by	Cost per month; basis
0	Controller	Claims systems programmers (the report program); actuaries (the hand adjustments)	Financial record; kept with the statutory working papers	Actuaries, the controller, the board (a typed summary)	Machine time on the shared mainframe plus about two actuary days a month for the hand adjustments (illustrative)
1	Director of management information	Warehouse load jobs; two warehouse developers	Financial and personal data (claimants); retention per statutory records rules	Finance analysts, actuaries, report writers	Share of mainframe DB2 capacity and two developers (illustrative)
2	Director of management information	BI team (cube design and rebuild job)	Aggregated financial data; no claimant names in the cube	CFO, board, actuaries (drill to line and state, not to claim)	OLAP server licence and a small BI team (illustrative)
3	Head of data warehousing	Appliance DBAs (vendor-trained); ETL team	Financial and personal data; retention per statutory records rules	Finance, actuaries, the cube, pricing analysts	Appliance purchase and annual support contract (illustrative)
4	None recorded for the raw zone; head of analytics for curated tables	Anyone with write access to the raw zone; the Hadoop platform team	Personal and sensitive data (claimant photos, notes) not yet classified	Data scientists, fraud analytics, the pricing team	Cluster hardware, support subscription and a platform team (illustrative)
5	Head of data platform	Pipeline service account; data engineers; the warehouse load role	Financial and personal data; US federal (GLBA) and state insurance privacy rules; retention duties	Finance and actuarial dashboards; the board pack; data scientists	Warehouse compute and storage for this domain, about 3,400 dollars a month (illustrative)
6	Head of data platform (platform); CFO (the metric)	Platform team; write jobs; catalog administrators (policies and grants)	Financial and personal data; GLBA and state privacy rules; policies enforced through the vendor catalog	Finance dashboards, actuaries' notebooks, pricing models, an assistant over the semantic layer	Storage, two engines and catalog for this domain, about 3,100 dollars a month (illustrative)

CHAPTER NINE

Anatomy of a Platform



On 3 June 2024 Snowflake announced Polaris Catalog, "a vendor-neutral, open catalog implementation for Apache Iceberg", and promised to open source it within 90 days. Ten days later, on 13 June 2024, Databricks published "Open Sourcing Unity Catalog" and placed its catalog's server code with LF AI and Data, a Linux Foundation umbrella. Two companies that compete for the same budgets released, in the same fortnight, the layer that says which tables exist and who may read them. That fortnight is a good place to start a chapter about platforms, because it shows where the contest had moved. Files were already open. The fight was over the layers above them. The decision this chapter changes: **before you compare platforms, write down who holds authority at each layer, because that is what you will have to rebuild if you leave**.

Eight layers, one question per layer

Part II followed the engines one era at a time. Each era bundled some layers and unbundled others. The appliance of 2008 sold storage, compute, metadata and security in one rack. The cloud warehouse separated storage from compute but kept the table format, the catalog and the policies inside one service. The lakehouse opened the files and the table format and left the rest to whoever runs the catalog. To compare any of these, you need one model that fits all of them.

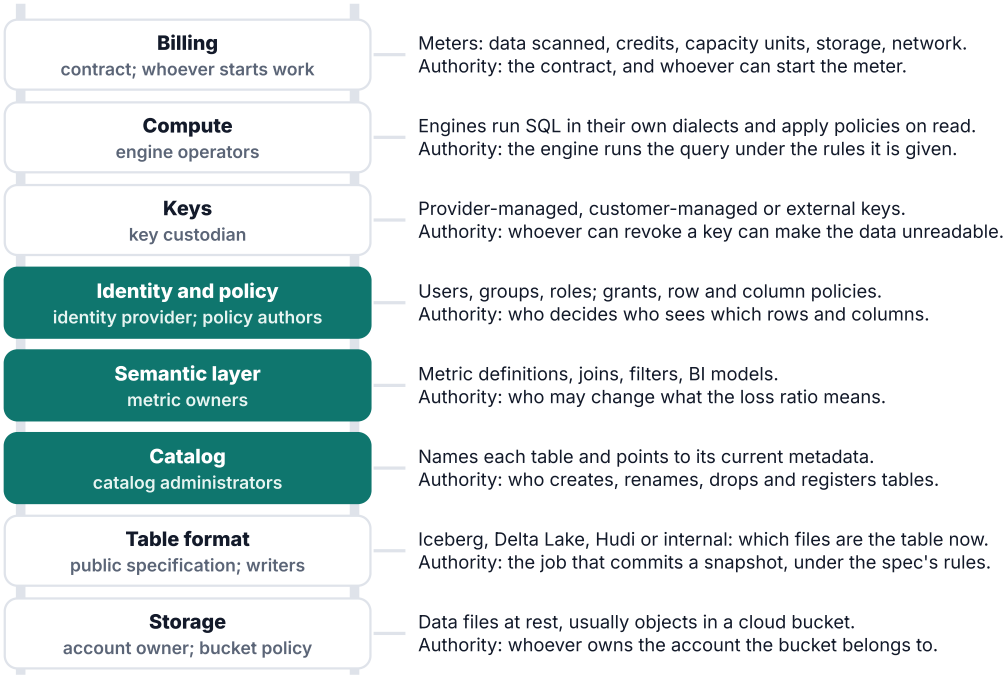
This book uses eight layers, from the bottom up:

- **Storage**: the bytes at rest, today usually objects in a cloud bucket. Question: whose account holds them?
- **Table format**: the metadata that turns a folder of files into a table with versions (Iceberg, Delta Lake, Hudi, or a vendor's internal format). Question: is the specification public, and who commits changes?

- **Catalog**: the register that names each table and points to its current metadata. Question: who can create, rename, drop and register tables, and through which interface can another engine find them?
- **Semantic layer**: the definitions of business measures, the joins and filters a BI tool or an assistant uses. Question: who may change what "loss ratio" means?
- **Identity and policy**: users, groups, roles, and the grants and row and column policies attached to them. Question: who decides who sees which rows?
- **Keys**: the encryption keys that protect the data. Question: who can revoke them?
- **Compute**: the engines that run queries, each with its own SQL dialect. Question: which engine applies the policies when it reads?
- **Billing**: the meters that turn work into an invoice. Question: who can start the meter, and what does it count?

FIGURE 9.1

Anatomy of a platform: eight layers, and who holds authority at each



Storage and open table formats travel with a coordinated copy of data and metadata files, registered again in a catalog and with paths rewritten if addresses change; the shaded middle rungs, where meaning and control live, stay with whoever runs them, and one layer can be held by more than one party.

Author's diagram; the layer model is the author's, informed by the Apache Iceberg Table Spec (catalog role) and Armbrust et al., CIDR 2021.

Figure 9.1 draws the layers as a ladder. The bottom two rungs are now largely open: a Parquet file and an Iceberg table can be read by many engines. The three shaded rungs in the middle (catalog, semantic layer, identity and policy) are where meaning and control live, and they are the rungs a file copy does not carry. Chapter Ten looks at them one by one; Chapter Eleven looks at identity, keys and recovery.

Authority, not features

A feature list tells you what a platform can do. An authority map tells you who decides. The difference matters on the day something goes wrong, and on the day you leave.

Take the insurer's board figure as it stood in February 2026: 61.8 percent (approved 5 February, data to 23 January). Walk it up the ladder. The claims and premium files sat in the insurer's own cloud account: storage authority stayed with the insurer. The tables were Iceberg, an open specification: table state changed whenever a job allowed to commit a snapshot did so, under the platform team's authority. The catalog was vendor-managed with an Iceberg REST interface, so another engine could find the tables, but the row and column policies were held in that vendor catalog and nowhere else. The metric's definition was written three times: in the 2021 view, which the platform's SQL warehouse engine ran at each close; in a semantic model for the dashboard; and in the analysts' own notebook query. Two engines read the tables through one catalog, each with its own dialect and its own meter.

Seen this way, the analyst's 64.1 in the prologue is not a mystery about engines. Part of the gap is a definition written three times and never checked against the contract; the rest is a gap in authority. No layer bound the board number to the snapshot it came from: the table format could have kept a tagged snapshot, the metric passport could have recorded its id, the semantic layer could have pinned it, and none of them did. The close table kept the result, not the state it was computed from. Chapter Eight showed what that cost.

An authority map also shows concentration. When one vendor holds the catalog, the policies, the semantic layer and the compute, the open files below matter less than they seem. The Iceberg specification, describing its version 4, says a table's location is "intended to be managed and supplied by a catalog". Whoever supplies it decides which engine sees which table, under which rules. That is not a defect; it is a choice, and it should be written down as one.

OUR VIEW, AS OF NOVEMBER 2026

Most platform reviews still start with compute speed. Start instead with the three middle rungs. Speed is easy to buy again later; a decade of definitions and policies is not.

The platform card

To compare platforms on authority, this book uses one short form, the **platform card**. It has ten rows, always in the same order, so that two cards can be laid side by side and read row by row. Chapter Twelve fills in seven cards, dated November 2026. The schema is here:

Row	What it records
Scope and date	Which offering, which region and edition, and the date the facts were checked; a card without a date is an advertisement.
Storage of record	Where the bytes live and in whose account, so you know whether the data sits in your bucket or the vendor's.
Table formats (native, read, write)	The format the platform keeps internally, the formats it can read, and the formats it can write so that another engine can read them back.
Catalog and its interface	Which catalog names the tables, and whether another engine can reach it through an open interface such as the Iceberg REST catalog.
Semantic layer	Where metric definitions, joins and BI models live, and in what language they are written.
Where policies are enforced	Which component applies grants and row and column policies, to which identities, and whether they still apply when a different engine reads the same files. Identity has no row of its own; it is recorded here.
Key custody	Who holds the encryption keys: the provider, the customer through the provider's key service, or the customer outside the provider.
Compute and SQL dialect	The engines offered and the SQL they speak, including procedural code and functions that exist on this platform only.
Billing meters	The units the invoice counts (data scanned, credits, capacity units, storage, network), named without prices.
What travels	Three lists: what moves with a file copy, what moves with work, and what must be rebuilt on the next platform.

The last row carries the book's thesis. On every card it has the same three states. **Travels**: the data files and, for open table formats, the table metadata. **Travels with work**: SQL in a different dialect, pipelines, the semantic model if an export or translation exists. **Must be**

rebuilt: whatever exists only inside one vendor's control plane, which in the insurer's case meant its row policies, masking rules and grants. Chapter Fifteen tests that row on the insurer's tables.

The card has limits. It records structure, not quality: it will not tell you which engine is faster on your queries or which support desk answers sooner. The full card, with a source link for every fact and fields that do not fit on a printed page, belongs in the dated web companion, where it can be corrected when a vendor ships.

Why this book prints no rankings

This book prints no rankings because products change faster than print, some license terms restrict publishing benchmark results without the vendor's consent (Oracle's Technology Network License Agreement, for one, forbids disclosing "results of any Program benchmark tests without Oracle's prior consent"), and a ranking answers someone else's workload. Chapter Sixteen shows how to run your own test on your own queries. Prices are left out because they change faster still and differ by contract; they live in the dated web companion.

What the book prints instead is structure that changes slowly: the layers, who holds them, and what a move costs in rebuilding. A platform can double its speed in a year. It rarely changes where it keeps your policies.

WHAT HASN'T CHANGED

The 2021 Lakehouse paper by Armbrust, Ghodsi, Xin and Zaharia described first generation warehouses as systems that "typically coupled compute and storage into an on-premises appliance". Since the appliance, the layers have been pulled apart. The appliance answered the question "who decides here" with one name for every layer; today a buyer has to ask it eight times.

ERA CARD

Reader. Every reader since the warehouse reaches data through all eight layers, whether they see them or not; an assistant reads only what the semantic layer defines and the policy rung allows.

Security. Control sits in identity, policy and keys, and each can be held by a different party; the map shows which party you must trust at each rung.

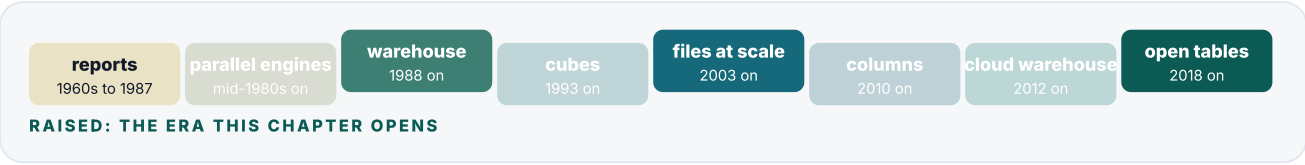
Cost. The billing rung meters compute and storage; the exit cost sits in the middle rungs, in what must be rebuilt.

AI. An agent asking for "the loss ratio" gets whatever the semantic layer defines and the policies allow; neither travels with the files.

Monday rule: for your most important board number, write one line per layer naming the person or vendor who holds authority there, and mark every layer where the answer is "nobody".

CHAPTER TEN

Catalog, Semantics and Policy



In November 1992 ISO published the third edition of its SQL standard, ISO/IEC 9075:1992, known as SQL-92. Among its 587 pages was a clause that gave every catalog a schema named INFORMATION_SCHEMA: a set of views describing every table, column and privilege in that catalog. The freely readable draft of July 1992 contains a sentence that reads today like a design brief. The query behind each view "ensures that a given user can access only those rows of the view that represent descriptors on which he has privileges." The description of the data and the rules about who may see it were specified together, in one place, inside one database. More than thirty years later they sit in different products, often from different vendors, and a board number can depend on all of them. The decision this chapter changes: **name one system of record for each of definitions, grants and policies, and know which engine enforces each one when it reads**.

From dictionary to information schema to metastore

A **data dictionary** is a list of the data a company keeps: each field, its name, its type, what it means, who owns it. It could be a binder or a separate product kept beside the database, and nothing forced a program to update it when a field changed. The insurer's STATUS_CODES table, whose loss in 2021 dropped status 7 policies from the board view, held dictionary knowledge, what each status code means, as rows of data, and so could be left behind by a migration.

The relational database pulled part of the dictionary inside. SQL-92 specified that the data in the INFORMATION_SCHEMA views is "a representation of all of the descriptors in all of the schemas in that catalog", and that a database "need do no more than simulate" the base tables behind them. A program could now ask the database what tables existed. What the database knew was structure: names, types, keys, privileges. It did not know that status 7 meant perpetual and in force.

Hadoop needed the same register for files. Hive's design document calls its **metastore** "the component that stores all the structure information of the various tables and partitions in the warehouse", and says it provides "two important but often overlooked features of a data warehouse: data abstraction and data discovery." The Hive metastore became the shared address book of the data lake: many engines learned to read it, so many engines could find the same tables.

The cloud copied that address book as a service. AWS announced the general availability of AWS Glue on 14 August 2017; the same day it told Amazon EMR users they could store "external table metadata for Hive and Spark" in the Glue Data Catalog "instead of utilizing an on-cluster or self-managed Hive Metastore." The catalog had become something you rent. AWS's current documentation describes the Data Catalog as "a centralized repository that stores metadata about your organization's data sets", with fine-grained access through AWS Lake Formation.

The open catalogs of 2024, and where lock-in now hides

Open table formats (Chapter Eight) moved the record of which files make up a table into open metadata files. They did not settle who points to the current metadata. The Apache Iceberg project answered with an interface, the **REST catalog specification**, an OpenAPI document that "defines the specification for the first version of the REST Catalog API": create and list namespaces, create, load, commit and rename tables and views, and load storage credentials for a table. Any catalog that implements it can be used by any engine that speaks it.

In 2024 the large vendors implemented it. Snowflake announced Polaris Catalog on 3 June 2024, saying it "relies on Iceberg's open source REST protocol". It entered the Apache Incubator on 9 August 2024 and graduated as Apache Polaris, a top-level project, on 15 February 2026, per the Incubator's project page checked on 20 November 2026. Databricks open sourced Unity Catalog on 13 June 2024 under the Apache 2.0 license, hosted at LF AI and Data, and wrote that it "implements the Iceberg REST Catalog API for access from the Iceberg engine ecosystem". AWS now documents an Iceberg REST endpoint for the Glue Data Catalog, where the same IAM actions and Lake Formation permissions are checked on every request.

So the address book is now reachable through an open interface at Snowflake, Databricks and AWS. That is real progress, and it moves the question up one layer. A REST catalog tells an engine where a table's metadata is and can hand it a short-lived credential to read the files. It has said far less about what the engine must do with the rows once it has them. Grants, row filters, masking rules, tags and the owners of each are stored in each vendor's

catalog in that vendor's model. When you leave, the files and the Iceberg metadata travel; those objects do not. **Lock-in did not disappear when the files opened. It moved into the catalog.**

On each platform. The catalogs differ in what they hold beyond the address book. Unity Catalog and Snowflake's catalog hold grants and policies for their own engines; Glue holds metadata while Lake Formation holds fine-grained permissions; an Apache Polaris deployment holds the tables and its own role-based privileges, which its documentation says an administrator assigns to catalog roles. Chapter Twelve records each on a card, dated November 2026.

Meaning above the tables: the semantic and metrics layer

A catalog knows that a table called claims has a column called incurred. It does not know how the board's loss ratio is built from it. That knowledge lives in the **semantic layer**: the models a BI tool reads, the named measures, the joins and filters, and increasingly the definitions an assistant uses to answer a question in plain language. Vendors describe it in their own words. Google's documentation says "LookML, short for Looker Modeling Language, is the language that is used in Looker to create semantic data models." Microsoft says Power BI semantic models "represent a source of data that's ready for reporting and visualization", and that a model built in Power BI Desktop "is effectively an Analysis Services tabular model." dbt Labs says its Semantic Layer, "powered by MetricFlow, simplifies the process of defining and using critical business metrics."

Each is a language. A measure written in one does not run in another without translation, and the translation is work: the "travels with work" state of the platform card. Vendors have noticed. On 23 September 2025 Snowflake, with Salesforce, dbt Labs, RelationalAI and others, announced the Open Semantic Interchange, a "common, vendor-neutral semantic model specification that standardizes how semantic metadata is defined and shared." It is an initiative, not yet a guarantee: treat semantic portability as something to test, not to assume.

The insurer's board metric shows why the semantic layer must hold more than a formula. Its contract, approved by the audit committee in 2019, fixes the population (status 7 policies included, the affiliate's pooled business reported separately), the basis, the netting of anticipated salvage and subrogation, and the rounding rule (half to even). A semantic model that holds only "losses incurred over net premiums earned" holds the part that was never in dispute. The parts that moved the number (the population filter, the netting, the rounding, and the snapshot it was read from) are exactly the parts that were written somewhere else: in a 2021 view, in a vendor's row policy, in an engine's default. A metric definition worth the name carries the rows of the metric passport from Chapter One: id and version, numerator,

denominator, population and exclusions, basis (netting and rounding included), the effective date and knowledge cutoff, the snapshot id or tag, the transformation version, the owner and approver, and the restatement status.

> *A semantic model that holds only the formula holds the part that was never in dispute.*

Two more tools are often sold as catalogs and are not. **OpenLineage** describes itself as "an open framework for data lineage collection and analysis": a specification for events that jobs emit when they run, built from three entities (dataset, job, run) and extensible "facets". It is hosted at LF AI and Data. An OpenLineage event says that a job read these datasets and wrote that one at this time; it is evidence for the Ledger's "how it moved" column. It does not name tables for engines, and it does not decide access. A lineage store tells you what happened; a catalog decides what can happen.

A **data contract** is a written agreement between the producer of a dataset and its consumers: schema, meaning, quality checks, service levels, owner. The Open Data Contract Standard, maintained by the Bitol project ("a Graduated project of the Linux Foundation AI & Data"), defines the keys and values of such a contract in YAML. A contract is where the producer promises what status 7 means. It is enforced only where a pipeline checks it: a contract nobody tests is a dictionary again.

Decided in one place, enforced in another

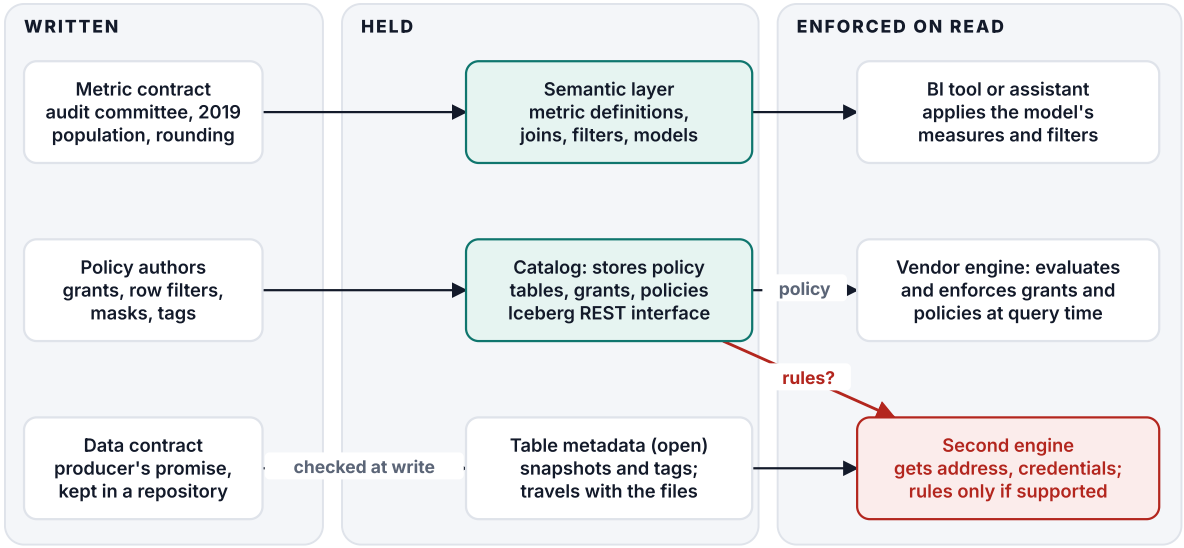
Access control has a vocabulary for the split this chapter keeps finding. NIST's guide to attribute-based access control (SP 800-162, first published January 2014) defines a **policy decision point**, which "computes access decisions by evaluating the applicable" policies, and a **policy enforcement point**, which "enforces policy decisions in response to a request from a subject requesting access to a protected object". It adds that the two "may be physically and logically separated from each other" (section 2.4.3), and it names a third function, the policy administration point, where policies are created, managed and stored.

On a data platform the catalog usually plays the administration point: it stores the policy. The decision is computed at query time, most often inside the engine that reads, which also enforces it. Snowflake documents a row access policy as "a schema-level object that determines whether a given row in a table or view can be viewed", evaluated at query time with the role of the policy owner. Google documents that, for column-level control, "BigQuery checks the column policy tag and its policy to see whether the user is authorized to access the data." Databricks defines a row filter as a SQL function that "evaluates each row at query time" and a column mask as a function that "returns the original value or a

masked version", and it states the boundary plainly: "You cannot use Iceberg REST catalog or Unity REST APIs to access tables with row filters or column masks, except for reads from external engines through cross-engine ABAC." The policy stays enforceable because the catalog refuses to hand the table to an engine it cannot trust to apply it; there the catalog does act as a decision point, for the table as a whole.

FIGURE 10.1

Where meaning, grants and policy live: written in one place, held in another, enforced where the data is read



The catalog stores and administers the policy; the engine that reads usually computes the decision and enforces it; a second engine reading the same files gets the address and the credentials, and the rules only where it supports them.

Author's diagram based on NIST SP 800-162 (policy administration, decision and enforcement points, section 2.4.3), the Apache Iceberg REST catalog specification and vendor documentation cited in this chapter.

Figure 10.1 lays out the three places. Meaning is written in a contract and held in a semantic model. Grants and policies are written by their authors and held in the catalog. Both are enforced only when an engine or a BI tool reads. When a second engine reads the same files through an open interface, it receives the address and the credentials; whether it also receives the rules is the question every exit test asks.

The open standard has started to answer it. The REST catalog specification released with Apache Iceberg 1.12.0 (tagged on 30 September 2026) added an optional "read-restrictions" field to the load-table response: a required row filter and required column actions, such as masking or replacing a value with null. A reader that supports it "must fail any read against

the loaded table that cannot apply a returned restriction in full." That puts the rule in the catalog's answer and the duty to enforce it on any engine that implements the field. As of November 2026 it is new; check which catalogs return it and which engines honour it before you rely on it.

The practical step is to give each object one system of record and to know, before any move, which state of the platform card it is in. For the insurer the list looked like this in November 2026:

Object	System of record	Used or enforced by	On a file copy
Metric definition (population, netting, rounding)	Semantic layer, one version	BI tool, assistant, engines	Travels with work: translated
Quarter-close snapshot tag	Iceberg table metadata	Any Iceberg reader	Travels, if retained
Grants	Catalog	Engine at query time	Must be rebuilt
Row filters and masks	Catalog	Engine at query time	Must be rebuilt, unless both sides support read restrictions
Classification tags	Catalog	Policies attached to tags	Must be rebuilt
Lineage events	Lineage store	Audit, the Ledger	Travels as an export
Data contract	Code repository	Pipeline checks	Travels; checks rerun

Two rows deserve a second look. A row filter that protects personal data belongs in the catalog. A row filter that decides which business is in a board number does not: it is part of the metric, and it should sit in the metric's definition where an auditor, a second engine and an assistant can all read it.

Hop 6, seen from the catalog

Chapter Eight told hop 6 from the table's side: in 2024 the insurer rewrote its claims and premium tables as Iceberg on object storage, under a vendor-managed catalog with an Iceberg REST interface, read by two engines. From the catalog's side the same move looks like this. The catalog now held, apart from any one engine, the authority the warehouse's services layer had held inside it. Catalog administrators now set the row and column policies

and the grants; the platform team and the write jobs committed snapshots. The close job on the SQL warehouse engine and the analysts' notebooks both read the tables through the catalog, so its row policies applied to both.

What the catalog did not hold is the "as of" of the board figure, 61.8 percent (approved 5 February, data to 23 January). No snapshot id was bound to it and the table metadata held no quarter-close tag, so when the analyst's notebook read the table on 17 February it read the latest snapshot of the quarterly loss table, committed on 13 February. Nothing but the table's 45-day maximum snapshot age kept the quarter-close snapshot of 23 January, and the Sunday expiry run would have removed it on 15 March; the tag came only on 24 February. One more thing sat in the catalog without anyone recording why: a row access policy on the claims table. Both readings passed through it, so it is not one of the effects in the Chapter Eight bridge; Chapter Fifteen shows what it had been doing. The loss code for hop 6 is TIME; seen from the catalog, it is also a gap in authority. The catalog resolved every reader to the table's current metadata, and so to the latest snapshot, unless the reader asked for another.

WHAT HASN'T CHANGED

SQL-92 required that the Information Schema show each user only the descriptors "on which he has privileges". In 1992 the catalog that described the data and the catalog that guarded it were one object. Every platform since has had to decide whether to keep them together, and every exit has had to find out.

ERA CARD

Reader. Engines find tables through the catalog; people and assistants find meaning through the semantic layer; neither sees the other's rules unless someone joins them.

Security. Policies are stored in the catalog and enforced by the engine that reads; a second engine that reads the files without the policies is a second, unguarded door.

Cost. Files copy at the price of egress; grants, policies and semantic models are rebuilt at the price of people.

AI. An assistant answers from the semantic layer it is given; if the metric's population, rounding and snapshot are not in it, the assistant will supply its own.

Monday rule: list the row policies and masking rules on the tables behind your board numbers, and for each one write down whether it protects data or computes the number; move every rule of the second kind into the metric's definition.

CHAPTER ELEVEN

Security and Resilience on a Shared Platform



Snowflake's documentation lays out a three-phase plan to deprecate single-factor password sign-ins. In the third phase, enforced account by account from August 2026, human users can no longer sign in with a password alone, and non-human users are blocked from password authentication altogether. When this book was checked, on 20 November 2026, the page still gave August to October 2026 as the estimated window, said enforcement may be extended for complex workloads, and recorded no completion. A platform changing a default is not news on its own. What makes this one worth a chapter opening is what it shows about the line between provider and customer. Until the change, whether a second factor protected a password sign-in was a setting the customer chose. Two years before the third phase, Mandiant had reported a criminal campaign that used stolen passwords against Snowflake customer accounts without a second factor; the box later in this chapter tells it from the report. The decision this chapter changes: **for every control that protects your board data, write down which side of the shared line it sits on, and prove you can restore the catalog as well as the files**.

The line in shared responsibility

Every cloud provider publishes a version of the same diagram. AWS calls its two halves "security of the cloud", the hardware, software, networking and facilities that AWS runs, and "security in the cloud", which is the customer's and depends on the services the customer chooses. Microsoft's page is more direct about what never moves: "Regardless of the type of deployment, you always retain" responsibility for data, endpoints, accounts and access management, "including role-based access control (RBAC), multifactor authentication, and conditional access policies."

Map that onto the eight layers of Chapter Nine. On a managed data platform the provider runs storage durability, the engines and the control plane. The customer holds the accounts, the grants and policies, the decision about keys and the decision about how long history is

kept. Those are the rungs where meaning and control live, and they are the rungs where the incident in this chapter began and where exits stall. A managed platform reduces what you operate. It does not reduce what you decide.

Control on board data	Provider runs	Customer decides	Evidence to keep
Sign-in and second factor	The sign-in service	Federation, MFA, service accounts	List of accounts without MFA, reviewed monthly
Grants and policies	The policy engine	Every grant, row policy, mask and tag	Policies as code, with an owner each
Keys	The key service	Provider-managed, customer-managed or external	Key owner, revocation runbook, outage test
History and backups	Durable storage	Retention, tags, backup copies	Restore test results, with times
Shares and copies	The sharing service	Who receives what, and what they may keep	A Ledger row for each share and copy

Identity first, then policy

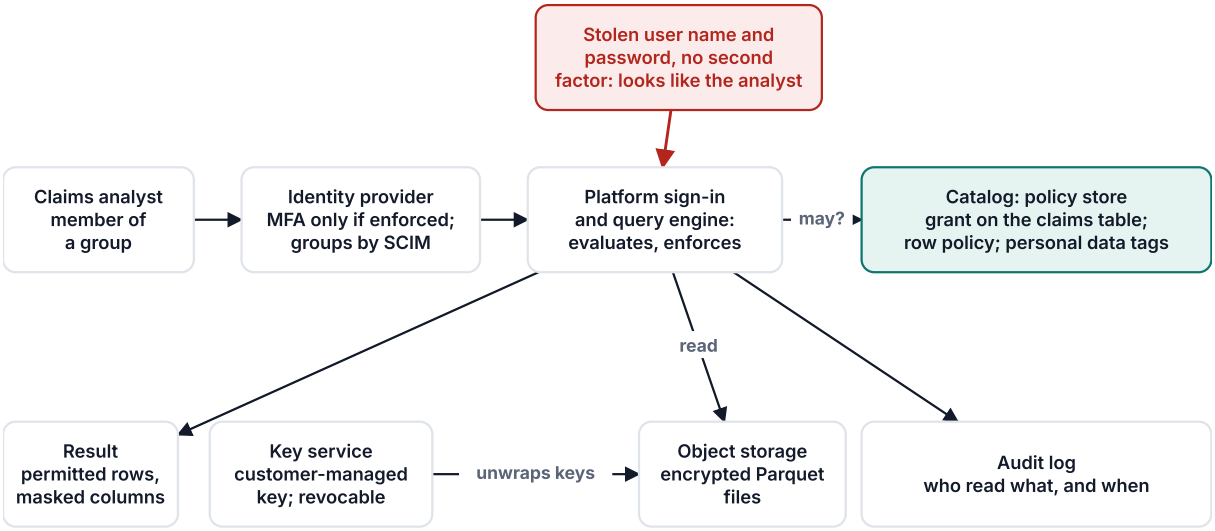
Access to a table starts with an identity. Most enterprises no longer keep passwords inside the data platform. They **federate**: the platform trusts a corporate identity provider, which signs the user in and asserts who they are. The standards are old and stable. SAML 2.0 was approved as an OASIS Standard on 1 March 2005 (its specification documents carry the date 15 March 2005). OpenID Connect Core 1.0 is, in its own words, "a simple identity layer on top of the OAuth 2.0 protocol." SCIM, published by the IETF as RFC 7644 in September 2015, lets the identity provider create, change and remove users and groups in the platform, so that a leaver loses access when HR closes the record rather than when someone remembers.

Federation moves the strongest control (sign-in, the second factor, the leaver process) out of the platform and into the identity provider. It leaves two gaps. The first is the account that was created locally before federation and never removed. The second is the **service account**, the identity a pipeline or a BI server uses, which often has more rights than any person and no second factor at all. Snowflake's phase three draws the same distinction: people get a second factor, machines lose passwords.

Once the identity is known, policy decides what it may see. A **grant** gives a role access to a table. A **row policy** filters rows: Snowflake documents a row access policy as "a schema-level object that determines whether a given row in a table or view can be viewed." A **masking policy** returns a masked value in place of the real one for roles that may not see it. A **tag** classifies a column, for example as personal data, and a policy can be attached to the tag rather than to each column: Snowflake's tag-based masking allows "a masking policy to be set on a tag using an ALTER TAG command", and every column that carries the tag is protected. Google's BigQuery uses policy tags the same way for column-level control. Tagging once and attaching policy to the tag is how a large estate stays consistent when new tables appear every week.

FIGURE 11.1

Who can read the claims table: identity, a policy decision, enforcement at read time, and a key that can say no



Every check after sign-in trusts the identity it is given: a stolen user name and password, presented at the platform's own sign-in without a second factor, passes the catalog, the engine and the key service as if it were the analyst.

Author's diagram based on NIST SP 800-162 (administration, decision and enforcement points), vendor documentation on row access, masking and key custody cited in this chapter, and Mandiant's UNC5537 report (10 June 2024).

Figure 11.1 follows one read of the insurer's claims table, which holds claimant names and notes covered by federal and state privacy rules. The analyst signs in through the identity provider with a second factor. The analyst's query reaches the platform's engine, which checks against the catalog whether the analyst's role may read the table, which rows the row policy allows and which columns the tags mask. The engine reads the encrypted files, the key

service releases the keys that decrypt them, the engine returns what the decision allows and writes an audit record. Every box after sign-in trusts the identity it was handed. That is the weakness the box below describes.

SOURCED FAILURE: UNC5537 AND SNOWFLAKE CUSTOMER ACCOUNTS, 2024

On 10 June 2024 Mandiant, part of Google Cloud, published "UNC5537 Targets Snowflake Customer Instances for Data Theft and Extortion" (updated 17 June). Its introduction states that the investigation "has not found any evidence to suggest that unauthorized access to Snowflake customer accounts stemmed from a breach of Snowflake's enterprise environment", and that every incident it responded to "was traced back to compromised customer credentials." In the same section it reports that Mandiant and Snowflake had notified "approximately 165 potentially exposed organizations." The report names none of them, and neither does this book.

The campaign overview explains how. The credentials "were primarily obtained from multiple infostealer malware campaigns that infected non-Snowflake owned systems." The report lists three conditions behind the successful logins: the impacted accounts "were not configured with multi-factor authentication enabled"; stolen credentials "were still valid, in some cases years after they were stolen, and had not been rotated or updated"; and the instances "did not have network allow lists in place to only allow access from trusted locations." Its outlook calls for "the universal enforcement of MFA and secure authentication" and for limiting traffic to trusted locations.

Read against Figure 11.1, the controls after sign-in had nothing to refuse: the identity presented to them was valid. The conditions the report lists sat on the customer's side of the line, in identity and network settings.

Keys: who can say no

Assume the data is encrypted at rest. The question for a CTO is who holds the key, because whoever can revoke it can make the data unreadable. There are three common arrangements:

- **Provider-managed keys.** The platform creates, rotates and holds the keys. Nothing to run, and no key of your own to revoke; access is withdrawn through accounts and grants instead.
- **Customer-managed keys.** The customer controls a key in the provider's key service, and the platform's keys are wrapped by it. Snowflake's Tri-Secret Secure "creates a composite master key that protects your Snowflake data" from a Snowflake key and a customer key; if the customer revokes its key, "your data can no longer be decrypted by Snowflake."

- **External keys.** The key never enters the provider's infrastructure. AWS launched an External Key Store for its key service on 29 November 2022, for "encryption keys stored in an external key management system under their control." Google Cloud's External Key Manager works on the same idea, and its documentation is blunt about the price: "Inability to access the external key can also cause permanent data loss."

Each step down this list gives the customer more control and adds one more thing that can fail. An external key is a strong answer to a regulator or a sovereignty requirement, and a new single point of failure in the recovery plan. Choose it for the datasets that need it, and test what happens when the key service is down before an auditor or an outage does it for you.

The health system, the book's second running company, shows where key custody meets law. Under HIPAA, a cloud provider that stores electronic protected health information for a hospital is a business associate. HHS's guidance on cloud computing says so even when the provider holds only encrypted data: a provider "that maintains encrypted ePHI on behalf [of] a covered entity (or another business associate) is a business associate, even if it does not hold a decryption key." So holding the key yourself does not remove the need for a business associate agreement. The guidance adds that encryption "does not maintain the integrity and availability of the ePHI", which is the resilience question of the last section. Whether an agreement is needed follows from what a provider does with the data for the hospital (creating, receiving, maintaining or transmitting it), not from the vendor's name.

The health system also tags its data before any policy can work. Columns in FHIR resources and imaging metadata that hold protected health information carry a PHI tag, and masking follows the tag. Images are harder. The DICOM standard defines an attribute, Burned In Annotation, that "indicates whether or not image contains sufficient burned in annotation to identify the patient and date the image was acquired", and adds that if it is absent "the image may or may not contain burned in annotation." A tag on the metadata does not reach text written into the pixels. DICOM's de-identification profile has a Clean Pixel Data option for that reason: it requires removing "any information burned in to the Pixel Data", "regardless of how that is achieved." The research copy is only de-identified if the pixels were cleaned too.

Sharing without copies, and clean rooms

The fastest way to lose track of data is to send someone a copy. The platforms of the 2020s offer a different route: let the recipient read the data in place. Databricks introduced Delta Sharing on 26 May 2021 as "an open protocol for secure real-time exchange of large datasets": the recipient's client asks a sharing server for a table, the server checks access and returns "short-lived pre-signed URLs" so the client reads the Parquet files directly from

cloud storage. Snowflake's Secure Data Sharing works inside its own platform: "no actual data is copied or transferred between accounts", and the consumer pays only for the compute used to query it. A **clean room** goes one step further and lets two parties run agreed analyses on combined data, under analysis rules that limit what either can see of the other's rows; AWS Clean Rooms, for one, became generally available on 21 March 2023.

A share is a grant across a company boundary, and a grant can be revoked. That is its strength over a copy. Its limit is the recipient's query result: once a partner exports what they read, that export is a copy, with its own location and its own deletion reach. Record each share in the Ledger as a hop with no file moved, and record what the recipient may keep.

Resilience: snapshots are not backups

Table formats keep history, and it is tempting to treat that history as a backup. It is not one. Iceberg's default maximum snapshot age for expiry is five days, with at least one snapshot kept; tags and branches are kept indefinitely by default unless given their own maximum age. Snowflake's Time Travel retention is one day by default and up to 90 days on its Enterprise edition, followed by a seven-day Fail-safe that, in Snowflake's words, "is not provided as a means for accessing historical data after the Time Travel retention period has ended. It is for use only by Snowflake to recover data that may have been lost or damaged due to extreme operational failures." Snowflake's own documentation advises protecting critical data with its Backups feature "instead of relying on Fail-safe."

A snapshot lives in the same table, under the same catalog, in the same account, usually within reach of the same administrators. A table dropped with its files, a compromised administrator or a wrong retention setting can take the history with it. A backup is a copy somewhere a single mistake or a single stolen password cannot reach, with its own retention and its own owner, and with the deletion reach this series asks you to record for every copy. A backup you have never restored is a hope. Schedule a restore test, time it, and record the time in the runbook.

Region failure needs a written plan, because the features are not uniform. Snowflake documents that database replication is available to all accounts while replication of other account objects and failover "require Business Critical Edition (or higher)". Open tables bring their own catch: Iceberg metadata commonly records fully qualified file paths, so a copy at a new address needs its metadata rewritten; Iceberg's rewrite_table_path procedure "stages a copy of the Iceberg table's metadata files where every absolute path source prefix is replaced by the specified target prefix."

Last, recover the catalog, not only the files. If the catalog is lost, Iceberg's register_table procedure "creates a catalog entry for a metadata.json file which already exists", which is how files become tables again. Its warning is the reason to rehearse it: "Having the same metadata.json registered in more than one catalog can lead to missing updates, loss of data, and table corruption." And registering a table restores its address, not its grants, policies or tags. Those come back only from wherever you exported them, which is a strong argument for keeping policies as code in a repository. For the insurer, that means the quarter-close snapshot of every board number carries a tag with retention set to the statutory records period, and the restore test includes reading the board figure from that tag in the recovered catalog.

WHAT HASN'T CHANGED

SAML 2.0 was approved as an OASIS Standard on 1 March 2005, seven years before Google made BigQuery publicly available in May 2012. The weak point federation removes, a password kept in each system, is the one the 2024 campaign used.

ERA CARD

Reader. Every reader arrives through an identity; people federated from a corporate directory, machines through service accounts that need the same scrutiny.

Security. Grants, policies and keys answer whatever identity they are given; the 2024 campaign entered through credentials without a second factor, on the customer's side of the line.

Cost. External keys, cross-region replication and restore tests all cost money; a recovery that has never been rehearsed costs more.

AI. An agent's service account is an identity like any other: give it its own grants and its own audit trail, never a person's credentials.

Monday rule: list every account that can read your board data without a second factor, including service accounts, and close or convert each one; then restore one board table and its policies into an empty catalog and time it.

CHAPTER TWELVE

Seven Cards: Six Platforms and an Open Stack



On 15 November 2023 Microsoft announced that Microsoft Fabric was "now generally available for purchase". From that day a CTO who ran analytics on Azure could buy three Microsoft analytics offerings with overlapping descriptions. After the February board meeting at which 61.8 percent (approved 5 February, data to 23 January) sat beside 64.1 percent (the lakehouse notebook, snapshot of 13 February), the insurer's CFO asked the chief architect for one page comparing "the six platforms": AWS, Microsoft Azure, Google Cloud, Databricks, Snowflake and Fabric. The architect sent back seven cards and one sentence: the six are not six of the same thing. The decision this chapter changes is the shape of the comparison itself: **compare platforms layer by layer, in two tiers, against an open stack you could assemble yourself, and read the last row of every card before the first**.

Why the six are not peers

Three of the six names are clouds. AWS, Microsoft Azure and Google Cloud sell the **substrate**: the object storage the files sit in, the identity system that decides who a caller is, the key service that encrypts the bytes, the network the data crosses and the meter that charges when it leaves. Three are platforms that run on top of a cloud: Databricks and Snowflake each run on all three clouds, and Microsoft Fabric runs as a software service on Microsoft's own foundation. A platform buys its storage, identity and keys from a tier 1 cloud, or hides that purchase inside its own bill. Comparing Snowflake with AWS compares a tenant with a landlord that also rents out analytics services of its own.

Each cloud also sells its own warehouse and lakehouse services: Amazon Redshift and Amazon Athena on AWS, Azure Synapse Analytics on Azure, Google BigQuery and its Lakehouse runtime catalog on Google Cloud. They are written into the tier 1 cards below, which is why BigQuery and Redshift have no card of their own.

The seventh card is the control: an **open stack** a team assembles from parts with public specifications. Object storage, the Apache Iceberg table format, an open catalog that implements the Iceberg REST interface, and an open engine. Nobody sells it as one product. It is on the page because every other card is easier to read beside it: whatever a platform does that the open stack cannot is either the value you pay for or the part you will rebuild when you leave.

Every card uses the ten rows defined in Chapter Nine, in the same order, so that a reader can run a finger along one row across seven cards. Every line comes from the vendor's own documentation, read on 20 November 2026, with the addresses in the sources for this chapter (Appendix E). Product names, defaults and preview labels move every quarter; the live cards, their links and the prices this book does not print are kept in the dated web companion. The **billing meter** row names the unit a vendor counts, never what a unit costs, because the unit decides how a workload's shape becomes a bill (Chapter Thirteen).

FIGURE 12.1

Same layers, two tiers, seven cards: the platforms run on the clouds, and the open stack is the control



Each column summarizes selected rows of one card of this chapter, as of November 2026; where a cell and a card differ, the card wins, and vertical position across the tiers implies no pairing. The clouds' own warehouses and lakehouse services (Amazon Redshift, Amazon Athena, Azure Synapse Analytics, Google BigQuery) belong to their tier 1 cards. Tier 1 rows are substrate rungs, not the tier 2 rows beside them: identity and policy is split across both tiers, and network and egress is a substrate meter rather than one of Chapter Nine's eight layers.

Author's diagram based on vendor and Apache project documentation read 20 November 2026 (sources for Chapter Twelve, Appendix E).

The last row, "What travels", always has three parts. **Travels**: what another engine can read in place or after a plain copy. **Travels with work**: what can move but must be converted, exported or rewritten. **Must be rebuilt**: what has no export and must be built again by people on the other side. The insurer's exit test in Chapter Fifteen lives in the third part.

Tier 1: the clouds as substrate

The three clouds differ less in their analytics features than in how storage, identity and keys are wired together. On AWS the Glue Data Catalog and Lake Formation sit between the engines and the files and decide which rows and columns a reader sees. On Azure the storage account itself carries directory and file permissions, and the Microsoft governance catalog is in transition. On Google Cloud the row and column policies are defined on BigQuery tables.

Platform card: Amazon Web Services (tier 1) · as of November 2026

Scope and date	Substrate services for analytics: Amazon S3 (general purpose buckets and S3 table buckets), AWS IAM, AWS KMS, AWS Glue Data Catalog, AWS Lake Formation; engines in the same tier: Amazon Athena, Amazon Redshift, Amazon EMR. Documentation read 20 November 2026.
Storage of record	Amazon S3. S3 Tables, announced 3 December 2024, store tables "as subresources" of a table bucket and run compaction, snapshot management and unreferenced file removal automatically. Redshift tables sit in Redshift Managed Storage.
Table formats (native, read, write)	Native: Apache Iceberg in table buckets, including Iceberg V3 features. Athena reads and writes Iceberg (it creates and operates on V2 tables) and reads Delta Lake read only, without Delta time travel.
Catalog and its interface	AWS Glue Data Catalog, with an Iceberg REST endpoint that "supports API operations specified in the Apache Iceberg REST specification" (single-level namespaces). Clients sign with SigV4 or present an OAuth 2.0 bearer token; Lake Formation permissions are enforced on every request.
Semantic layer	Not a substrate service: definitions live in the platform or BI tool chosen above this tier.
Where policies are enforced	IAM and service control policies on buckets, namespaces and tables. Lake Formation data filters give column-, row- and cell-level security on SELECT in integrated engines (Athena, EMR, Redshift Spectrum); filters apply to reads only. With row or cell filters present, Athena refuses the Iceberg metadata tables ($snapshots, $files) to the filtered reader.
Key custody	SSE-S3 is the default for every bucket; new uploads are encrypted automatically since 5 January 2023. Options: SSE-KMS and DSSE-KMS with customer managed keys, SSE-C, client-side encryption. An external key store keeps key material in a key manager you run outside AWS; you then answer for its availability.
Compute and SQL dialect	Athena (standard SQL, engine version 3; Spark on Athena). Redshift: "based on PostgreSQL", with important differences in storage, functions and types. EMR runs Spark and Hadoop.
Billing meters	S3: storage, requests, Data Transfer OUT; S3 Tables add object monitoring and compaction charges. Athena: data scanned, or DPU capacity. Redshift Serverless: RPU use metered per second (60-second minimum), storage per GB-month.
What travels	**Travels:** Parquet, ORC or Avro data files and Iceberg metadata in S3. **Travels with work:** Glue table registrations (re-register in another Iceberg REST catalog); Redshift SQL out of a PostgreSQL-derived dialect. **Must be rebuilt:** IAM and Lake Formation grants and data filters; KMS key policies (data is decrypted and re-encrypted under new keys).

Platform card: Microsoft Azure (tier 1) · as of November 2026

Scope and date	Substrate services for analytics: Azure Storage with Data Lake Storage capabilities, Microsoft Entra ID, Azure Key Vault and Key Vault Managed HSM, bandwidth; engine in the same tier: Azure Synapse Analytics. Azure Databricks and Fabric have their own cards. Documentation read 20 November 2026.
Storage of record	Azure Data Lake Storage, which "isn't a dedicated service or account type": Blob Storage with the hierarchical namespace switched on. Files persist as blobs; Hadoop-compatible access through the ABFS driver.
Table formats (native, read, write)	No table format at the storage layer: files. Synapse serverless SQL pool reads Parquet, Delta Lake and delimited text in place; it writes files with CREATE EXTERNAL TABLE AS SELECT and supports no DML.
Catalog and its interface	Microsoft Purview Unified Catalog (governance domains, data products, access policies). Microsoft states it is "being retired": no new accounts, maintenance mode "through November 12, 2027", governance consolidating in Fabric (retirement notice of 12 November 2026). Synapse serverless databases hold metadata objects only.
Semantic layer	Not a substrate service: definitions live in the platform or BI tool chosen above this tier.
Where policies are enforced	Azure RBAC and POSIX-like access control lists at directory and file level on the storage account. Synapse serverless SQL: logins and users, GRANT, DENY and REVOKE per object, Entra integration; storage reached by SAS, user identity pass-through or workspace identity.
Key custody	All data encrypted at rest with Microsoft-managed or customer-managed keys. A customer-managed key in Key Vault or Managed HSM wraps the account encryption key, and may sit in another tenant or region. Disabling the key makes reads and writes fail with 403.
Compute and SQL dialect	Synapse serverless SQL pool, T-SQL ("slightly enhanced/extended" for files); Apache Spark for Azure Synapse.
Billing meters	Storage at Blob Storage levels (capacity and transactions). Bandwidth: data in free; outbound between regions and Internet Egress charged by volume tier. Serverless SQL: data processed by queries, no charge for reserved resources.
What travels	**Travels:** Parquet and Delta files in Data Lake Storage, over HDFS-compatible and REST interfaces. **Travels with work:** Synapse T-SQL views and CETAS scripts; Purview glossary terms and data products, before the retirement date. **Must be rebuilt:** RBAC assignments, ACLs, Entra and managed identities, Key Vault keys and their permissions.

Platform card: Google Cloud (tier 1) · as of November 2026

Scope and date	Substrate services for analytics: Cloud Storage, IAM, Cloud KMS and Cloud External Key Manager, network; engines and catalog in the same tier: BigQuery and the Lakehouse runtime catalog (its API and roles are still named BigLake). Documentation read 20 November 2026.
Storage of record	BigQuery storage, "a completely managed service", columnar and separate from compute. Iceberg managed tables keep Parquet data and Iceberg metadata in Cloud Storage buckets you control.
Table formats (native, read, write)	Native: BigQuery tables. Iceberg managed tables: BigQuery writes (DML, loads, Storage Write API); other engines read exported Iceberg V2 metadata, and Google warns that changing files outside BigQuery can cause query failures or data loss. Through the REST catalog: Iceberg V2 GA, V3 in Preview.
Catalog and its interface	Lakehouse runtime catalog with an Apache Iceberg REST catalog endpoint "built on the Apache Iceberg REST Catalog API", with credential vending; Spark and Trino named as clients.
Semantic layer	Not a substrate service: Google's Looker, with its LookML models (Chapter Ten), is a separate product above this tier.
Where policies are enforced	IAM roles and permissions on BigQuery resources. Row-level access policies "act as filters to hide or display certain rows"; column-level security through policy tags in a taxonomy; data masking applied on top of row-level security. Checked at query time in BigQuery.
Key custody	Encrypted at rest by default with Google-owned keys. CMEK in Cloud KMS for BigQuery; Cloud EKM uses keys held by a supported external key manager, which Google states it does not cache or store.
Compute and SQL dialect	BigQuery with GoogleSQL, described as "ANSI-compliant"; a legacy SQL dialect remains with restrictions. Open engines through the REST catalog.
Billing meters	BigQuery on-demand: bytes processed (TiB). Capacity: slot-hours, three editions. Storage: logical or physical bytes, chosen per dataset. Iceberg managed tables: loads and exports use Enterprise pay-as-you-go slots. Cloud Storage: data storage, data processing (operations, retrieval, inter-region replication), network usage.
What travels	**Travels:** Cloud Storage objects; the Parquet files and exported metadata of Iceberg managed tables. **Travels with work:** BigQuery native tables (export to files first); GoogleSQL code. **Must be rebuilt:** IAM bindings, row access policies, policy tag taxonomies and masking rules, CMEK and EKM bindings.

Tier 2: the platforms

The three platforms hold the layers in the middle of the stack: the table format they write by default, the catalog that names the tables, the semantic layer that defines a metric, the policy engine and the compute meter. Each now reads and writes Apache Iceberg in some form, and each keeps a native path that is richer than its open one. The cards show where the two paths part.

Readers confuse three Microsoft names, so here they are in one paragraph. **Microsoft Fabric** is a software-as-a-service analytics product, generally available since 15 November 2023, billed by capacity (capacity units bought as F SKUs) and built around OneLake, one lake per tenant; Microsoft writes that "you don't need an Azure account to use Fabric". **Azure Synapse Analytics** is an Azure service organized in workspaces: its serverless SQL pool charges for data processed and queries files in Data Lake Storage with T-SQL, and Microsoft's own Synapse pages now send new warehouse users to Fabric Data Warehouse

and offer a migration path for dedicated SQL pools. **Azure Databricks** is the Databricks platform on Azure: Databricks software, Unity Catalog and DBU meters, with charges billed directly by Microsoft. They overlap: Fabric can mirror Azure Databricks catalogs into OneLake, and Azure Databricks can reach OneLake through its Data Lake Storage interface. They still sit in different places on the page, because they hold different layers.

Platform card: Databricks (tier 2) · as of November 2026

Scope and date	The Databricks platform, documented for AWS, Azure (as Azure Databricks) and Google Cloud. The control plane is in the Databricks account; classic compute runs in your cloud account, serverless compute in the Databricks account. Documentation read 20 November 2026.
Storage of record	Classic workspaces: a storage bucket in your cloud account, plus your own cloud storage for Unity Catalog tables. Serverless workspaces: a Databricks-managed default storage location. Managed table files are stored in the schema or catalog that contains them.
Table formats (native, read, write)	Native: Delta Lake by default; managed Iceberg tables with USING iceberg. Through the Iceberg REST endpoint: managed Iceberg read and write; foreign Iceberg read only; Delta with Iceberg reads enabled read only.
Catalog and its interface	Unity Catalog, three-level namespace catalog.schema.object. Implements the Iceberg REST catalog at /api/2.1/unity-catalog/iceberg-rest; an outside client needs the EXTERNAL USE SCHEMA privilege. An open-source Unity Catalog exists.
Semantic layer	Unity Catalog metric views: KPIs defined once in YAML, queried across any dimension by people, AI/BI dashboards and Genie.
Where policies are enforced	Unity Catalog privileges (ACLs by UI or SQL). Row filters and column masks are SQL functions evaluated at query time; attribute-based policies (ABAC). Tables with row filters or column masks cannot be reached through the Iceberg REST catalog or Unity REST APIs, except reads through cross-engine ABAC.
Key custody	Customer-managed keys from your cloud's key service for three uses: managed services (notebooks, secrets, SQL query data in the control plane), workspace storage, and Unity Catalog catalogs on default storage, with a separate key per catalog possible.
Compute and SQL dialect	Apache Spark and Databricks SQL on SQL warehouses; notebooks in Python, R, Scala and SQL. In Databricks SQL, ANSI_MODE defaults to TRUE for accounts created on or after 19 October 2022.
Billing meters	Databricks Units (DBUs), "a normalized unit of processing power", per second. Classic compute instances are billed by your cloud provider; Azure Databricks usage is billed by Microsoft.
What travels	**Travels:** Delta and Iceberg files in your own object storage; managed Iceberg tables to any Iceberg REST client. **Travels with work:** Delta tables to Iceberg readers (enable Iceberg reads); notebooks, jobs and Spark code; metric view YAML. **Must be rebuilt:** grants, row filters, column masks and ABAC policies; data in Databricks default storage is copied out first.

Platform card: Snowflake (tier 2) · as of November 2026

Scope and date	Snowflake on AWS, Microsoft Azure and Google Cloud; it cannot run locally or on private cloud infrastructure. Documentation read 20 November 2026.
Storage of record	Snowflake tables: loaded data is reorganized into an "internally optimized, compressed, columnar format" in cloud storage Snowflake manages. Iceberg tables: your storage, reached through an external volume; Snowflake provides no Fail-safe for them.
Table formats (native, read, write)	Native: Snowflake's internal format. Apache Iceberg tables (generally available 10 June 2024): read and write with Snowflake as the catalog (full platform support) or with an external catalog (limited platform support).
Catalog and its interface	Snowflake Horizon Catalog. Its Horizon Iceberg REST Catalog API, built on Apache Polaris, lets engines such as Spark, Trino, Dremio and DuckDB read and write Snowflake-managed Iceberg v2 and v3 tables. Snowflake Open Catalog is "a managed service for Apache Polaris". Catalog-linked databases stay in sync with a remote Iceberg REST catalog.
Semantic layer	Semantic views: schema-level objects holding facts, dimensions, metrics and relationships; Snowflake treats them as metadata.
Where policies are enforced	Row access policies (schema-level, evaluated with the policy owner's role) and masking policies applied by query rewrite at runtime, both Enterprise Edition or higher, inside Snowflake. Policy-protected tables can be reached over the Horizon REST API through a documented policy-enforcement setup for Spark; Snowflake's sample engine configuration "doesn't enforce data protection policies".
Key custody	Tri-Secret Secure (Business Critical or higher) combines a Snowflake-maintained key with a customer-managed key in the hosting cloud's key service; once that key is revoked, Snowflake can no longer decrypt the data.
Compute and SQL dialect	Virtual warehouses, independent compute clusters, running Snowflake SQL on Snowflake's own query engine; external Iceberg engines through Horizon.
Billing meters	Credits: warehouses per second with a 60-second minimum per start; serverless features by compute-hours; cloud services above 10 percent of daily warehouse use. Storage per TB per month (daily average). Data transfer per TB to another region or cloud.
What travels	**Travels:** Iceberg tables on external volumes, readable through Horizon or Open Catalog. **Travels with work:** native tables (unload to files); SQL, procedures and semantic view definitions. **Must be rebuilt:** roles and grants, row access and masking policies, Tri-Secret Secure key setup.

Platform card: Microsoft Fabric with OneLake (tier 2) · as of November 2026

Scope and date	A software-as-a-service analytics platform, generally available 15 November 2023, in a Microsoft Entra tenant: Data Factory, Data Engineering (Spark), Data Warehouse, Real-Time Intelligence, Data Science, Databases, Power BI, Fabric IQ (preview). Documentation read 20 November 2026.
Storage of record	OneLake, built on Data Lake Storage: one per tenant, it cannot be deleted or duplicated. Workspaces belong to a capacity tied to a region. Shortcuts reach Data Lake Storage, Amazon S3, Google Cloud Storage and Iceberg sources without copying. Deleted files kept seven days by default.
Table formats (native, read, write)	Native: Delta Parquet, the format in which Microsoft says Fabric's analytical engines (T-SQL, Spark, Analysis Services and others) store data in OneLake. Iceberg through metadata virtualization in both directions; Delta to Iceberg produces V2 metadata, Parquet data files only, and the latest table metadata version is converted, not the history.
Catalog and its interface	OneLake catalog for discovery and governance across the tenant; Microsoft Purview adds information protection and data loss prevention. Data access over Data Lake Storage (ADLS Gen2) APIs; metadata mirroring syncs an external catalog.
Semantic layer	Power BI semantic models, reading OneLake in Direct Lake mode without a copy; Fabric IQ (preview) adds ontology and semantic models.
Where policies are enforced	Workspace roles; OneLake security roles down to folders, tables, rows and columns, stored by OneLake and enforced in SQL, Spark and Power BI. Sensitivity labels keep their protection when data is exported.
Key custody	Microsoft-managed keys for all data at rest. Workspace customer-managed keys in Key Vault or Managed HSM wrap the data keys; revoking makes reads and writes fail within 60 minutes. Spark cluster temporary data and some item metadata are not covered.
Compute and SQL dialect	Fabric Data Warehouse in T-SQL (SQL Database Engine surface, multi-table ACID); read-only SQL analytics endpoint over each lakehouse; Spark; Analysis Services; KQL in Real-Time Intelligence.
Billing meters	Capacity units (F2 to F8192), billed per second through Azure with a one-minute minimum. OneLake storage pay-as-you-go per GB, outside capacity units; OneLake reads, writes and row-level security consume capacity units. Per-user licenses for Power BI content on SKUs below F64.
What travels	**Travels:** Delta Parquet files over Data Lake Storage APIs; the latest snapshot as virtual Iceberg metadata. **Travels with work:** T-SQL warehouse code, Spark notebooks, pipelines, semantic models. **Must be rebuilt:** OneLake security roles, workspace roles, sensitivity labels; snapshot history kept outside the virtual Iceberg view.

The control card: an open stack you assemble

The open stack has no vendor and no single bill. Its parts are an object store from any tier 1 cloud or an S3-compatible service, the Iceberg table specification, a catalog that speaks the Iceberg REST interface (Apache Polaris, whose graduation to an Apache top-level project the Incubator records on 15 February 2026, is one) and an engine such as Trino or Apache Spark. A team, or a provider it hires, runs the catalog and the engine. Its card is short where the platforms are long, and that shortness is the information.

Platform card: the self-assembled open stack (control) · as of November 2026

Scope and date	Object storage + Apache Iceberg + an Iceberg REST catalog (here Apache Polaris, top-level project since February 2026) + an open engine (Trino, Apache Spark). Project documentation read 20 November 2026.
Storage of record	Your buckets on any object store the engine can reach.
Table formats (native, read, write)	Native: Iceberg; data files Parquet by default, or Avro or ORC. Trino and Spark read and write. Snapshot expiry defaults: snapshots older than 5 days expire, one kept at minimum; tags and branches are kept indefinitely by default.
Catalog and its interface	Polaris "implements Iceberg's REST API". Spark connects with catalog type rest; Trino with iceberg.catalog.type rest.
Semantic layer	None among these parts: add a metrics layer or BI model and own it.
Where policies are enforced	Polaris role-based access control: privileges granted to catalog roles, catalog roles to principal roles, on catalogs, namespaces and tables. TABLE_READ_DATA hands the engine short-lived read-only storage credentials. Row and column rules belong to the engine or the semantic layer.
Key custody	The object store's encryption and key service, under your account.
Compute and SQL dialect	Trino, "an ANSI SQL compliant query engine"; Spark SQL, ANSI dialect on by default since Spark 4.0 (spark.sql.ansi.enabled). Both read a snapshot by id, timestamp or tag (FOR VERSION AS OF).
Billing meters	No license meter for the open-source parts; the infrastructure meters of where it runs (storage, compute, network) and the people who operate it.
What travels	**Travels:** files, Iceberg metadata, snapshots and tags, readable by any engine that speaks the Iceberg REST interface and supports the table's format version and features. **Travels with work:** moving tables to another bucket or region (metadata holds absolute paths); SQL between Trino and Spark. **Must be rebuilt:** catalog roles and grants in a new catalog; row and column rules held in the engine; runbooks.

Read the insurer's thread against these rows. On 17 February the quarter-close snapshot of the quarterly loss table carried no tag; at Iceberg's defaults an expiry run removes any untagged snapshot older than five days once newer ones exist, and the insurer's 45-day setting only delayed the removal to 15 March. A tag, kept indefinitely by default, is a cheap fix wherever the table is Iceberg, and the open stack offers it with nothing else to buy. A row policy that does business logic meets a different fate on each card: Databricks documents that tables with row filters cannot be read through its Iceberg REST endpoint, AWS refuses the snapshot metadata tables to a reader under a Lake Formation filter, and Snowflake's sample configuration for outside engines does not enforce its policies. The files travel; the rule stays home. Chapter Fifteen runs that test on the insurer's tables.

Where the cards meet

Warehouse and lakehouse features are meeting at every vendor. On 10 June 2024 Snowflake announced general availability of Apache Iceberg tables. On 3 December 2024 AWS announced S3 Tables, which it called "the first cloud object store with built-in Apache Iceberg support". On 11 June 2025 Databricks announced a public preview of full Iceberg support in Unity Catalog and, in the same release, Unity Catalog Metrics, which "brings business metric definitions traditionally embedded within BI tools to the data platform". Google documents Iceberg managed tables in BigQuery with "the same fully managed

experience as standard BigQuery tables", and Microsoft documents Fabric Data Warehouse as T-SQL with multi-table ACID transactions over Delta tables. Warehouse vendors took up open tables; lakehouse vendors took up semantic layers and warehouse transactions.

> *The files converged. The rules that change a number did not.*

OUR VIEW, AS OF NOVEMBER 2026

Convergence makes the cards look alike in the format row and leaves them different in the policy, semantic and key rows. That is where a choice is really made, and where the cost of leaving sits. Once the middle rungs are written down, pick the platform for its compute and its people; keep the table format, the tag on every published snapshot and the definition of every board metric in a form the control card could read.

Two chapters use these cards. Chapter Fifteen takes the "must be rebuilt" parts and prices them in an exit test on the insurer's board number. Chapter Sixteen maps the workload classes of Chapter One to formats, engines and tiers, records where a team chooses lock-in on purpose, and shows how to run your own benchmark.

ERA CARD

Reader. Many engines read the same files: a warehouse engine, a Spark job, a BI model and an outside engine through an Iceberg REST catalog.

Security. Policies are enforced where the vendor's engine runs; a reader that arrives through an open interface may not meet them.

Cost. Six different units (DBUs, credits, capacity units, bytes processed, slot-hours, RPUs) plus storage and egress; compare workloads, not unit prices.

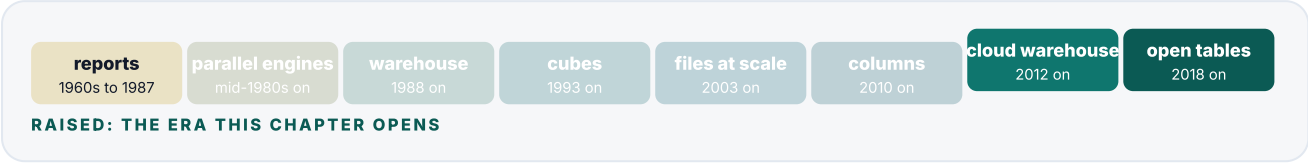
AI. An assistant quotes whichever semantic object it is pointed at: a metric view, a semantic view or a semantic model. The card's semantic row says which one holds the contract.

Monday rule: fill in the ten rows for the platform you run today, with a URL and a date on every line, and list each "must be rebuilt" item that touches a board number.

PART IV · THE PRICE

CHAPTER THIRTEEN

What Work Costs



On 20 June 2024 the FinOps Foundation announced that version 1.0 of FOCUS, the FinOps Open Cost and Usage Specification, had been "formally ratified for General Availability": one shape for the billing data that cloud and platform vendors send their customers. It answered a problem every CTO of a cloud data platform knows. The invoice arrives in the vendor's units, and nobody can say what the board's loss ratio cost to produce last quarter, or why the same query cost a hundred times more in one notebook than in one dashboard. In 2026 the insurer's CFO asked that question about the quarterly loss ratio. The platform team could show the meters. It could not show the work. The decision this chapter changes: **price the work a platform does, not the platform**.

The meters

A **billing meter** is the unit a vendor counts and charges for. Chapter Seven told how the cloud warehouse moved the cost of a question from a capital budget to a monthly invoice. Each platform chose its own unit, and the choice decides which habits of your teams are expensive. The table lists the meters a data platform bill is made of, in each vendor's own words. It gives units only; prices change often and live in the dated web companion to this book.

Meter	What is counted	Where you meet it (vendor documentation, November 2026)
Bytes processed	Data read by each query	BigQuery on-demand: "the total data processed in the columns you select", rounded up to the nearest MB, at least 10 MB per table referenced and per query
Slot capacity	Virtual compute units, reserved or autoscaled over time	BigQuery editions: slot-hours, billed per second with a one-minute minimum (fluid scaling, an opt-in, has no minimum)
Credits	Running time of a virtual warehouse, by size	Snowflake: billed per second, with a 60-second minimum each time a warehouse starts or resumes; each larger size roughly doubles the credits per hour; cloud services charged only when their daily use exceeds 10 percent of daily warehouse use
DBUs	A normalized unit of processing	Databricks: the Databricks Unit, "a normalized unit of processing power"; pay-as-you-go use billed at per-second granularity
Capacity units	A reserved pool of compute	Microsoft Fabric: CUs per F capacity (F2 to F8192), billed per second with a one-minute minimum
Processing units	Serverless compute over time	Amazon Redshift Serverless: RPUs, metered per second with a 60-second minimum

Meter	What is counted	Where you meet it (vendor documentation, November 2026)
Storage	Bytes kept, per month	Usually GB-months or TB-months, under each product's rules; BigQuery lowers the rate for tables or table partitions not modified for 90 consecutive days; Redshift bills managed storage apart from compute
Network	Bytes that leave a region or a provider	Data transfer out, cross-region replication and exports, priced by source and destination
Requests and metadata	Operations on objects and catalogs	Amazon S3 charges per request by type (PUT, COPY, POST, LIST, GET). Catalog and metadata work is metered too; Snowflake, for example, bills it inside the cloud services line above

Read down the middle column and two families appear. One family counts **what a query touches**: bytes processed. The other counts **how long compute runs**: slots, credits, DBUs, capacity units and processing units, each a reserved or elastic amount of machine for a stretch of time. Storage, network and requests sit underneath both. An open lakehouse adds the object store's request meter and the catalog's own line to whatever engine reads the tables, so one board query can appear on three invoices.

The two families reward different habits. Under a bytes meter, a team that reads fewer columns and fewer partitions pays less, mostly regardless of how long the query takes. Under a time meter, a team that keeps compute busy and lets it stop when idle pays less, mostly regardless of how much data it reads in that time. A platform card (defined in Chapter Nine and filled for seven platforms in Chapter Twelve) records the meters in its ninth line for this reason: the meter is a design decision, not a footnote.

Why one query can cost very different amounts

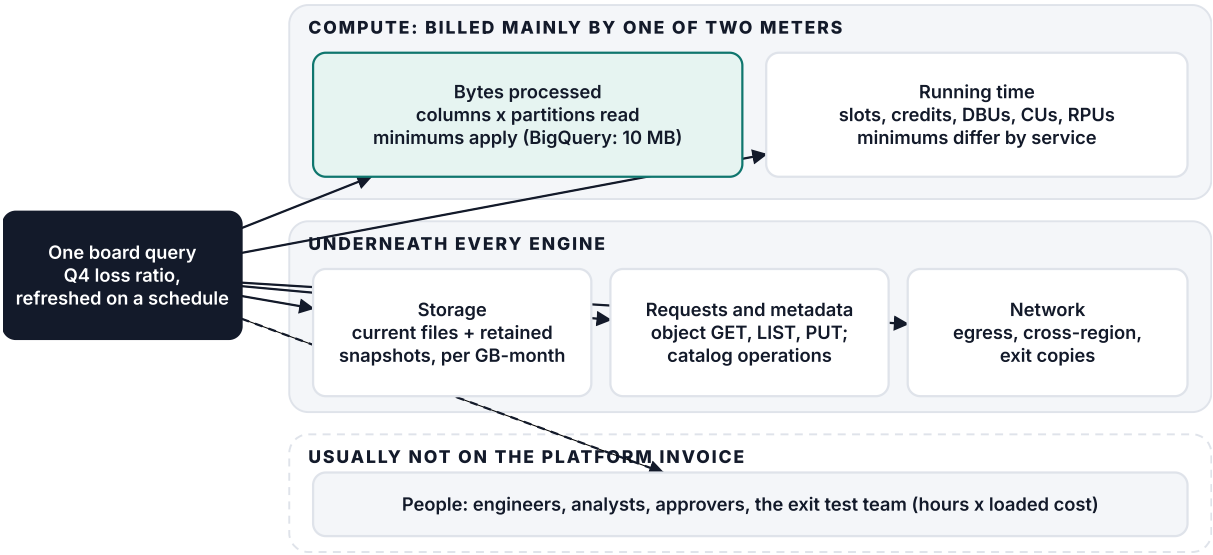
The insurer's loss-ratio query is a few dozen lines of SQL. Its cost depends less on the SQL than on what the engine has to read and how long the meter runs. Five things move it.

- **Columns.** Column files (Chapter Six) let an engine read only the columns a query names. BigQuery bills "the total data processed in the columns you select", so `SELECT *` on a wide claims table pays for every column, needed or not.
- **Partitions and pruning.** A table partitioned by accounting month lets the engine skip every month outside the quarter. A notebook that filters on a derived date the engine cannot match to the partition reads ten years to answer one quarter.
- **Limits that do not limit.** Google's documentation states that "for non-clustered tables, applying a LIMIT clause to a query doesn't affect the amount of data that is read". A preview of ten rows can bill a full scan.
- **Minimums and caches.** BigQuery on-demand charges at least 10 MB per table referenced. Several time meters charge at least one minute: Snowflake each time a warehouse starts or resumes, Redshift Serverless for each period of activity, and BigQuery editions and Fabric by default. Databricks documents per-second billing and states no minimum. BigQuery does not charge for results served from its cache. A dashboard that refreshes small queries often can cost more in minimums than in work.
- **Work you did not mean to do.** AWS documents that a cancelled Redshift Serverless query is billed for the time it ran, and that health-check queries from connection pools are billable activity. Google documents that cancelling a running BigQuery job "might incur charges up to the full cost for the query".

An open table format adds two more. Many small files mean many object requests and more metadata to plan with, which the request meter counts. Snapshots (Chapter Eight) keep old data files until they expire, so a retention policy is also a storage line: tagging the quarter-close snapshot, as Chapter Seventeen recommends, keeps those files and their cost on purpose.

FIGURE 13.1

One query, several meters: compute is billed by bytes or by time, and storage, requests, network and people sit underneath



Which compute meter applies decides which habits are expensive: a bytes meter punishes wide scans, a time meter punishes idle compute; storage, requests and network can add charges under either.

Author's diagram based on vendor billing documentation (Google BigQuery, Snowflake, Databricks, Microsoft Fabric, Amazon Redshift Serverless, Amazon S3), checked 20 November 2026; units only, no prices.

FOCUS: one shape for the bill

Each vendor's billing export used to have its own columns, names and grain. FOCUS gives them a common schema, which providers publish or customers convert to. The specification's site describes it as formed "to establish an open, consensus-driven standard for billing data"; it is a Joint Development Foundation project supported by the FinOps Foundation. Version 1.0 was ratified on 20 June 2024. The current version is 1.4: its steering committee ratified it on 4 June 2026, and it added Invoice Detail and Billing Period datasets so usage rows can be reconciled with invoices.

A FOCUS row carries columns such as Billed Cost, Effective Cost, List Cost, Consumed Quantity, Consumed Unit, Pricing Unit, Service Name, Charge Category, Resource ID, Charge Period Start and Tags. Two of them matter most to a data leader. **Consumed Quantity** with its **Consumed Unit** keeps the meter (bytes, slot-hours, credits) next to the money, so a team can test whether cost rose because use rose or because a rate, a discount or a commitment changed. **Tags** carry the labels your own teams attach to resources, and they are the bridge to the work.

FOCUS describes what was billed. It does not know why. A warehouse that runs the loss-ratio job and forty other jobs typically appears as one billed resource with one cost. To price a board number, a team joins the FOCUS rows to the platform's query history, where each query carries the metric id from its passport (Chapter One) as a query tag or label, and shares the warehouse's time across the queries that used it. The arithmetic is simple. The discipline is tagging every scheduled query with the metric it serves, from the day it is written.

> **WHAT HASN'T CHANGED**
>
> The thread's first Ledger row, in 1987, priced the loss report as machine time on a shared mainframe plus about two actuary days a month for hand adjustments (illustrative). Shared machines have always needed a rule for splitting their cost across the work, and the people beside the machine have always been part of the price.

A worked cost model

The model below prices one workload: the dashboard that shows the board loss ratio, refreshed every 15 minutes, against the claims table in the insurer's lakehouse. Every quantity is invented for the example and every rate is a symbol. Where a column shows cost units, the rate is an illustrative one cost unit per terabyte scanned, chosen to make the ratios readable; it is not a vendor price.

Assumptions (illustrative). The claims table holds ten years in 2.4 TB of compressed Parquet, partitioned by accounting month. One quarter is about 60 GB. The board query needs 9 of the table's 64 columns, among them the widest (amounts and reserve histories), about 18 GB of the quarter's bytes. The quarter-close result, kept as a small table at the tagged snapshot, is about 50 MB. The dashboard refreshes 96 times a day, 2,880 times in a 30-day month.

Way the query runs	Read per run	Read per month	Bytes meter (cost units)	Time meter (compute hours)
A. Unpartitioned filter, all columns	2.4 TB	6,912 TB	6,912	272
B. Quarter partition, 9 columns	18 GB	52 TB	52	242
C. Quarter-close table at the tag	50 MB	0.14 TB	0.14	241

The bytes column is the product of reads and runs: A costs about 133 times B for the same number. The time column assumes an engine that bills by running time, needs 40 seconds for A, 3 seconds for B and 1 second for C, and suspends after five idle minutes. With a refresh every 15 minutes it wakes for every run, so each run bills the query plus 300 seconds of idle before suspension: about 340, 303 and 301 seconds, which over 2,880 runs is the 272, 242 and 241 hours in the table. Under that meter the three ways differ by about 13 percent, because idle time before suspension, not the query, is most of the bill. On the time meter the cheap fix is the suspend setting or a refresh that runs once after close; on the bytes meter it is the partition and the column list. The same dashboard needs different fixes on different platforms.

The rest of the workload adds lines that do not depend on how the query is written:

- **Storage:** (2.4 TB of current files + retained snapshots, about 0.3 TB with the quarter-close tags) × the storage rate per TB-month.
- **Requests:** object reads and lists per month × the request rate; it grows with the number of small files, which compaction reduces.
- **Catalog and metadata:** the catalog's own meter, where it has one, or its share of a platform fee.
- **Network:** zero in a normal month inside one region; in the month of an exit test, the copy's terabytes × the transfer rate (Chapter Fifteen).
- **People:** the hours below × a loaded hourly cost.

Fill the rates from the dated web companion or your own contract, keep the quantities from your platform's query history, and write the total into the cost column of the Ledger row for the board figure with its basis. A number in that column without a basis is a guess.

Splitting a shared bill. On a time meter the warehouse is one resource and the dashboard is one of its users. Three rules for sharing its cost are common: by each query's share of execution time, by its share of bytes read, or by reserving a separate warehouse or capacity for the workload so its bill stands alone. Each rule gives a different price for the same board number, and each can be defensible if it fits the purpose of the calculation and is applied the same way every period. What is wrong is to change the rule from quarter to quarter, or to leave idle time unassigned, so that the cheapest workload on paper is the one that keeps the warehouse awake. Write the rule into the cost column of the board figure's Ledger row, as part of its basis, and keep it until you decide to change it on purpose.

FOCUS helps with one more distinction. Its rows carry four costs: List Cost (at public rates), Contracted Cost (at the negotiated rates), Effective Cost (after commitments and discounts are spread over the usage they cover) and Billed Cost (what the invoice for the period

shows). A commitment bought for the year makes the invoice smooth and the marginal query look free; the work still used capacity that someone paid for.

People are a cost line

The invoice shows machines. The work also needs people: platform engineers who keep pipelines, tables and catalogs healthy; an analyst who reconciles the bill; the actuary and the controller who check the number before it is approved; and, at an exit test, the engineers who port SQL and rebuild policies. This book gives no share of total cost for people: I found no primary source that gives one that would hold across companies. Count them in hours per workload, as the thread's 1987 row did, and put them on the same page as the meters.

Two patterns are worth watching. Engineering hours spent tuning queries to a bytes meter are a real saving only while the meter stays; a move to a time-billed engine can make some of that work worthless and create new work in sizing and suspension. And the hours to rebuild what does not travel (policies, grants, the semantic layer) are invisible on every monthly invoice until the day of the exit, which is why Chapter Fifteen prices them in advance.

On each platform. Bytes-billed engines make the column list and the partition filter the main cost control; time-billed warehouses make sizing, concurrency and suspension the main cost control; capacity platforms make the cost of a query visible mainly as contention with other work in the same capacity, so the price of a heavy notebook is paid in a slower dashboard rather than on the invoice.

> **ERA CARD**
>
> **Reader.** The CFO and the platform owner reading one bill, joined by a FinOps analyst who maps meters to work.
>
> **Security.** Billing exports name resources, projects and tags; they are internal data with their own grants, and tags must never carry personal data.
>
> **Cost.** Meters count bytes or time; the same query can differ by two orders of magnitude on one meter and barely at all on another.
>
> **AI.** An assistant that writes SQL on request runs queries nobody reviewed; give it a budget, a tag and the metric's quarter-close table rather than the raw claims table.

Monday rule: tag every scheduled query that feeds a board number with that number's metric id, and ask for next month's bill as FOCUS rows joined to those tags.

CHAPTER FOURTEEN

Location, Transfer and Jurisdiction



On 16 July 2020 the Court of Justice of the European Union, sitting as a Grand Chamber in Case C-311/18 (known as Schrems II), declared invalid the Privacy Shield decision under which companies had sent personal data from the EU to the United States. It left the Commission's standard contractual clauses standing, with a duty on the exporter to verify, case by case, whether the law of the destination country ensures adequate protection. Many boards heard the ruling as "keep the data in Europe". That is not what it said, and it is not what the law says. The decision this chapter changes: **for each dataset, record three separate facts: where it is stored, who can reach it from where, and on what basis it is transferred**.

Location is not transfer

The General Data Protection Regulation (GDPR, Regulation (EU) 2016/679) governs international data in Chapter V, Articles 44 to 50. Article 44 sets the principle: a transfer of personal data to a third country "shall take place only if" the conditions of the chapter are met. Article 45 allows transfers to countries with an adequacy decision; Article 46 allows them with appropriate safeguards, such as standard data protection clauses; Article 47 covers binding corporate rules inside a group; Article 48 deals with orders from foreign courts and authorities; Article 49 lists narrow derogations; Article 50 covers cooperation between authorities.

The chapter regulates **transfers**. It contains no general rule that personal data must be stored inside the EU. Data residency, keeping data in a chosen region, is a contract term, a security design or a sector rule, and it is often a good choice. It is not, by itself, what Chapter V asks for, and a dataset stored in Frankfurt can still be transferred.

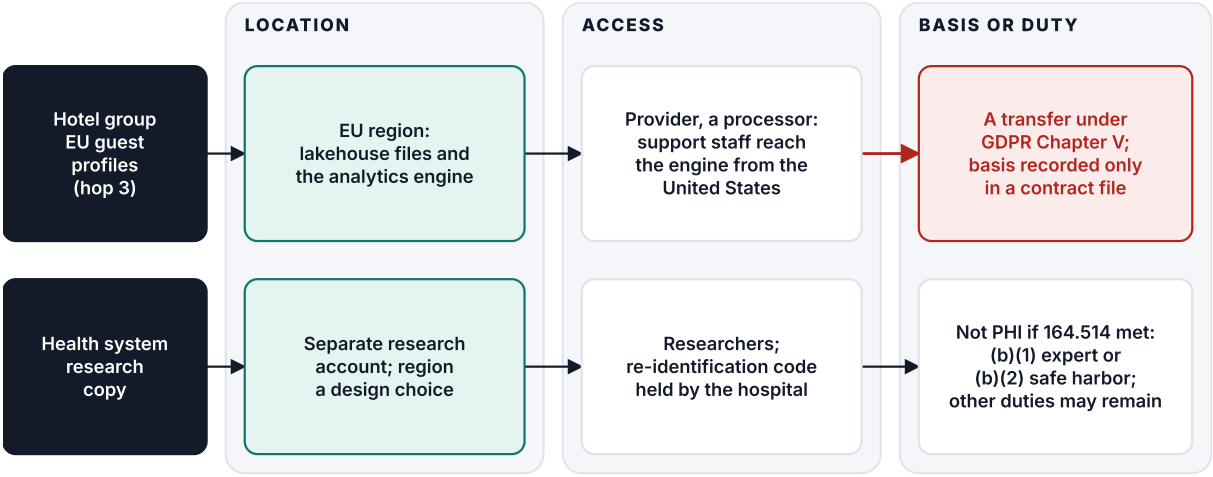
The GDPR does not define "transfer". The European Data Protection Board filled the gap in its Guidelines 05/2021 on the interplay between Article 3 and Chapter V (version 2.0, adopted 14 February 2023). Its title names the two parts of the law it joins: Article 3, which sets the territorial scope of the GDPR, and Chapter V. It set three cumulative criteria: an

exporter subject to the GDPR for the processing; who discloses or otherwise makes the data available to another controller, joint controller or processor; who is in a third country, whether or not that importer is itself subject to the GDPR, or is an international organisation. The guidelines then make the point that matters most for a cloud platform: remote access from a third country, "even if it takes place only by means of displaying personal data on a screen, for example in support situations, troubleshooting or for administration purposes", is a transfer when the three criteria are met. The same guidelines give a counter-example: an employee who takes a laptop on a business trip abroad and works on the employer's data there is not a transfer, because the employee is part of the controller, not another party.

So three facts must be kept apart. **Location** is where the bytes sit: a region, a data center, a country. **Access** is who can read them and from where, including support staff and service operators. **Transfer** is the legal event that happens when data is made available to another party in a third country, and it needs a basis under Articles 45 to 49. A platform card's "storage of record" line answers the first. The line on where policies are enforced answers the second. The third lives in contracts and records of processing, and it is the one most often missing from the platform's own catalog.

FIGURE 14.1

Location, access and transfer are three facts: the guest files and engine never left the EU, and the research copy, once de-identified, can leave the hospital's account



Moving files decides location; identities and support decide access; the transfer basis or the de-identification method decides the duty under each law, and all three are written into the classification column of the dataset's Ledger row.

Author's diagram based on GDPR Chapter V, EDPB Guidelines 05/2021 (version 2.0), 45 CFR 164.514 and the book's companies (the hotel group's second Ledger, hop 3; the health system).

Schrems II, the Data Privacy Framework and the appeal

Schrems II did two things. It invalidated the Privacy Shield decision (Decision (EU) 2016/1250), and it kept standard contractual clauses as a valid tool while requiring the exporter, case by case, to check whether the destination's law lets the importer keep its promises. After the ruling, many transfers to the United States ran on standard clauses plus a written assessment of the destination.

On 10 July 2023 the Commission adopted Implementing Decision (EU) 2023/1795, finding that the United States ensures an adequate level of protection for personal data transferred to organisations that join the EU-US Data Privacy Framework. Philippe Latombe, described in the court's press release as a French citizen and user of various IT platforms that collect his personal data and transfer them to the United States, asked the General Court to annul it. On 3 September 2025 the General Court dismissed the action in Case T-553/23; its press release says the court confirmed that, on the date the decision was adopted, the United States ensured an adequate level of protection. On 31 October 2025 Mr Latombe brought an appeal to the Court of Justice, registered as Case C-703/25 P. The state of that appeal is a moving date: check the court's case register, linked from the dated web companion, before you rely on the framework for a new dataset.

For a data platform the practical lesson is design, not prediction. A transfer that rests on one adequacy decision has one point of failure. Record the basis for each transfer in the Ledger row for the dataset, with the date you last checked it, and know which fallback (standard clauses with an assessment, or a change of design) you would use if the basis fell.

> **WHAT HASN'T CHANGED**
>
> Schrems II was the second time the Court of Justice struck down an EU-US transfer arrangement; on 6 October 2015, in Case C-362/14, brought by the same complainant, it had declared the Safe Harbour decision invalid. Neither judgment moved a file. What changed overnight was the legal basis the affected transfers relied on, which is why the basis belongs beside the dataset, not only in the legal department's folder.

Jurisdiction and the sovereign offers

The United States CLOUD Act (the Clarifying Lawful Overseas Use of Data Act, Division V of Public Law 115-141, approved 23 March 2018) is often described more broadly than its text. Stated narrowly, it does three things. It added section 2713 to title 18 of the US Code: a provider of electronic communication or remote computing services must meet its existing obligations under that chapter to preserve or disclose data "within such provider's possession, custody, or control, regardless of whether" the data is located inside or outside

the United States. It gave providers a route to ask a court to modify or quash such process where the customer is not a United States person and does not reside in the United States, and disclosure would create a material risk of violating the laws of a qualifying foreign government, which the statute defines as one with an executive agreement in force under section 2523. And it authorised those agreements. The Department of Justice lists agreements signed with the United Kingdom and Australia; for the EU it lists negotiations, resumed in March 2023, and no agreement, so an EU customer cannot count on that route today. The Act does not create a new way for the government to read data without the legal process the chapter already required.

On the EU side, GDPR Article 48 says that a third-country court order to transfer personal data is recognised or enforceable only if based on an international agreement, such as a mutual legal assistance treaty, "without prejudice to other grounds for transfer pursuant to this Chapter". A provider subject to both laws can face a conflict. That conflict is a fact about the **provider's** jurisdiction, not about the region where the files sit, and it is what vendors' sovereign offers try to address.

Two examples, in the vendors' own words. On 14 January 2026 AWS announced the general availability of the AWS European Sovereign Cloud, which it describes as "a physically and logically separate cloud infrastructure, with all components located entirely within the EU", with its first Region in Brandenburg, Germany, and that "will be operated exclusively by EU residents located in the EU". On 26 February 2025 Microsoft announced that it had completed the EU Data Boundary for the Microsoft Cloud, under which customer data and pseudonymized personal data for its core cloud services, including most Azure services, are stored and processed in the EU and EFTA. Other providers and national partners offer comparable arrangements; the web companion keeps a dated list.

Read such offers against the three facts. They change **location** and the **operator's** access. They do not change your own access: if your analytics team or a contractor in a third country reads the data, you have made a transfer that needs a basis, whatever the region. They are also new platforms in the sense of Chapter Nine, with their own service lists and dates, so a move into one is a hop that deserves a Ledger row.

The hotel group's EU guests

The hotel and retail group runs about 60 hotels, several in the EU, and keeps a profile for each guest: stays, preferences, contact details and marketing consent. Its second Ledger follows that dataset across four platforms. The first hop loaded profiles from each hotel's property management system into a cloud warehouse in an EU region; consent had been captured hotel by hotel, with different wording and purposes, and the load mapped it to a single flag, so the purpose of each consent did not travel. The second hop came with the

acquisition of a retail chain that ran on a second cloud: guests and loyalty members were matched by email into a lakehouse, and one person could end up with two keys. The third hop is the one this chapter is about. The files stayed in the EU region, and so did the analytics engine the group bought to run models over them, but the provider's support staff can reach that engine from the United States. The provider is a separate processor, and its support staff work in a third country, so under the EDPB's three criteria that access is a transfer: the files and the engine stay in the EU, and the access is the transfer. The basis existed, but it was recorded only in a contract file that nobody had linked to the dataset, so the platform team could not see it and the catalog said "EU only". The fourth hop fed pricing-model features and an assistant over the semantic layer, which read the consent flag as "contactable" for every purpose.

EU guest profiles: one dataset, four platforms

Hop	From	To	What was lost or assumed
1	Property management systems in EU hotels	Cloud warehouse, EU region	RIGHTS **Marketing consent captured per hotel, mapped to one flag; the purpose of each consent not carried**
2	Cloud warehouse, EU region	Lakehouse after the acquisition of a retail chain (second cloud)	IDENTITY **Guests and loyalty members matched by email; two keys for one person**
3	Lakehouse, EU region	Analytics engine in the EU region, operated by a provider whose support staff can access it from the United States	RIGHTS **Access from outside the EU is a transfer even though the files stay in the EU region; transfer basis recorded only in a contract file**
4	Lakehouse	Pricing model features and an assistant over the semantic layer	RIGHTS **Consent flag read as 'contactable' for all purposes**

The repair is not a move. It is three facts written for each hop into the Ledger's existing classification column, beside the legal basis it already holds: the location (region and provider), the access path (which identities, from which countries, including the provider's support), and the transfer basis with its date. With those facts, the group found that hop 3 needed a recorded basis and hop 4 needed the consent purposes restored, and that neither problem would have been solved by moving the files to a different region.

The health system's research copy

In the United States, the HIPAA Privacy and Security Rules govern protected health information (PHI) held by covered entities and their business associates. The Department of Health and Human Services' Office for Civil Rights addresses cloud services in its *Guidance on HIPAA and Cloud Computing*. A cloud service provider that creates, receives, maintains or transmits electronic PHI for a covered entity is a business associate, and needs a business associate agreement (BAA), "even if the CSP processes or stores only encrypted ePHI and lacks an encryption key". The same guidance answers the location question directly: the

HIPAA Rules allow a covered entity to use a provider that stores electronic PHI on servers outside the United States, provided there is a BAA and the other requirements are met. As with the GDPR, the duty attaches to the relationship and the safeguards, not to a map.

The health system keeps imaging studies (DICOM files, some with patient names burned into the pixels) and FHIR resources in its lakehouse, with PHI tags on columns and files (Chapter Eleven). A BAA covers the provider where it is a business associate for the services that hold PHI; it is not a blanket agreement with every service a cloud sells. Its researchers want a copy they can use with fewer restrictions.

The rule for that copy is 45 CFR 164.514. Health information that does not identify an individual, and gives no reasonable basis to believe it could, "is not individually identifiable health information". The regulation gives two ways to get there. **Expert determination**, in paragraph (b)(1): a qualified person applies statistical and scientific methods, finds that the risk of identification by an anticipated recipient is very small, and documents the methods and results. **Safe harbor**, in paragraph (b)(2): eighteen listed identifiers, (A) to (R), are removed, and the covered entity has no actual knowledge that what remains could identify someone. Names are identifier (A), and identifier (Q) is "full face photographic images and any comparable images", which is why names burned into imaging pixels, and faces in the images themselves, are a de-identification task, not a metadata task. Paragraph (c) lets the covered entity keep a re-identification code, provided it is not derived from or related to information about the individual and cannot otherwise be translated to identify the individual, and provided the entity does not use or disclose the code for any other purpose or disclose the mechanism for re-identification.

Done properly, the research copy is a different dataset with different duties: it can sit in another account or region, under another agreement, because, for HIPAA purposes, a properly de-identified copy is no longer PHI, though data use agreements, other laws and the hospital's own policies may still apply. Done carelessly, it is PHI in a place without a BAA. The difference is invisible in the files. It lives in the Ledger row for the copy: the method used, the expert's report or the safe harbor checklist, the date, the owner, and whether a re-identification code exists and who holds it. The rule of this series is that every copy has a named owner and a known deletion reach; the research copy is a copy, and it gets its own row.

ERA CARD

Reader. The data protection officer and the privacy counsel beside the platform owner, reading the same Ledger row.

Security. Who can reach data from where is a policy and identity fact; support and operator access count.

> **Cost.** Residency and sovereign regions can narrow the service list and change prices; record the reason so the cost is a decision, not an accident.
>
> **AI.** A model or assistant run by a provider in a third country that reads EU personal data makes a transfer when the EDPB's three criteria are met, even when the training files stay put.

Monday rule: for the five datasets with the most personal data, write three facts into the classification column of their Ledger rows (location, access path from where, transfer basis with the date checked) and fill them from named sources.

CHAPTER FIFTEEN

Migration and the Price of Leaving



On 11 January 2024 Google Cloud announced that it would waive data transfer fees, under conditions, for customers moving their data off its cloud. On 5 March 2024 AWS followed with a waiver, on request and as credits, of data transfer out to the internet for customers moving outside AWS, and Microsoft publishes a similar programme for Azure. Each press line read like an open door. In June 2026 the insurer tested what leaving would take, without leaving: it copied the files behind its board loss ratio to object storage in a second region, registered them in an open catalog and asked an open engine for the Q4 2025 figure. The files arrived intact. The number did not: the engine returned 62.5 percent where the board had approved 61.8 percent (approved 5 February, data to 23 January). The decision this chapter changes: **an exit is a test you run on one important number, not a clause you read in a contract**.

Egress and the 2024 waivers

Egress is data leaving a provider's network, and **egress charges** are the fees for it; a copy between two regions of one provider is billed as data transfer too. They are one line among the meters of Chapter Thirteen, small in a normal month and large in the month you copy everything. The 2024 waivers remove that line for a switch, under conditions that each provider sets on its own pages. They are worth reading before you plan, because they decide the order of the work.

- **Google Cloud** (11 January 2024): submit an Exit Notice; within 14 days a billing support agent is assigned; you then have a 30-day Initiation Period to start the migration and a Migration Period of at least 30 days to finish it; you submit a Completion Notice, the agreement for the services you leave terminates, and the final invoice carries a credit for the transfer. An editor's note on the announcement says free exit does not require leaving Google Cloud entirely: it can be service by service.

- **AWS** (5 March 2024): request through AWS Support; credits once approved; no need to close the account. The page has been updated twice. On 30 September 2025 AWS removed the requirement to exceed 100 GB of transfer out per month. On 8 October 2026 it stated that the waiver applies when a customer moves all its data off AWS from all eligible accounts, or all its data off one AWS service; that customers outside the EU have 180 days to finish; and that EU and UK customers follow the AWS addenda written for the EU Data Act and for UK switching.
- **Microsoft Azure**: Microsoft's pages describe free egress for customers leaving Azure over the internet for another provider or their own data center. The customer opens a support request with the subscription id, the start date and an estimate of the volume; has 60 days from that start date (180 days for UK billing addresses transferring from UK data centers); cancels all Azure subscriptions associated with the account; and then claims an invoice-level credit. UK customers may also leave single services and use Microsoft's own network for the transfer.

Read the conditions together and three limits appear. The waivers are about **leaving**: running two clouds in parallel, or copying data out for a test while you stay, is a different case, priced as ordinary egress unless a provider says otherwise. They cover the **transfer**, not the compute to read and rewrite the data, the second platform's storage, or the engineering. And they run on **notices and deadlines**: a team that files an exit notice before it knows what must be rebuilt can run out of migration period. The insurer's June test was not an exit, so its copy was billed as ordinary data transfer and recorded in the hop's cost column.

The law: the EU Data Act and the UK investigation

The EU Data Act (Regulation (EU) 2023/2854) has applied since 12 September 2025. Its Chapter VI, "Switching between data processing services", Articles 23 to 31, sets duties for cloud and platform providers serving EU customers. Article 23 requires providers to remove obstacles to switching to another provider or to on-premises infrastructure. Article 25 requires a written contract with, among other terms, a maximum notice period to start the switch of no more than two months, and a transitional period of at most 30 calendar days in which the provider keeps the service running and assists; where 30 days is technically unfeasible the provider may set an alternative period of up to seven months, and the customer may extend the period once. Article 29 phases out the charges: from 11 January 2024 to 12 January 2027 providers may impose only reduced switching charges that "shall not exceed the costs incurred by the provider of data processing services that are directly linked to the switching process concerned", and from 12 January 2027 no switching charges at all. Article 2(36) defines switching charges to include data egress charges. Article 31 lifts

some of these duties (Article 23, point (d), Article 29 and Article 30(1) and (3)) for services mostly custom-built for one customer and not offered at broad commercial scale, and the whole chapter for non-production versions offered for testing for a limited period.

Two articles keep expectations honest. Article 34(2), outside Chapter VI, covers **in-parallel use** of two providers: there, providers "may impose data egress charges, but only for the purpose of passing on egress costs incurred, without exceeding such costs". So egress for multi-cloud does not disappear in 2027; it moves to cost. And Article 2(38) defines the **exportable data** a switch must carry as input and output data, including metadata, generated by the customer's use of the service, "excluding any assets or data protected by intellectual property rights, or constituting a trade secret" of the provider or third parties. Your tables, and the metadata your use of the service generated, are in scope, subject to those exclusions. The provider's engine, optimizer and proprietary policy language ordinarily are not.

The Act is a moving text. A Commission proposal of 19 November 2025, the Digital Omnibus (COM(2025) 837), would amend it among other laws; on 20 November 2026 the European Parliament's procedure file shows it awaiting committee decision, with no amending act published. The articles above are quoted from the text in force on that date.

In the United Kingdom, the Competition and Markets Authority opened a market investigation into cloud services on 5 October 2023 and published its final decision on 31 July 2025. It found egress fees to be a barrier to switching and to multi-cloud, and recommended that the CMA prioritise strategic market status (SMS) investigations into Microsoft and AWS under its digital markets powers. The follow-up has taken two paths, according to the CMA's update of 16 October 2026. In March 2026 Amazon and Microsoft announced steps to lower egress fees and improve interoperability, which the CMA reviews. On 14 May 2026 the CMA opened an SMS investigation into Microsoft's business software ecosystem, and on 16 October 2026 it consulted on a proposed decision to designate Microsoft. The CMA says it will next focus on identity and access management across clouds and on the portability of Kubernetes containers, and its board will consider progress in spring 2027.

None of this makes leaving free. It makes the transfer cheaper and the contract clearer. The rest of the price is the work.

Migration mechanics

Each platform move in this book's thread, from the 1994 warehouse to the 2024 lakehouse, went through some version of five steps, whether the team named them or not. Naming them makes them plannable.

- **Parallel run.** Old and new platforms run the same workloads on the same inputs for a fixed period. The period should cover at least one close of every number the board reads, because quarter-end logic runs four times a year.
- **Reconciliation.** Compare results, not row counts. For each board number, compare the value with its five times: same effective date, same knowledge cutoff, same snapshot or tag, same calculation version. A match on rows with a mismatch on the number is the finding that matters.
- **Cutover.** One dated moment when readers, schedules and writers move, recorded as a hop in the Ledger with its owner. Writers move once: the old platform stops accepting writes before the new one starts, because two systems both accepting writes is how a migration creates two truths.
- **Rollback.** A written way back, tested before cutover, with the date after which it expires. After the first new write that the old platform cannot accept, rollback becomes a second migration.
- **Decommission.** Retire the old platform: grants revoked, schedules stopped, data deleted or archived under its retention rule, keys destroyed where deletion depends on them. The rule of this series: every copy has a named owner and a known deletion reach, and a decommission that leaves an untagged copy in a forgotten account has not finished.

The thread shows what skipping a step costs. In 2008 the appliance migration left history before 1995 on tape in a layout nobody could load (Chapter Four): a decommission without an archive test. In 2021 the cloud warehouse migration reconciled rows, not the status code meaning (Chapter Seven): a parallel run that compared counts.

What travels and what must be rebuilt

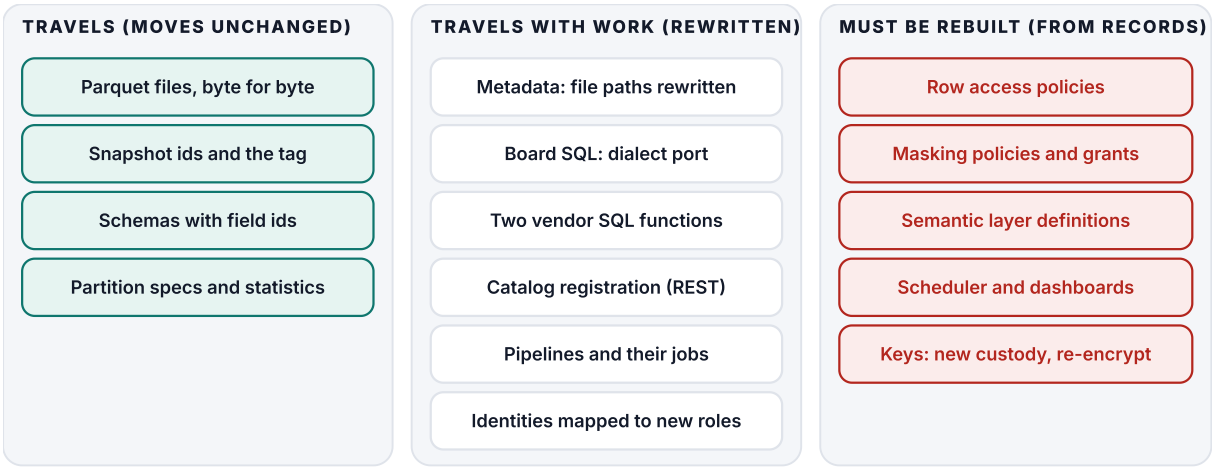
Open table formats changed the first column of an exit plan. In 2008 the data itself had to be unloaded and converted. In 2026 the insurer's Parquet data files could be copied byte for byte, with no conversion. The Iceberg metadata needed one piece of work: its absolute file paths were rewritten for the new location, and the snapshots, schemas and tag it records came through unchanged, so another engine read the tables at the tag. That is real progress, and it is the part vendors' exit pages are about.

The other two columns did not shrink. Some things **travel with work**, which in this book means they move after rewriting: table metadata with its file paths rewritten, SQL ported to another dialect, vendor-specific functions rewritten, tables registered in a new catalog, pipelines and their jobs re-pointed, identities mapped to new roles. Some things **must be rebuilt**, recreated from records, because nothing on the source side exports them in a form the destination can enforce: row access and masking policies, grants, the semantic layer's metric definitions, dashboards and the scheduler, and key custody, where data is re-

encrypted under keys the new platform's custodian holds. The platform card's tenth line (the card is defined in Chapter Nine and filled for seven platforms in Chapter Twelve) records these three states for each platform, and the Data Act's definition of exportable data explains part of why the third column exists.

FIGURE 15.1

What travels on exit: the data files moved byte for byte; metadata and SQL moved after rewriting; meaning and control had to be rebuilt



In the insurer's June 2026 exit test the data files were copied byte for byte and the snapshots, schemas and tag came through unchanged; the metadata files were rewritten for their paths and the SQL was ported; the right column had no export the destination could enforce, so a row policy that was doing business logic disappeared.

Author's diagram based on the book's thread (hop 7, exit test, June 2026), the Apache Iceberg table specification and its rewrite_table_path procedure (release 1.8), and Regulation (EU) 2023/2854, Art. 2(38) (exportable data).

The **exit inventory** (Appendix A) turns the figure into a priced list. Its rows are: files; table metadata; catalog and grants; policies; keys; code (dialect, procedures, user-defined functions); semantic layer and dashboards; pipelines; identity; runbooks and people; contracts and egress. For each row a team writes where it lives now, which of the three states it is in, the effort in person-days, the charges (egress, licences, a period of running both, with the notices, deadlines and credits that govern them), the owner and the evidence. Price it with the cost model of Chapter Thirteen. The rows that cost most are rarely files.

The exit test: 62.5, then 61.8

The insurer's exit test in June 2026 was the seventh hop of the thread. The platform team copied the Iceberg data and metadata files of the claims and premium tables to object storage in a second region, and paid egress on the copy. The data files arrived byte for byte. Before registering the tables in an open catalog through the Iceberg REST interface, the team rewrote the absolute file paths in the metadata files, manifest lists and manifests for the new location, with the path-rewrite procedure Iceberg has offered since release 1.8. It then ran the board query on an open SQL engine at the quarter-close tag `q4-2025-close`, placed on 24 February on the snapshot behind the approved figure (effective 31 December 2025, claims data received by 23 January 2026). The data files travelled; the metadata, the metric's SQL and two vendor-specific SQL functions travelled with work; row access policies, masking policies, grants, the semantic layer's metric definitions and the scheduler stayed in the vendor platform and must be rebuilt.

The first run returned 62.5 percent (exact 62.480). One thing explained it: the rows. A row access policy in the vendor catalog had been hiding the intercompany pooling rows (business assumed from the affiliate: 10,000 of premium and 7,823 of losses, in thousands of dollars, illustrative) from every role except reinsurance accounting, so neither the board view nor the analyst's notebook had ever seen them. The close job of 26 January had run on the platform's SQL warehouse engine and read the tables through the vendor catalog, which is why that row policy filtered the board figure. The contract reports that business separately, so the exclusion was right, but it lived in a security feature, not in the metric. On the second engine no policy applied, and the rows came back.

Fix 1 wrote the intercompany exclusion into the metric's SQL and into the semantic layer, where an auditor can read it. The exact value fell back to 61.850, the board's own exact value at the tag, but the engine printed 61.9. The old close job had rounded with a vendor-specific half-to-even function; the port had mapped it to the second engine's ordinary rounding function, which rounds a tie away from zero, where the contract says half to even. That difference had been there from the start. It was invisible in the first run because 62.480 prints as 62.5 under either rule, and it showed only once the rows were fixed. Fix 2 pinned the rounding rule in the metric definition, and the second run returned 61.8, matching the approved figure. Then the team ran one more query: with the 2021 population filter also corrected, so that status 7 policies are counted as the 2019 contract requires, the same engine returned 62.2. That confirmed, on a second engine, the figure the insurer had already restated for Q4 2025 in its Q1 2026 board pack of 12 May 2026, after the bridge analysis of February (Chapters Eight and Seventeen).

TYPE CONTRACT

Type, precision and rounding. Inputs as decimals; the ratio computed on unrounded parts; percent to one decimal, rounded half to even, once, at the end.

Encoding and serialization. Parquet decimals in Iceberg tables; the result exported as text with one decimal and its exact value beside it.

Unit or currency. Numerator and denominator in thousands of US dollars; the result in percent.

Time basis. Effective 31 December 2025; knowledge cutoff 23 January 2026; read at tag q4-2025-close, never at the latest snapshot.

Identity scope. The insurer's direct business, net of ceded reinsurance; status 7 policies included; business assumed from the affiliate excluded in the metric's SQL, not by a row policy.

Absence. A policy with no earned premium adds zero to the denominator and still adds its losses; a missing row is an error, not a zero.

Authority. CFO owner, appointed actuary approver; the metric contract is the authority, one version-controlled SQL definition implements it and the semantic layer publishes it; an engine's default is not part of it.

FIGURE 15.2

The exit test at tag q4-2025-close: 62.5 on the first run, 61.8 after two fixes, 62.2 with the population corrected

Step (in this order)	Exact	Printed	What changed
Board figure on the vendor platform (approved 5 Feb 2026, data to 23 Jan 2026)	61.850	61.8	Baseline: contract rounding; 2021 population filter still applied
First run (June 2026): second engine at the tag; intercompany pooling rows the vendor row policy had hidden are back	62.480	62.5	Row policy did not travel
Fix 1: intercompany exclusion written into the metric's SQL; the engine rounds half up	61.850	61.9	Rounding rule did not travel; visible only now
Fix 2: rounding pinned to half to even (second run)	61.850	61.8	Matches the approved figure
With the 2021 population filter also corrected (status 7 included)	62.158	62.2	Confirms the restatement in the Q1 2026 pack of 12 May 2026

Steps in the order they happened, all at tag q4-2025-close (effective 31 December 2025, claims data to 23 January 2026); 62.480 prints as 62.5 under either rounding rule, so the rounding difference appeared only after fix 1, and the exact column, not the printed one, is the one to compare.

Author's illustrative figures for the book's thread (synthetic Q4 2025 rows, thousands of dollars); the bridge can be rebuilt from the table in Appendix A.

THE HOP · 7 · 2026 · LAKEHOUSE (VENDOR CATALOG, FIRST REGION) TO EXIT TEST: COPIED FILES IN A SECOND REGION, AN OPEN CATALOG (ICEBERG REST), AN OPEN SQL ENGINE

Record	Property loss ratio, Q4 board figure (dataset), hop 7, 2026
From	Lakehouse (vendor catalog, first region)
To	Exit test: copied files in a second region, an open catalog (Iceberg REST), an open SQL engine
Mechanism and schedule	Copy data and metadata files; register tables; run the board query at tag q4-2025-close; compare with the approved figure
Format change; what was lost or assumed	RIGHTS MEANING PRECISION Same Parquet and Iceberg files; SQL ported to the second engine's dialect. **Row access and masking policies and grants stay in the vendor catalog and do not travel; one row policy had been carrying a business rule (the intercompany exclusion), so its meaning was lost with it; the rounding rule lived in the old engine's dialect, not in the metric; egress is paid on the copy**
Accountable owner	Head of data platform; CFO (the metric)
Who can change it	Exit test team (named engineers); the open catalog's administrators
Classification; retention or erasure duty	Financial and personal data; the copy is a new location with the same duties
Read by (people, apps, models, agents)	The exit test team, internal audit
Cost per month; basis	Per run: copy, egress and test engineering, recorded for each run and shown here spread over the quarter (illustrative)

> *The files were the easy part. The number depended on a policy nobody had filed under the metric.*

The test cost a copy, engineering time and some engine time, recorded in the hop's cost column per run. What it bought was knowledge no contract review could give: that a security feature was doing business logic, that the rounding rule lived in the old engine's dialect rather than in the metric, and that the restated figure reproduced on a second engine. The same test serves audit, recovery after a regional failure and the arrival of a second engine. That is the answer to the fair objection that companies leave a platform once a decade: the exit test is not a plan to leave. It is a practical proof that you know what your number depends on.

OUR VIEW, AS OF NOVEMBER 2026

Lock-in chosen on purpose is a valid decision when it is written down with its reason and a review date, and the exit test shows what the choice costs. Whether to own or rent the platform itself is a separate decision, and the Medallion series covers it.

ERA CARD

Reader. A second engine, and internal audit reading its result beside the approved figure.

Security. Policies, masking and grants do not leave with the files; until rebuilt, a copy is less protected than its source, or protected by rules nobody wrote down.

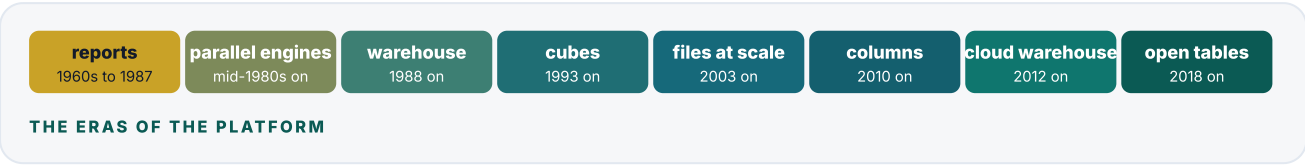
Cost. Egress for a switch is waived or capped under conditions; the rebuild of what does not travel is the larger and less visible bill.

AI. An assistant pointed at the copied tables answers with whatever rules travelled; it would have quoted 62.5 with confidence.

Monday rule: pick one board number, copy its tagged snapshot to a second engine this quarter, and write down every difference before you explain any of them.

CHAPTER SIXTEEN

Choosing per Workload



THE ERAS OF THE PLATFORM

After its exit test returned 62.5 percent in June 2026 for a quarter the board had approved at 61.8 percent (approved 5 February 2026, data to 23 January), the insurer's CFO asked the head of data platform a short question: which platform should hold the board number? It is the question most platform reviews start with, and it is the wrong one. A platform does not run "the data". It runs workloads, each with its own reader, its own service objective and its own way of becoming hard to leave. The board figure, a claims operations dashboard refreshed every five minutes and a seven-year archive of claim files have almost nothing in common except the storage underneath. The decision this chapter changes: **choose a format, an engine and a platform tier per workload, and write down what each choice makes hard to leave, with a reason and a review date**.

Start from the workload, not the vendor

Chapter One named nine workload classes: board and regulatory reporting, interactive BI, high-concurrency dashboards, ad hoc analysis, large transformations, streaming ingest, data science, sharing, and retention. Each has a service objective that a team can write in one line: how fresh the data must be, how fast an answer must come back, how many readers at once, how exactly a figure must reproduce, and how long it must be kept. Those lines, not a feature list, are the input to a platform decision.

Three facts from the earlier chapters shape every row of the table that follows. First, open file and table formats (Chapters Six and Eight) let more than one engine read the same files, so the storage choice and the engine choice can be made apart. Second, meaning, control and code do not travel with the files (Chapters Ten and Fifteen): a metric definition in a semantic layer, a row access policy in a catalog and a stored procedure in a dialect stay where they were written. Third, the clouds are a substrate and the platforms sit on them

(Chapter Twelve): **tier 1** is the cloud itself (object storage, identity, keys, network, egress), **tier 2** is a data platform running on it, and the self-assembled open stack (object storage, an open table format, an open catalog, an open engine) is the control case.

So the question for each workload is not "which vendor" but "which layer must this workload depend on, and what does that dependency cost to undo". A dashboard that depends on a platform's cache and BI model is fine, as long as someone has decided that on purpose.

The decision table

Table 16.1 maps the nine classes to the usual choice of format, engine and tier, and to the thing each choice makes hard to leave. It is a starting point for your own table, not a ranking: no row names a vendor, because the same row can be met on several platforms, and the platform cards (defined in Chapter Nine, seven of them filled in Chapter Twelve) say which features each platform offers as of November 2026.

Table 16.1 · Workload classes to formats, engines and tiers

Workload	Format	Engine	Tier	Hard to leave
Board and regulatory reporting	Open table format; quarter-close snapshot tagged with a retention period	A governed SQL engine; the metric implemented in SQL and published by the semantic layer	Tier 2 or open stack	Metric definitions in the semantic layer; row policies doing business logic; approval workflow
Interactive BI	Open tables plus a BI model	Engine with caching close to the BI tool	Tier 2	The BI semantic model, dashboards, cache settings, dialect
High-concurrency dashboards	Pre-computed aggregates beside the open tables	Engine with result caching and scaling for many readers	Tier 2	Concurrency features and capacity commitments; the refresh schedule
Ad hoc analysis	Open tables	Any SQL engine that reads the catalog, including open engines	Tier 2 or open stack	Little, if analysts keep to standard SQL; user-defined functions and saved notebooks
Large transformations	Open tables; Parquet data files	Distributed batch engine or the platform's SQL	Tier 2 or open stack	Pipeline code, stored procedures, the scheduler and its run history
Streaming ingest	Open table with frequent commits and scheduled compaction	Streaming engine writing snapshots	Tier 1 services or tier 2	Connectors, delivery guarantees, the maintenance jobs that keep small files in check

Workload	Format	Engine	Tier	Hard to leave
Data science	Parquet and open tables	Notebooks and dataframe engines	Tier 2	Notebooks, feature definitions, experiment and model registries
Sharing	An open sharing protocol or the platform's native sharing	The provider's sharing service	Tier 2	Recipients' grants, which live in the provider's catalog
Retention	Parquet and table metadata on object storage; tags for what must be kept	None by default; any engine on demand	Tier 1	Storage classes, key custody, legal holds and their audit trail

Read the last column first. No row lists the data files. Every entry is a definition, a grant, a piece of code, a schedule, a key or a commitment: the meaning, control and code that this book has followed since the prologue. The open table format moved the data out of that column, which is a real gain over the 2008 appliance (Chapter Four). It did not empty the column. Format interoperability, such as Delta UniForm (announced with Delta Lake 3.0 in June 2023) writing Iceberg metadata beside Delta, or Apache XTable (incubating since 11 February 2024), makes more engines able to read the files; it does not carry a row policy or a metric definition from one catalog to another.

The same table gives different answers in the other two companies of this book. The health system's largest workload is retention: imaging studies and FHIR resources kept for years, some with protected health information burned into the pixels. Its hard-to-leave column is key custody and the scope of each business associate agreement, not a metric, so its first decision is which keys it holds itself (Chapter Eleven) and which copies sit under which agreement. The hotel and retail group's hardest row is sharing: since the acquisition of a retail chain on a second cloud, point-of-sale data and guest profiles meet across two platforms, and every grant to a recipient lives in one provider's catalog. Its first decision is which side holds the grants, and where the EU guests' transfer basis is recorded (Chapter Fourteen).

On each platform. The tier column hides real differences: whether the platform writes the open format natively or reads it through a conversion, whether its catalog exposes an Iceberg REST interface, and where it enforces policy. Those are rows on the platform card (Chapter Nine; blank template in Appendix A), and they decide how wide the last column of Table 16.1 is on your platform.

Deliberate lock-in, written down

Some lock-in is the price of a good product. A platform's BI cache, its managed streaming or its sharing service can be worth more than the cost of leaving it. The mistake is not depending on a vendor. The mistake is depending on one without knowing it, and finding out on the day of an exit test, as the insurer did when a row access policy turned out to be doing the work of a business rule.

The fix is a Ledger row. The Ledger (the Reader's Brief and Appendix A) already has a column for what a hop lost or assumed; this book adds three optional columns: the **exit path** (travels: moves unchanged; travels with work: moves after rewriting; must be rebuilt: recreated from records), **own or rent**, and **as of**. A deliberate lock-in is a row whose exit path says "must be rebuilt" and whose loss column carries a sentence that starts "accepted because". It also carries a review date. At the insurer, the interactive BI dashboards were left on the platform's BI model, with the reason written ("faster to build; the board figure does not depend on it") and a review date set for the quarter before the platform contract renews. The board figure itself was moved the other way: its definition into the metric's SQL and its rows behind a tag, so that the one number that must reproduce depends on as little as possible.

Whether to own the layer or rent it at all, and how to weigh that against staff and contracts, is the subject of the Medallion series; this book records the choice and its date, and leaves the economics there.

> *Lock-in you chose, dated and reviewed, is a decision. Lock-in you discover in an exit test is a finding.*

Run your own benchmark

This book prints no rankings, for three reasons. Performance depends on the workload, the data and the configuration more than on the product. Prices and features change faster than a book. And the license and service terms of some database and analytics products restrict

publishing benchmark results without the vendor's consent, so a published comparison may be incomplete for reasons a reader cannot see. Read the terms of every product you test before you write anything down that will leave the building.

A test of your own workload is more useful than any published result anyway. Public suites help to understand engine behavior: TPC-DS, the Transaction Processing Performance Council's decision support benchmark, is now at version 4, and its site notes that V4 results are not comparable with earlier versions. Your board reporting is not TPC-DS. Five steps make an internal benchmark worth trusting:

- **Pick the real queries.** The ten numbers the board reads (Chapter Seventeen), the twenty most-run dashboard queries and the three heaviest nightly jobs, taken from query history, not invented.
- **Freeze the data.** Run every engine against the same tagged snapshot of the same files. With an open table format this is possible without a copy per engine; without one, the copy is part of what you are testing.
- **Define correct before fast.** Each query must return the approved figure at the tag, to the printed decimal. Once the rows were fixed, the insurer's second engine returned 61.9 where the approved figure was 61.8, because it rounded half up (Chapter Fifteen). A fast wrong answer fails the benchmark.
- **Measure at your concurrency, and read the meter.** Cold and warm runs, at the number of readers you expect at 9 a.m. on the day the board pack goes out, with the billing meter that moved for each run: bytes scanned, credits, capacity units or slot time (Chapter Thirteen).
- **Date the result.** Record the engine version, the configuration, the snapshot tag and the date. A benchmark is an "as of" number like any other; repeat it when the engine or the workload changes.

The result goes into the platform card as evidence for the rows it tests, and into the decision table as the reason a row says what it says.

> **OUR VIEW, AS OF NOVEMBER 2026**
>
> For the few numbers a board or a regulator reads, depend on the open layers wherever you can: open tables, tagged snapshots, definitions in SQL and in a semantic layer whose export you have tested. For everything else, use the platform's best features freely and record the dependency. Spending effort on portability where it pays, and nowhere else, is cheaper than either a fully open stack or a fully native one.

ERA CARD

Reader. The CTO and the data architect, choosing per workload instead of per vendor, with the board as the reader who must not notice the choice.

Security. Each tier choice moves where policies are enforced; a policy that lives in one catalog protects only the engines that ask that catalog.

Cost. Each workload is billed on its own meter, and capacity commitments are a lock-in line of their own.

AI. An assistant over the semantic layer reads whatever definition that layer holds; the workloads whose numbers it quotes need passports first.

Monday rule: write Table 16.1 for your own company, one row per workload, and for each "must be rebuilt" in the last column add a Ledger row with the reason and a review date.

CHAPTER SEVENTEEN

A Platform Ledger and Exit Test in 90 Days



The Apache Iceberg configuration reference lists the default for `history.expire.max-snapshot-age-ms` as 432000000 milliseconds: five days. When a team runs snapshot expiry with that default, snapshots older than five days that no branch or tag still needs can be removed, keeping at least one. The insurer's platform team had raised it to 45 days in 2024 for audit reads, which buys time, not protection. Five days is a sensible default for a table that changes every hour and an unsafe one for a number a board approved and a regulator may ask about years later. At the insurer, the quarter-close snapshot behind the Q4 2025 figure, 61.8 percent (approved 5 February 2026, data to 23 January), was not protected by anything when the board met in February: no tag, no recorded snapshot id, no "as of" beside the number. This chapter is the plan the insurer ran after that meeting: thirteen weeks, from Monday 23 February to Friday 22 May 2026, cut to the minimum that works. It runs alongside the 90-day Ledger plan of *From Tally to Token* (its chapter "Filling the Ledger in 90 Days") and applies the same method, in the same three blocks named days 1 to 30, 31 to 60 and 61 to 90, to the numbers a board reads. The decision it changes: **within the thirteen weeks, every number the board reads carries a metric passport, a Ledger row and a protected close snapshot for each table it reads, and one of them is ready for an exit test that then runs every quarter**.

FIGURE 17.1

Thirteen weeks, three blocks (23 February to 22 May 2026): every board number gets a passport, a Ledger row and a protected close snapshot

Days	Work	Deliverable	Owner
1 to 30	List the numbers in the last board pack (ten at the insurer); find or write each metric contract; tag the last quarter-close snapshot	A metric passport and a Ledger row for each number, gaps visible	Each number's owner (CFO for the loss ratio); data platform team
31 to 60	Make the catalog and semantic layer the system of record; export policies and grants as code; tag every quarter-close snapshot with a retention period	One authoritative definition per number; policies under change control; tags with snapshot ids in the passports	Head of data platform; catalog administrators
61 to 90	Publish the restatement in the Q1 2026 pack; set up the exit test on one board number at its close tag; price the exit inventory	Q4 2025 restated at 62.2 (data to 23 Jan 2026) in the pack of 12 May 2026; exit test ready; exit inventory priced in person-days and charges	CFO (owner) and appointed actuary (approver); exit test team
After the plan	Run the exit test every quarter, first in June 2026; reconcile every difference	June run: 62.5, then 61.8 after two fixes, 62.2 with status 7 included, confirming the restatement; each result in the Ledger	Exit test team; head of data platform

Each block ends with a deliverable the board's owners sign; nothing in the first block changes a definition, and nothing in the third is a plan to leave.

Author's plan, applied to the insurer (illustrative); exit test results from Chapter Fifteen.

Days 1 to 30: the ten numbers the board reads

Start with the board pack, not the platform. Open the last pack and list every number in it that a director could quote in a minute of the meeting. At the insurer the list had ten entries: the property loss ratio, the casualty loss ratio, the loss adjustment expense ratio, the expense ratio, the combined ratio, net premiums written, net premiums earned, prior-year reserve development, policyholders' surplus and the risk-based capital ratio. Your list will differ; ten is a good size because it is small enough to finish and large enough to cover the numbers that move decisions.

For each number, three artifacts, in this order:

- **The metric contract** (Chapter One), written or found. For the loss ratio it existed: the board's audit committee approved it in 2019. For three of the ten it did not, and writing it took longer than anything else in the month, because it meant asking the owner questions nobody had asked: is business assumed from the affiliate in or out, which rounding rule, which basis.
- **The metric passport**: eleven rows, from metric id and version to restatement status (the template is in Appendix A). The passport is where the five times live: the effective date and the knowledge cutoff have rows of their own, the calculation is recorded with the transformation version, the approval with the owner and approver, and the publication with the restatement status. A passport with an empty row is still useful; the empty row is the finding.
- **A Ledger row** for the hop where the number is computed today: from which tables, by which job, read by whom, owned by whom, with the loss codes the team can already see. Use the optional **as of** column for the five times and the **exit path** column for what would travel if the computation moved.

Do not try to fix definitions in the first month. At the insurer the population row of the loss-ratio passport was easy to fill and hard to accept: the bridge (Chapter Eight) had shown that the 2021 filter drops status 7 policies the contract includes. The team wrote the gap into the passport and left the decision to the CFO, the appointed actuary and the audit committee. One step cannot wait for day 31, though: tag the last quarter-close snapshot before the next expiry run can remove it. The insurer tagged its Q4 2025 close in the first week: on Tuesday 24 February 2026 the platform team found the snapshot committed at the 23 January cutoff still in the table, only because the table's maximum snapshot age was 45 days, and created the tag `q4-2025-close` on it, with retention set to the statutory records period. The weekly expiry run, every Sunday, would have removed that snapshot on Sunday 15 March: 45 days after 23 January is Monday 9 March, and the next run fell six days later. A list of ten passports with honest gaps, signed by each number's owner, is the deliverable of day 30.

METRIC PASSPORT

Metric id and version	Property loss ratio, board figure; contract of 2019
Population and exclusions	Gap: the warehouse view drops status 7 policies (a 2021 filter); the contract includes them
Snapshot id or tag	Gap: none recorded; the quarter-close snapshot is unprotected until tagged
Restatement status	Open: owner and approver to decide

The box shows four of the eleven rows as the insurer's team first filled them. The full passport, as it stood at the end of the plan, is in Appendix A.

Days 31 to 60: one system of record for meaning and control

The second month moves meaning and control out of the places where they hid. Three pieces of work, each with a named owner.

Catalog and semantic layer as the system of record. For each of the ten numbers, the metric contract is the authority; one version-controlled SQL definition implements it; the semantic layer publishes that definition, registered in the catalog with its owner, its tables and the code tables its filters depend on; and a release check fails if the SQL and the semantic layer's metric disagree. Dashboards and notebooks read the metric, not their own copies of the SQL. This is the step that removes the two-numbers problem at its root: a notebook that computes its own loss ratio from the latest snapshot is no longer the same thing as the board figure, and the catalog says so. Chapter Ten describes the layers; the work here is to choose, for each metric, which one is authoritative and to delete or label the rest.

Policies as code. Row access policies, masking policies and grants are exported from the catalog into text files kept under change control, reviewed like any other change, and applied by a deployment job. The policies become readable by someone who is not a catalog administrator, and they can be translated and applied to a second catalog, whose policy language and identity model may differ. Readable is not the same as understood. The insurer's export listed dozens of policies, among them a row access policy on the claims and premium tables that hid rows from every role except reinsurance accounting; nobody read it as a business rule until the exit test in June showed what it was doing. The loss-ratio passport's population row already said "business assumed from the affiliate: excluded", so the question in month one had an answer, and nobody asked where that exclusion was

enforced. It was enforced by the policy, not by the metric. A useful review question for every exported policy: if this policy disappeared, would any approved number change? If the answer is yes, the policy is doing business logic, and the rule belongs in the metric.

Tags with retention on quarter-close snapshots. In an Iceberg table, a tag is, in the specification's words, a label "for individual snapshots", recorded in the table metadata with the snapshot id it points to. A tag can carry its own `max-ref-age-ms`; when snapshots expire, the snapshot a valid tag references is kept, whatever the table's default age. The Iceberg Java configuration lists the default for references other than the main branch as forever, but the catalog or the platform running your maintenance may set its own, so set the value on every tag you create. The insurer now tags every quarter-close snapshot of the claims and premium tables (`q4-2025-close` for the quarter in this book), sets the tag's age to the statutory records retention period, and writes the tag and the snapshot id into the passport at the moment of approval. Check that your storage lifecycle rules cannot delete the data files under a tagged snapshot, and that the orphan-file job runs against the table's current metadata, or the metadata will outlive the data.

A team that finds its last close snapshot already expired should record that as a COMPLETENESS loss on the Ledger and rebuild the close from the warehouse's own close table, if one exists, before tagging anything.

Days 61 to 90: the restatement, the exit test and the priced inventory

The third month proves the first two. It carried the restatement: the Q1 2026 board pack, published on Tuesday 12 May 2026, restated Q4 2025 to 62.2 on the strength of the February bridge (the thread closes below). It also prepared the exit test. Pick one board number, the one with the longest history and the most readers, and set the test up: copy its tables' data and metadata files to object storage outside the platform's control, register them in an open catalog, port the metric's SQL to a second engine, and agree what will be compared with the approved figure at the quarter-close tag.

At the insurer the first run came in June 2026, just after the plan, as the first of the quarterly runs; Chapter Fifteen tells it in full. It returned 62.5 because the second engine saw the rows assumed from the affiliate under the intercompany pooling agreement, which the vendor catalog's row access policy had been hiding from every role except reinsurance accounting: a security feature doing business logic, and a policy that did not travel. Writing the intercompany exclusion into the metric's SQL brought the exact value back to 61.850, printed 61.9, because the engine rounds half up where the contract says half to even; that difference showed only after the rows were fixed. Pinning the rounding rule gave 61.8, matching the approved figure, and with the 2021 population filter also corrected the same engine returned 62.2, confirming the restatement.

The test has two outputs. The first is a reconciliation: every difference between the second engine's answer and the approved figure, explained in exact values, not printed ones, as the bridge in Chapter Eight explained the 2.250 points between 61.850 and 64.100 (2.3 on the screen, the difference of two rounded figures). The second is the **exit inventory** (template in Appendix A), one row for each thing the board figure depends on: files, table metadata, catalog and grants, policies, keys, code, the semantic layer and dashboards, pipelines, identity, runbooks and people, contracts and egress. For each row the team writes where it lives, whether it travels, travels with work or must be rebuilt, the effort in person-days, the charges (egress, licences, a period of running both), the owner and the evidence.

At the insurer, the data files travelled, copied byte for byte. The table metadata, the metric's SQL and two vendor-specific SQL functions travelled with work: the metadata's file paths were rewritten and the SQL was ported. Row access policies, masking policies, grants, the semantic layer's metric definitions and the scheduler had to be rebuilt. The inventory priced each of those, and the price was what the board wanted to see: not a plan to leave, but the cost of knowing what its number depends on.

Two pricing notes belong in every inventory. The egress on the test copy is paid: the providers' 2024 waivers of data transfer fees for customers leaving a cloud come with conditions, such as Google Cloud's requirement to terminate the agreement for the service being left and the AWS requirement, in its October 2026 terms, to move all data off AWS or all data off one service, so an exit test that stays is not covered. And for customers in the European Union, the Data Act ends switching charges from 12 January 2027 (Article 29) but allows a provider to charge egress at cost for data used in parallel with another provider (Article 34); a quarterly exit test that keeps the source running looks more like parallel use than a switch, so ask counsel how the contract and the provider's addendum treat it. Prices go in the dated web companion, not in the inventory's template.

The thread closed

The plan ends where the prologue began, with one number and its times.

The restatement. The written contract asks for status 7 policies to be included. On that contract, at the board's knowledge cutoff of 23 January 2026, Q4 2025 is **62.2 percent**: losses of 155,395 over net premiums earned of 250,000 (thousands of dollars, illustrative) is 62.158, shown as 62.2. The status 7 losses (770) and the anticipated salvage and subrogation (240) did not develop between 23 January and 13 February (illustrative), so the amounts in the bridge serve the restatement unchanged. It is not 62.1. The approved figure is exactly 61.850, shown as 61.8 because the contract rounds half to even, and the status 7 effect is +0.308 unrounded, so 61.850 + 0.308 = 62.158. Adding the rounded parts, 61.8 + 0.3, would have printed 62.1, a third wrong number. Add unrounded parts, round once. The

insurer restated Q4 2025 at 62.2 in its Q1 2026 board pack, published on Tuesday 12 May 2026, with a disclosed note giving the old figure, the new one, the reason and the quarters back to 2021 in which the same filter had moved the figure by one to three tenths of a point (illustrative).

The passport. The board figure for Q4 2025 now carries a metric passport with the tag `q4-2025-close`, the snapshot id that tag references, and the five times: effective 31 December 2025; knowledge cutoff 23 January 2026; calculated 26 January 2026; approved 5 February 2026 by the appointed actuary, countersigned by the CFO as owner; published in the board pack of 10 February 2026. Its restatement row records the move from 61.8 to 62.2, the pack that carried it and that pack's publication on 12 May 2026; the June exit test confirmed 62.2 on a second engine. Appendix A prints it in full.

The note. The analyst's note that put two numbers on one screen is closed with both numbers and their times: 61.8 percent, computed at the 23 January cutoff by the close job on the platform's SQL warehouse engine, through a view whose 2021 filter drops status 7 policies the contract includes, approved 5 February; 64.1 percent, read on 17 February from the latest snapshot of 13 February, a valid current estimate on a different definition (salvage and subrogation not netted) and a later knowledge time. Neither was the contract figure; the note says so, and points to the restatement.

The test. The exit test now runs every quarter, on the new close tag, as a scheduled piece of work with an owner and a result recorded in the Ledger. It is a practical proof that the insurer knows what its board number depends on, and the same run serves internal audit, the recovery plan and any second engine the team may want later.

> **WHAT HASN'T CHANGED**
>
> Actuaries read a loss development triangle at an evaluation date, and an estimate without one cannot be compared with the next (Chapter One). Every platform hop in this book, from the 1987 printout to the 2024 lakehouse, made that date easier to drop. The thirteen weeks put it back beside the number, where the actuaries had kept it all along.

The Ledger rows for the insurer's board figure, hop 0 to hop 7, are printed together in Appendix A. The last row is the exit test of June 2026, with its RIGHTS, MEANING and PRECISION losses: the policies and grants that stayed in the vendor's catalog, the business rule one of those policies carried, and the rounding rule that lived in the old engine's dialect rather than in the metric.

ERA CARD

Reader. The board, reading a number that now carries its times; internal audit, reading the passport behind it.

Security. Policies kept as code can be read, reviewed, ported and retested; a policy that encodes a business rule is moved into the metric.

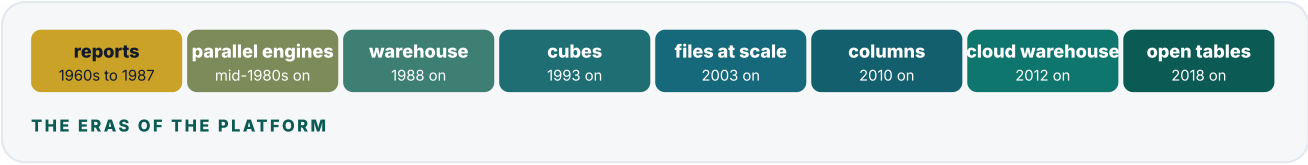
Cost. Thirteen weeks of a small team, one test copy's egress each quarter, and an exit inventory priced in person-days.

AI. An assistant that quotes the board figure can quote its passport too: tag, snapshot id and the five times.

Monday rule: open your last board pack, list the numbers a director could quote, and for the first of them fill the eleven rows of a metric passport before Friday, leaving the gaps visible.

CLOSING

Epilogue: The Format Outlives the Platform



THE ERAS OF THE PLATFORM

On 30 July 2013 Twitter announced Parquet 1.0, columnar storage for Hadoop built with Cloudera. Eight years later the insurer unloaded its appliance tables to Parquet files to load its cloud warehouse. Three years after that its lakehouse wrote the same claims and premium data as Parquet files under Iceberg metadata. In June 2026 the exit test copied those data files byte for byte to a second region, and an open engine read them once the table metadata's file paths had been rewritten for the new location. The format, and the records it carried, had passed through three platforms and arrived intact. The board figure did not arrive with it. That gap, between data that travels and meaning that does not, is the lesson this book ends on.

What travelled, and what did not

The exit test of Chapter Fifteen kept a precise list. The data files travelled, byte for byte. The table metadata travelled with work: its file paths were rewritten, and the snapshots, schemas and quarter-close tag it records came through unchanged. The metric's SQL and two vendor-specific SQL functions travelled with work too, ported to a new dialect, and the rounding rule was lost in that port. What had to be rebuilt was everything that told a reader what the numbers meant and who could see them: the row access and masking policies, the grants, the semantic layer's metric definitions and the scheduler that ran the close job. One of those row policies had been quietly doing business logic, hiding intercompany rows from the board view, and nobody had written it down as part of the metric. So the first run on the second platform returned a number nobody had seen before.

Read the thread again with that list in mind. In 1987 the context lived in pencil, in a binder. In 1994 the reserve rule lived in a spreadsheet beside the warehouse. In 1999 the "as of" was the cube's build time, printed nowhere. In 2021 the meaning of status 7 lived in a table nobody moved. In 2024 the "as of" lived in a snapshot nobody recorded, and the population rule lived partly in a security policy. Each time, the data moved and the thing that gave it meaning stayed where it was, in a place the next platform did not read.

Open formats did change something real. A 2008 appliance held its data in its own storage; leaving meant unloading, converting and reloading every table. A 2026 lakehouse holds its data in files any engine that speaks the format can open. That is a large gain, and the book does not discount it. It moves the hard part of leaving from the data to the meaning and the controls, which are smaller, scattered and easy to forget until a third number appears.

One more reader now stands at the end of the path. An assistant over the semantic layer, asked for the fourth-quarter loss ratio, quotes whichever number it finds, with the confidence of whatever it read; what such systems learn from is covered in depth later in this series.

Names will move; the five times will not

Much of this book will date. Platform names change, features ship and are renamed, projects graduate or stall, laws take effect on schedules that slip. That is why every platform card in Chapter Twelve is dated November 2026, why the prices sit in a dated web companion and not in print, and why each fact carries the date it was checked in Appendix E. Read the cards as a snapshot with a tag, not as the current state.

Some things in the book do not depend on any vendor. A number has an effective date, a knowledge cutoff, a calculation, an approval and a publication. A dataset moves in hops, and each hop can lose meaning, time, rows, an owner or a right. A result is trustworthy when it can be rerun from recorded data and return the published figure. None of this is new. In 1934 an actuary tested a company's Schedule P reserves "as of December 31, 1930" against what was later paid, and today's NAIC instructions still speak of "the earliest evaluation date" in the loss triangles. The idea has outlived every system that has carried insurance data since. The Ledger, the metric passport and the five times are that idea, written down so a platform can keep it.

OUR VIEW, AS OF NOVEMBER 2026

What changes. Table formats and catalogs are converging on shared interfaces. Apache Polaris, which graduated to an Apache top-level project in February 2026, implements Iceberg's REST catalog interface, and Unity Catalog, open sourced in June 2024, states that it is compatible with the same interface. Data and table metadata will keep getting easier to move between engines. I expect the point of lock-in to keep moving up the stack: into the semantic layer that holds metric definitions, into the policy engine that decides who sees which rows, and into the operating knowledge of the people who run them.

What does not change. Someone has to write down what a number means, as of when, and who may see it, and keep that record beside the number. No format does this for you. A platform can store a definition and a snapshot id; it cannot decide that the board figure needs one.

What I expect. Teams that run a small exit test on one board metric each quarter will find their hidden business logic in policies, functions and filters long before an auditor or a migration does. Teams that do not will find it on the day two numbers meet on one screen. Deliberate lock-in remains a valid choice when it is recorded with its reason and a review date; whether to own or rent each layer is the Medallion series' subject.

Back to the green-bar page

The 1987 printout is still in the actuaries' binder, pencil and all. It was a poor platform. It could not be queried, joined or shared, and its context lived in someone's handwriting. But it named the month it described, and it kept the machine's number and the approved number on one sheet with the difference written between them. When the board asked at its meeting of 19 February 2026 which number was the fourth quarter, the oldest record in the chain was the one that could have answered in its own words.

That is not an argument for paper. It is an argument for making the platform do what the page did: keep each number's definition and times with it, record which data it came from, and test now and then that it can be reproduced somewhere else. Formats will outlive the platforms that write them. The meaning will outlive them only if someone carries it.

> *Files travel. Meaning travels only when someone writes it down and moves it with them.*

Monday rule: choose one board number, record its snapshot and definition beside it this quarter, and schedule the first run of an exit test that reproduces it on a second engine.

APPENDICES

APPENDIX A

Templates: Ledger, Metric Passport, Platform Card, Exit Inventory

The four tools of the book and the bridge table from which every thread number can be rebuilt. The blank templates are a free download on the book's page, blog.jcardena.com/en/ebooks/from-warehouse-to-lakehouse/. The insurer and its numbers are illustrative.

The Data Lineage Ledger

The Data Lineage Ledger: one row per hop

Record or dataset, hop	From	To	Mechanism and schedule	Format change; what was lost or assumed	Accountable owner	Who can change it	Classification; retention or erasure duty	Read by (people, apps, models, agents)	Cost per month; basis

Loss codes

MEANING	a code or definition did not travel with the value
CONTEXT	notes or surrounding knowledge stayed behind
PRECISION	type or rounding changed the value
TIME	century, time zone, or event versus processing time changed
IDENTITY	the key was renumbered, merged or split
OWNERSHIP	a copy has no accountable owner
RIGHTS	purpose, consent or legal basis did not travel
COMPLETENESS	rows or fields were dropped

One row per hop; loss codes first, then one sentence. Optional columns: **exit path**, **own or rent**, and **as of** (the five times). For personal data, location, access path and transfer basis (with the date checked) are written into the classification column beside the legal basis, not

added as columns (Chapter Fourteen). A deliberate lock-in (Chapter Sixteen) carries a reason and a review date. The worked example follows the board figure from hop 0 (1987) to hop 7 (June 2026).

THE LEDGER, PART 1 OF 2

One Number, Since 1987: the full Ledger: movement and meaning

Hop, year	From	To	Mechanism and schedule	Format change; what was lost or assumed
0 1987	IMS claims database and the IMS policy master	Monthly loss report, printed on green-bar paper by a COBOL report program	Monthly batch job after month end; the report is read in the actuaries' room	**CONTEXT** Fixed-width print lines, 132 columns; totals by line of business. **Reserve adjustments written by hand on the printout and kept in the actuaries' binder; the printed figure and the used figure differ**
1 1994	IMS extracts (claims, policy)	Enterprise data warehouse on DB2 (a separate analytic copy)	Monthly extract programs, transfer and load jobs; subject areas for claims and premium	**MEANING** Fixed-width extracts loaded to normalized DB2 tables with history. **Incurred loss is defined once in the warehouse, but the reserve adjustment rule stays in the actuaries' spreadsheet, so the warehouse figure still needs a manual step**
2 1999	Enterprise data warehouse	Claims star schema and a multidimensional cube for the board pack	Cube rebuilt after each quarter close; the board pack reads the cube	**TIME** Fact table of claim transactions, dimensions for line, state, accident period; pre-aggregated cube cells. **The cube's build time is the only 'as of', and it is printed nowhere; later corrections reach the warehouse but not the frozen cube**
3 2008	DB2 enterprise data warehouse	MPP warehouse appliance (shared-nothing, one rack)	One-time migration: unload, convert, reload; load jobs and stored procedures rewritten for the appliance	**COMPLETENESS** DB2 tables to appliance tables distributed on claim number. **History before 1995 not migrated; kept on tape in the old load format, with a record layout no current job can read**
4 2014	Appliance extracts and raw claim files (adjuster notes, photos, third-party reports)	Hadoop data lake beside the appliance (HDFS, Hive tables stored as ORC)	Daily file drops into a raw zone; Hive tables defined over the files; nightly jobs	**OWNERSHIP** Text, images and PDFs as files; tabular extracts as ORC; schema applied when read. **Adjuster notes and photos land in a raw zone with no accountable owner and no retention rule; schema on read means each team reads them differently**
5 2021	MPP appliance and the Hadoop lake's curated tables	Cloud data warehouse (loaded via Parquet), in the same programme that moved the policy domain from DB2 for z/OS	Bulk migration plus nightly loads; loss-ratio SQL rewritten in the new dialect; the appliance retired; the lake keeps running for claim files	**MEANING** Appliance tables unloaded to Parquet and loaded to warehouse tables; DECIMAL types kept; status as a bare integer. **The STATUS_CODES lookup table is not carried (as in From Tally to Token); a rewritten filter reads 'premium due = 0' as 'not a live policy' and drops status 7 policies from the loss-ratio view**

Hop, year	From	To	Mechanism and schedule	Format change; what was lost or assumed
6 2024	Cloud warehouse tables and the lake's claim files	Lakehouse: Iceberg tables on object storage, a vendor-managed catalog with an Iceberg REST interface, two engines	Tables rewritten as Iceberg under a vendor-managed catalog; the platform's SQL warehouse engine runs the close job and writes the dashboard's close table; the notebook engine reads the latest snapshot; the reserving batch commits the quarterly loss table every Friday	TIME Parquet data files, Iceberg metadata, manifests and snapshots; row and column policies held in the vendor catalog. **Numbers are computed at read time from the latest snapshot; no snapshot id, tag or 'as of' is kept with the board figure, and quarter-close snapshots are removed by the weekly expiry run once older than the platform's 45-day setting**
7 2026	Lakehouse (vendor catalog, first region)	Exit test: copied files in a second region, an open catalog (Iceberg REST), an open SQL engine	Copy data and metadata files; register tables; run the board query at tag q4-2025-close; compare with the approved figure	RIGHTS MEANING PRECISION Same Parquet and Iceberg files; SQL ported to the second engine's dialect. **Row access and masking policies and grants stay in the vendor catalog and do not travel; one row policy had been carrying a business rule (the intercompany exclusion), so its meaning was lost with it; the rounding rule lived in the old engine's dialect, not in the metric; egress is paid on the copy**

THE LEDGER, PART 2 OF 2

One Number, Since 1987: the full Ledger: control, readers and cost

Hop	Accountable owner	Who can change it	Classification; retention or erasure	Read by	Cost per month; basis
0	Controller	Claims systems programmers (the report program); actuaries (the hand adjustments)	Financial record; kept with the statutory working papers	Actuaries, the controller, the board (a typed summary)	Machine time on the shared mainframe plus about two actuary days a month for the hand adjustments (illustrative)
1	Director of management information	Warehouse load jobs; two warehouse developers	Financial and personal data (claimants); retention per statutory records rules	Finance analysts, actuaries, report writers	Share of mainframe DB2 capacity and two developers (illustrative)
2	Director of management information	BI team (cube design and rebuild job)	Aggregated financial data; no claimant names in the cube	CFO, board, actuaries (drill to line and state, not to claim)	OLAP server licence and a small BI team (illustrative)
3	Head of data warehousing	Appliance DBAs (vendor-trained); ETL team	Financial and personal data; retention per statutory records rules	Finance, actuaries, the cube, pricing analysts	Appliance purchase and annual support contract (illustrative)
4	None recorded for the raw zone; head of analytics for curated tables	Anyone with write access to the raw zone; the Hadoop platform team	Personal and sensitive data (claimant photos, notes) not yet classified	Data scientists, fraud analytics, the pricing team	Cluster hardware, support subscription and a platform team (illustrative)
5	Head of data platform	Pipeline service account; data engineers; the warehouse load role	Financial and personal data; US federal (GLBA) and state insurance privacy rules; retention duties	Finance and actuarial dashboards; the board pack; data scientists	Warehouse compute and storage for this domain, about 3,400 dollars a month (illustrative)
6	Head of data platform (platform); CFO (the metric)	Platform team; write jobs; catalog administrators (policies and grants)	Financial and personal data; GLBA and state privacy rules; policies enforced through the vendor catalog	Finance dashboards, actuaries' notebooks, pricing models, an assistant over the semantic layer	Storage, two engines and catalog for this domain, about 3,100 dollars a month (illustrative)
7	Head of data platform; CFO (the metric)	Exit test team (named engineers); the open catalog's administrators	Financial and personal data; the copy is a new location with the same duties	The exit test team, internal audit	Per run: copy, egress and test engineering, recorded for each run and shown here spread over the quarter (illustrative)

The metric passport

METRIC PASSPORT (BLANK)

Metric id and version	
Numerator	
Denominator	
Population and exclusions	
Basis	
Effective date	
Knowledge cutoff	
Snapshot id or tag	
Transformation version	
Owner and approver	
Restatement status	

METRIC PASSPORT: PROPERTY LOSS RATIO, Q4 2025 (AS OF JUNE 2026)

Metric id and version	Property loss ratio, board figure; contract of 2019; definition revised 2026
Numerator	Losses incurred, net of reinsurance and of anticipated salvage and subrogation, excluding LAE: 155,395 (thousands, illustrative)
Denominator	Net premiums earned: 250,000
Population and exclusions	Status 7 included; business assumed from the affiliate excluded in the metric's SQL
Basis	Management metric from statutory amounts; calendar quarter; one decimal, half to even
Effective date	31 December 2025 (quarter end)
Knowledge cutoff	Claims received by 23 January 2026
Snapshot id or tag	Tag `q4-2025-close` and the snapshot id it references; retention: statutory records period
Transformation version	Close job version and run of 26 January 2026, on the platform's SQL warehouse engine through the vendor catalog; restated with the metric definition of the Q1 2026 pack
Owner and approver	Owner: CFO; approved 5 February 2026 by the appointed actuary, countersigned by the CFO
Restatement status	Published 10 February 2026 at 61.8 (exact 61.850); restated 12 May 2026 to 62.2 (exact 62.158) in the Q1 2026 pack, disclosed note; confirmed by the June 2026 exit test

The platform card

Ten rows, this order, a date and a source per fact; the schema is set out in Chapter Nine and seven filled cards are in Chapter Twelve.

Platform card: _____ · as of _____

Scope and date	
Storage of record	
Table formats (native, read, write)	
Catalog and its interface	
Semantic layer	
Where policies are enforced	
Key custody	
Compute and SQL dialect	
Billing meters	
What travels (travels / travels with work / must be rebuilt)	

The exit inventory

One row per dependency of an important result, priced in your dated copy. The exit path takes one of three values: **travels** (moves unchanged, as the insurer's data files did, copied byte for byte); **travels with work** (moves after rewriting, as its table metadata did with rewritten file paths, with the metric's SQL and two vendor-specific functions); **must be rebuilt** (recreated from records: row and masking policies, grants, the scheduler, the semantic layer's definitions).

Item	Where it lives now	Exit path	Effort (person-days)	Charges	Owner	Evidence
Files						
Table metadata						
Catalog and grants						
Policies (row, column, masking)						
Keys						
Code (dialect, procedures, UDFs)						
Semantic layer and dashboards						
Pipelines and scheduler						
Identity						
Runbooks and people						
Contracts and egress						

The bridge table: rebuild the numbers

Ratio = losses / net premiums earned x 100, rounded once to one decimal (half to even on the contract). Thousands of dollars, illustrative.

Input	Amount
Net premiums earned	250,000
Board-basis losses, cutoff 23 Jan 2026, status 7 filtered out	154,625
Later development (13 Feb 2026 snapshot)	4,615
Status 7 losses	770
Anticipated salvage and subrogation	240
Affiliate business: premiums earned	10,000
Affiliate business: losses	7,823

Step	Exact	Printed	Effect exact	Effect printed
Board figure as approved: cutoff 23 Jan 2026, contract rounding, 2021 population filter applied	61.850	61.8		
+ later knowledge (latest snapshot, 13 Feb 2026)	63.696	63.7	+1.846	+1.9
+ status 7 losses kept (population)	64.004	64.0	+0.308	+0.3
+ anticipated salvage and subrogation not netted (definition)	64.100	64.1	+0.096	+0.1
Exit test first run (June 2026): intercompany rows the row policy had hidden are back	62.480	62.5		
Fix 1: exclusion written into the metric's SQL; the engine rounds half up	61.850	61.9		
Fix 2: rounding pinned to half to even (second run)	61.850	61.8		
Restated: written contract at the board cutoff	62.158	62.2		

Checks: 154,625 / 250,000 = 61.850; adding 4,615, 770 and 240 gives 160,250 / 250,000 = 64.100; (154,625 + 7,823) / (250,000 + 10,000) = 62.480; 155,395 / 250,000 = 62.158, printed 62.2. Printed effects are differences of rounded figures: add unrounded parts, round once. The 2.3 on the board's screen is 64.1 minus 61.8; the exact gap is 2.250. The status 7 losses (770) and the anticipated salvage and subrogation (240) did not develop between 23 January and 13 February (illustrative), so the same amounts serve the bridge and the restatement. 62.480 prints 62.5 under either rule, so half up shows only after fix 1.

APPENDIX B

Timeline

Twenty-nine dated points from the book, each checked against a primary source: a paper, a standard, a statute, a regulator's page or the announcement of the company or foundation concerned. Where only a month or a year is certain, only that is given. Figure B.1 draws the same arc with the insurer's systems beside it, to show that each kind of platform kept running after the next one arrived.

FIGURE B.1

The arc of analytics platforms, 1971 to 2026: new ideas joined the old ones instead of replacing them



Read the lanes as overlapping, not as a relay: no milestone ended the lane above it, and the insurer (bottom two lanes, illustrative) ran its warehouse line and its lake side by side from 2014, with the board figure computed in 2026 by the lakehouse platform's SQL warehouse engine.

Sources: Gorry and Scott Morton, MIT working paper 510-71 (1971); ACM SIGMOD 1985; IEEE Database Engineering 9(1), 1986; IBM Systems Journal 1988; Library of Congress (Inmon 1992, Kimball 1996); Codd, Codd and Salley 1993; SOSP 2003; OSDI 2004; VLDB 2005; PVLDB 2010; Dixon 2010; announcements by Google, AWS, Snowflake, Twitter, the Apache Software Foundation, Databricks and Microsoft; ASF incubator records. Insurer lanes from the thread (illustrative). Details in Appendix B and Appendix E.

Date	Event	Ch	Source
1971	Gorry and Scott Morton, "A Framework for Management Information Systems"	2	MIT Sloan School working paper 510-71
May 1985	Copeland and Khoshafian, "A Decomposition Storage Model": storing a table column by column	4	Proceedings of ACM SIGMOD 1985
Mar 1986	Stonebraker, "The Case for Shared Nothing"	4	*Database Engineering* 9(1), IEEE Computer Society
1988	Devlin and Murphy, "An Architecture for a Business and Information System"	2	*IBM Systems Journal* 27(1)
1992	Inmon, *Building the Data Warehouse*	2	Library of Congress, LCCN 91041284
1993	Codd, Codd and Salley, "Providing OLAP to User-Analysts: An IT Mandate"	3	The white paper
1996	Kimball, *The Data Warehouse Toolkit*	2	Library of Congress, LCCN 95050557
Oct 2003	Ghemawat, Gobioff and Leung, "The Google File System", SOSP	5	ACM, doi.org/10.1145/945445.945450
Dec 2004	Dean and Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", OSDI '04	5	USENIX
2005	Stonebraker et al., "C-Store: A Column-oriented DBMS", 31st VLDB Conference, Trondheim	4	VLDB proceedings
Sep 2010	Melnik et al., "Dremel: Interactive Analysis of Web-Scale Datasets"	6	*PVLDB*, doi.org/10.14778/1920841.1920886
14 Oct 2010	James Dixon uses and explains the term "data lake" in "Pentaho, Hadoop, and Data Lakes"	5	The author's blog
Dec 2011	SQL:2011 published, with temporal tables (application-time and system-time periods)	1	ISO/IEC 9075-2:2011 (published December 2011); Kulkarni and Michels, *SIGMOD Record* 41(3)
1 May 2012	Google BigQuery made publicly available	7	Google Developers Blog
14 Feb 2013	Amazon Redshift available to all customers (preview announced 28 Nov 2012)	7	AWS What's New
30 Jul 2013	Parquet 1.0 announced by Twitter, with Cloudera	6	Twitter Blog
23 Jun 2015	Snowflake generally available	7	Snowflake press release
17 Feb 2016	Apache Arrow announced as a top-level project	6	The Apache Software Foundation

Date	Event	Ch	Source
16 Nov 2018	Iceberg enters the Apache Incubator	8	Apache Incubator, Iceberg status page
24 Apr 2019	Delta Lake open sourced	8	Databricks press release
3 Dec 2019	Amazon Redshift RA3 nodes with managed storage	7	AWS What's New
16 Jul 2020	Schrems II: the Court of Justice invalidates the Privacy Shield decision (C-311/18)	14	EUR-Lex, CELEX 62018CJ0311
15 Nov 2023	Microsoft Fabric generally available	12	Microsoft Fabric Blog
Jan to Mar 2024	Google Cloud (11 Jan) and AWS (5 Mar) announce data transfer fee waivers for customers leaving	15	Google Cloud Blog; AWS News Blog
10 Jun 2024	Mandiant reports the UNC5537 campaign against Snowflake customer accounts	11	Google Cloud Threat Intelligence blog
13 Jun 2024	Unity Catalog open sourced, hosted at LF AI and Data (Databricks blog; press release dated 12 June)	10	Databricks blog
31 Jul 2025	UK CMA final decision in the cloud services market investigation	15	CMA case page
12 Sep 2025	EU Data Act applies, including Chapter VI on switching (Arts. 23 to 31)	15	Regulation (EU) 2023/2854, Art. 50
15 Feb 2026	Apache Polaris graduates to a top-level project	10	Apache Incubator, Polaris status page (ASF announcement 5 Mar 2026)

One date ahead of the cutoff matters to every exit plan: under Article 29 of the Data Act, switching charges end on 12 January 2027, while egress for data used in parallel with another provider may still be charged at cost (Article 34). Full references for every row are in Appendix E.

APPENDIX C

Failure Library

Real failures on analytics platforms, in migrations and in the copies and backups around them, each with the official report, regulator decision or the operator's own published review behind it. One line each, with only facts the source states; the last column names the chapter whose decision the case tests. None of these organizations is connected with the illustrative companies of this book.

Case, year	What the source states	Official report or primary source	Read with
GitLab.com database, 2017	On 31 January a primary database directory was removed by mistake while replication was being repaired; the backup methods in place did not work, the restore came from a snapshot taken about six hours earlier, and changes to roughly 5,000 projects, 5,000 comments and 700 new user accounts were lost.	GitLab, "Postmortem of database outage of January 31", 10 February 2017. about.gitlab.com/blog/postmortem-of-database-outage-of-january-31/	Ch 11: restore tests
TSB migration, 2018	The regulators found that TSB "failed to organise and control the IT migration programme adequately" and did not manage the operational risks of outsourcing to a critical supplier; fines of 29.75 million pounds (FCA) and 18.9 million pounds (PRA).	FCA press release, "TSB fined £48.65m for operational resilience failings", 20 December 2022. fca.org.uk/news/press-releases/tsb-fined-48m-operational-resilience-failings	Ch 15: migration mechanics
Capital One, 2019 incident	The OCC cited the bank's failure to establish effective risk assessment processes before moving major IT operations to the public cloud, and imposed an 80 million dollar civil money penalty.	OCC News Release 2020-101, 6 August 2020. occ.gov/news-issuances/news-releases/2020/nr-occ-2020-101.html	Ch 11: shared responsibility
Public Health England case reporting, 2020	15,841 positive cases between 25 September and 2 October were left out of the reported daily figures because some files exceeded the maximum file size the loading process accepted; files were then split.	Public Health England, "PHE statement on delayed reporting of COVID-19 cases", 4 October 2020, updated 5 October. gov.uk/government/news/phe-statement-on-delayed-reporting-of-covid-19-cases	Ch 6: what a file promises
Atlassian cloud sites, 2022	On 5 April a maintenance script given the IDs of whole sites instead of one app deleted 883 sites of 775 customers; all were restored by 18 April, some customers after up to 14 days.	Atlassian, "Post-Incident Review on the Atlassian April 2022 outage", 29 April 2022. atlassian.com/blog/atlassian-engineering/post-incident-review-april-2022-outage	Ch 11: metadata recovery
Meta Platforms Ireland, 2023	Transfers of personal data to the United States on standard contractual clauses after Schrems II infringed Article 46(1) GDPR; a 1.2 billion euro fine and orders to suspend transfers and stop unlawful storage in the United States.	Data Protection Commission, decision of 12 May 2023, announced 22 May 2023. dataprotection.ie	Ch 14: transfer, not location

Case, year	What the source states	Official report or primary source	Read with
UNC5537 and Snowflake customer accounts, 2024	Mandiant traced every incident it responded to in the campaign to compromised customer credentials, found no evidence of a breach of Snowflake's enterprise environment, and listed three factors: no multi-factor authentication, credentials from infostealer malware not rotated, no network allow lists; about 165 potentially exposed organizations notified.	Mandiant, "UNC5537 Targets Snowflake Customer Instances for Data Theft and Extortion", Google Cloud Threat Intelligence blog, 10 June 2024, sections "Introduction" (no breach of the enterprise environment) and "Campaign Overview" (the three factors). cloud.google.com/blog/topics/threat-intelligence/unc5537-snowflake-data-theft-extortion	Ch 11: identity and policy
Google Cloud VMware Engine, UniSuper, 2024	A parameter left blank when a private cloud was provisioned gave it a default one-year term, and it was deleted automatically at the end of that term without notice; backups in Cloud Storage were not affected and, with third-party backup software, were key to the restore.	Google Cloud, "Sharing details on a recent incident impacting one of our customers", 24 May 2024. cloud.google.com/blog/products/infrastructure/details-of-google-cloud-gcve-incident	Ch 15: exit and backups elsewhere

How to use the library

Read the cases by the decision they test, not by industry. Three of them (GitLab, Atlassian, the deleted private cloud) are restore cases: the data or the configuration was gone, and what mattered was whether a copy existed outside the failing system and whether anyone had restored from it before. Two (TSB, Capital One) are migration and cloud governance findings by regulators. One (Public Health England) is a format limit inside a pipeline. One (Meta) is a transfer finding: the basis for sending data to the United States, which the decision found wanting. The campaign Mandiant reported is about identity: the platform was used through customer accounts with stolen passwords. Each points to a Ledger column: who can change it, where the copy lives, and what the exit path really is.

APPENDIX D

Glossary

Terms used in this book, in plain words, as of November 2026.

Term	Meaning
ACID	Atomicity, consistency, isolation, durability: the guarantees of a transaction.
Appliance	A warehouse sold as one rack of hardware and software from one vendor.
Application time, system time	When a fact was true in the world; when the database recorded it (the SQL:2011 periods; the literature calls the first valid time).
As of	The time a number describes or was known at; in this book, its five times.
Bridge	A table that explains the difference between two figures, step by step.
Business associate agreement	The HIPAA contract a covered entity signs with a provider that handles its protected health information.
Case reserve	An insurer's estimate of what a reported claim will still cost.
Catalog	The register of tables: where their files are, their schemas, owners and grants.
Change data capture (CDC)	Reading a database's changes as a stream of events.
Columnar file	A file that stores each column together, such as Parquet or ORC.
Compaction	Rewriting many small data files into fewer large ones.
Cube	Pre-computed totals by dimension, read by OLAP tools.
Data lake	Files in their original or open formats on low-cost storage, structured when read.
Dialect	One engine's version of SQL: its functions, types and rounding.
Egress	Data sent out of a cloud provider's network, and its charge.
Exit inventory	The list of what an important result depends on, priced for a move.
Exit path	Whether an item travels, travels with work, or must be rebuilt.
Exit test	Reproducing an approved result on another catalog and engine, and explaining any difference.
Five times	Effective date, knowledge cutoff, calculation, approval and publication of a number.
IBNR	Reserves for losses incurred but not yet reported.
Iceberg REST catalog	An open interface through which engines find and commit Iceberg tables.
Knowledge cutoff	The last moment whose data a figure includes.

Term	Meaning
Lakehouse	Warehouse features (transactions, versions, governance) over open files on object storage.
Ledger	The Data Lineage Ledger: one row per hop, ten columns, eight loss codes.
Lock-in, deliberate	A dependency accepted on purpose, recorded with a reason and a review date.
Loss ratio	Losses incurred divided by premiums earned, for a period and basis.
Metric contract	The written definition of a number: what is counted, on which basis, rounded how.
Metric passport	Eleven rows that travel with an approved number, from id to restatement status.
MPP	Massively parallel processing: many nodes each work on a share of the data.
Object storage	Cloud storage of files as objects addressed by key.
Open table format	Metadata that turns a set of files into a table with versions: Iceberg, Delta Lake, Hudi.
Platform card	Ten rows that describe a platform, dated, in the same order for every platform.
Policies as code	Access policies and grants kept as reviewed text files and applied by a job.
Restatement	Replacing a published figure, with a note that gives the old and new values.
Row access policy	A rule that hides rows from some readers; it should not carry business logic.
Salvage and subrogation	Recoveries on a paid claim: sale of damaged property, or claims against a liable party.
Schema on read	Structure applied by the reader, not when data is written.
Semantic layer	Where metrics and business names are defined once for every tool.
Shared nothing	Nodes with their own processors, memory and disks, joined only by a network.
Snapshot	One committed version of a table, with an id.
Snapshot expiry	Removing old snapshots from table metadata under a retention policy.
Status 7	The insurer's code for a perpetual policy, deposit held, no premium due, in force.
Tag	In Iceberg, a named label for one snapshot, with its own retention.
Tier 1, tier 2	The cloud as substrate; a data platform running on it.
Time travel	Querying a table as it was at an earlier snapshot.
Type Contract	Seven lines on a value: type and rounding, encoding, unit, time basis, identity scope, absence, authority.
Workload class	A kind of work with its own service objective, such as board reporting or streaming ingest.
Zero-ETL	Vendor term for reading or replicating data without a pipeline the customer builds; still a hop.

APPENDIX E

Sources

Every dated fact in this book was checked against the source listed for its chapter in November 2026 (edition 1.1 rechecks on 20 November 2026). Primary sources come first: original papers, standards, statutes, judgments, official reports, vendor announcements and institutional histories. This list holds 364 sources; the claim-by-claim record behind it is kept with the book and is available on request.

Reader's Brief: One Number, Six Questions

- Book thread

Two Numbers for One Question

- NAIC, Glossary of Insurance Terms https://content.naic.org/glossary-insurance-terms
- Book thread
- NAIC, 2025 P&C Statement of Actuarial Opinion Instructions (exposure draft), paragraph 1 https://content.naic.org/sites/default/files/call_materials/027_Property_2025_Actuarial_Opinion_01272025_0.pdf
- Apache Iceberg documentation, Maintenance, Expire Snapshots https://iceberg.apache.org/docs/latest/maintenance/

Chapter 1. The Workload and the Metric Contract

- Casualty Actuarial Society, abstract page https://www.casact.org/abstract/reports-casualty-insurance-loss-reserve-schedules; paper https://www.casact.org/sites/default/files/database/proceed_proceed34_34050.pdf
- Jones, Wilkes and Murphy with Smith, "Service Level Objectives", Site Reliability Engineering, chapter 4, Google https://sre.google/sre-book/service-level-objectives/
- NAIC, Glossary of Insurance Terms https://content.naic.org/glossary-insurance-terms; NAIC Consumer Glossary https://content.naic.org/consumer_glossary
- NAIC, SSAP No. 55, Unpaid Claims, Losses and Loss Adjustment Expenses, paragraph 6 (NAIC text, copy hosted by the American Academy of Actuaries) https://www.actuary.org/wp-content/uploads/2017/11/055_RRQQ.pdf
- NAIC, SSAP No. 55, paragraph 14 (as above
- NAIC, P&C, Title, and Life Industry Snapshots for the Period Ended December 31, 2025 https://content.naic.org/sites/default/files/2025-ye-snapshot.pdf
- Krishna Kulkarni and Jan-Eike Michels, "Temporal features in SQL:2011", SIGMOD Record 41(3), September 2012, pp. 34 to 43 https://sigmodrecord.org/2012/09/30/temporal-features-in-sql2011/
- Richard T. Snodgrass, publications page, University of Arizona https://www2.cs.arizona.edu/~rts/publications.html
- Apache Iceberg documentation, Branching and Tagging https://iceberg.apache.org/docs/latest/branching/
- NAIC, Property/Casualty Annual Statement Instructions, Schedule P (text in NAIC Blanks Working Group materials, Property 2025 Schedule P Phase 2) https://content.naic.org/sites/default/files/call_materials/547_Property_2025_Schedule_P_Phase2_012726.docx
- Book thread
- NAIC Blanks (E) Working Group, proposal 2023-16BWG (modified) https://content.naic.org/sites/default/files/inline-files/2023-16BWG_Modified_0.pdf
- Apache Iceberg, Table Spec, Snapshots https://iceberg.apache.org/spec/

Chapter 2. Why the Warehouse Existed

- MIT DSpace record https://dspace.mit.edu/handle/1721.1/47936; scanned text, Internet Archive https://archive.org/details/frameworkformana00gorr

- same working paper, section on the framework https://archive.org/details/frameworkformana00gorr
- Crossref record, DOI 10.1147/sj.271.0060 https://doi.org/10.1147/sj.271.0060
- Barry Devlin, "Data Warehousing 3.0", TDWI, 1 February 2016 https://tdwi.org/articles/2016/02/01/data-warehousing-30.aspx
- Library of Congress catalog record, LCCN 91041284 https://lccn.loc.gov/91041284
- Library of Congress catalog record, LCCN 95050557 https://lccn.loc.gov/95050557
- Margy Ross, "Differences of Opinion", Kimball Group, 3 March 2004 https://www.kimballgroup.com/2004/03/differences-of-opinion/
- Kimball Group, "Kimball Data Warehouse Bus Architecture" https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/kimball-data-warehouse-bus-architecture/
- Margy Ross, "The Matrix: Revisited", 1 December 2005 https://decisionworks.com/2005/12/the-matrix-revisited/

Chapter 3. Cubes and the Frozen Answer

- White paper text https://staff.icar.cnr.it/manco/Teaching/2006/datamining/articoli/olapcoddwp.pdf
- Library of Congress catalog record, LCCN 95050557 https://lccn.loc.gov/95050557
- Microsoft Learn, "Partition Storage Modes and Processing" (SQL Server Analysis Services) https://learn.microsoft.com/en-us/analysis-services/multidimensional-models-olap-logical-cube-objects/partitions-partition-storage-modes-and-processing
- Microsoft News Center, "Microsoft Releases Specification for Universal OLAP Data Access", 9 September 1997 https://news.microsoft.com/source/1997/09/09/microsoft-releases-specification-for-universal-olap-data-access/
- Microsoft Learn (archived), "Multidimensional Expressions (OLE DB)" https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms714949(v=vs.85)

Chapter 4. Parallel Engines and the Appliance

- IEEE Computer Society, Database Engineering, March 1986 issue http://sites.computer.org/debull/86MAR-CD.pdf
- Crossref record, DOI 10.1109/MC.1984.1658999 https://doi.org/10.1109/MC.1984.1658999
- Computer History Museum, catalog 102713120 and 102713121 https://www.computerhistory.org/collections/catalog/102713120; https://www.computerhistory.org/collections/catalog/102713121
- Crossref record, DOI 10.1145/318898.318923 https://doi.org/10.1145/318898.318923; paper text https://cs-people.bu.edu/mathan/reading-groups/papers-classics/dsm.pdf
- Paper text, VLDB archive https://www.vldb.org/archives/website/2005/program/paper/thu/p553-stonebraker.pdf
- Netezza, "Netezza Unleashes Tera-scale Data Appliance for Business Intelligence" (hosted copy) https://dssresources.com/subscriber/password/news/news2002/september/netezza09232002.html
- Netezza Corporation, Form S-1, US Securities and Exchange Commission, 2007 https://www.sec.gov/Archives/edgar/data/1132484/000095013507001814/b64586s1sv1.htm
- Oracle, "Oracle Exadata: A guide for decision makers", version 2.11, September 2026 https://www.oracle.com/database/technologies/exadata/exadata-dmg/
- Armbrust, Ghodsi, Xin and Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics", CIDR 2021 https://vldb.org/cidrdb/2021/lakehouse-a-new-generation-of-open-platforms-that-unify-data-warehousing-and-advanced-analytics.html
- Teradata, "Teradata Cloud" product page https://www.teradata.com/platform/vantagecloud
- IBM, "IBM Db2 Warehouse" product page https://www.ibm.com/products/db2/warehouse
- Oracle, "Oracle Introduces Exadata X11M Platform", press release https://www.oracle.com/news/announcement/oracle-introduces-exadata-x11m-platform-2025-01-07/

Chapter 5. Files at Scale: Hadoop and the Lake

- Google Research publication page https://research.google/pubs/the-google-file-system/; paper PDF https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
- Ghemawat, Gobioff and Leung, SOSP 2003, sections 2.2, 2.3, 2.5 and 2.6.1 https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
- Google Research publication page https://research.google/pubs/mapreduce-simplified-data-processing-on-large-clusters/; paper PDF https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

- Apache JIRA, HADOOP-1 https://issues.apache.org/jira/browse/HADOOP-1
- Apache Hadoop, HDFS Architecture https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
- ASF board minutes, 16 January 2008 https://www.apache.org/foundation/records/minutes/2008/board_minutes_2008_01_16.txt; ASF board decisions for Hadoop https://www.apache.org/foundation/board/minutes/Hadoop.html
- Apache Hadoop news archive, item of 31 July 2008 https://hadoop.apache.org/news/page/2.html
- The Apache Software Foundation, "The Apache Software Foundation Announces Apache Hadoop v1.0" https://news.apache.org/foundation/entry/the_apache_software_foundation_announces21
- Apache Hadoop home page https://hadoop.apache.org/
- Paper PDF https://www.vldb.org/pvldb/vol2/vldb09-938.pdf DOI 10.14778/1687553.1687609
- ASF board decisions for Hive https://www.apache.org/foundation/board/minutes/Hive.html; Apache Hadoop news, "Hive and Pig Graduate" https://hadoop.apache.org/news/2010-10-xx-hive-pig.html
- James Dixon's Blog https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/
- Armbrust, Ghodsi, Xin and Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics", CIDR 2021 https://vldb.org/cidrdb/2021/lakehouse-a-new-generation-of-open-platforms-that-unify-data-warehousing-and-advanced-analytics.html

Chapter 6. Columns on Disk: Parquet, ORC and Arrow

- Google Research publication page https://research.google/pubs/dremel-interactive-analysis-of-web-scale-datasets-2/
- Paper PDF, Meta Research https://research.facebook.com/file/350031566825285/rcfile-a-fast-and-space-efficient-data-placement-structure-in-mapreduce-based-warehouse-systems.pdf DOI 10.1109/ICDE.2011.5767933
- Apache ORC documentation https://orc.apache.org/docs/
- Apache Hive, LanguageManual ORC https://hive.apache.org/docs/latest/language/languagemanual-orc/
- Apache Hive downloads (news) https://hive.apache.org/general/downloads/
- Apache ORC Specification v1 https://orc.apache.org/specification/ORCv1/
- Dmitriy Ryaboy, "Announcing Parquet 1.0: Columnar Storage for Hadoop", Twitter Blog (archived) https://web.archive.org/web/20130910203755/https://blog.twitter.com/2013/announcing-parquet-10-columnar-storage-for-hadoop
- The Apache Software Foundation blog https://blogsarchive.apache.org/foundation/entry/the_apache_software_foundation_announces75
- Apache Parquet format, README https://github.com/apache/parquet-format/blob/master/README.md
- Apache Parquet documentation, Concepts https://parquet.apache.org/docs/concepts/
- Apache Parquet documentation, File Format https://parquet.apache.org/docs/file-format/
- Apache Parquet documentation, Encodings https://parquet.apache.org/docs/file-format/data-pages/encodings/
- Apache Parquet documentation, Page Index https://parquet.apache.org/docs/file-format/pageindex/
- Apache Parquet format, LogicalTypes.md https://github.com/apache/parquet-format/blob/master/LogicalTypes.md
- Apache Parquet documentation, Parquet Modular Encryption https://parquet.apache.org/docs/file-format/data-pages/encryption/
- The Apache Software Foundation, "The Apache Software Foundation Announces Apache Arrow as a Top-Level Project" https://blogsarchive.apache.org/foundation/entry/the_apache_software_foundation_announces87
- Apache Arrow, Arrow Columnar Format https://arrow.apache.org/docs/format/Columnar.html
- Apache Parquet format, parquet.thrift https://github.com/apache/parquet-format/blob/master/src/main/thrift/parquet.thrift

Chapter 7. The Cloud Warehouse

- Google Developers Blog, "Google BigQuery brings Big Data analytics to all businesses" https://developers.googleblog.com/2012/05/google-bigquery-brings-big-data.html
- Melnik et al., "Dremel: Interactive Analysis of Web-Scale Datasets", PVLDB 3(1), 2010 https://www.vldb.org/pvldb/vldb2010/papers/R29.pdf
- Melnik et al., "Dremel: A Decade of Interactive SQL Analysis at Web Scale", PVLDB 13(12): 3461 to 3472, 2020 https://www.vldb.org/pvldb/vol13/p3461-melnik.pdf
- Google Cloud documentation, "BigQuery overview" https://docs.cloud.google.com/bigquery/docs/introduction
- Google Cloud, "BigQuery pricing" https://cloud.google.com/bigquery/pricing

- AWS What's New, "Announcing Amazon Redshift" https://aws.amazon.com/about-aws/whats-new/2012/11/28/announcing-amazon-redshift
- AWS What's New https://aws.amazon.com/about-aws/whats-new/2013/02/14/amazon-redshift-available-to-all-customers/
- AWS What's New, "Amazon Redshift introduces RA3 nodes with managed storage" https://aws.amazon.com/about-aws/whats-new/2019/12/amazon-redshift-announces-ra3-nodes-managed-storage
- Amazon Redshift Management Guide, "Amazon Redshift provisioned clusters" https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html
- AWS What's New, "Amazon Redshift Serverless is now generally available" https://aws.amazon.com/about-aws/whats-new/2022/07/amazon-redshift-serverless-generally-available
- Amazon Redshift Management Guide, "Billing for on-demand compute capacity" https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-billing-on-demand.html; "Billing for Amazon Redshift Serverless" https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-billing.html
- Snowflake press release, "Snowflake Redefines the Data Warehouse for the Cloud Era", 23 June 2015 https://www.snowflake.com/?p=1440
- Dageville et al., "The Snowflake Elastic Data Warehouse", SIGMOD 2016, DOI 10.1145/2882903.2903741, copy at https://www.cs.cmu.edu/~15721-f24/papers/Snowflake.pdf
- Snowflake documentation, "Overview of warehouses" https://docs.snowflake.com/en/user-guide/warehouses-overview
- Snowflake documentation, Understanding overall cost https://docs.snowflake.com/en/user-guide/cost-understanding-overall
- Snowflake documentation, "Understanding and using Time Travel" https://docs.snowflake.com/en/user-guide/data-time-travel
- Google Cloud documentation, "Data retention with time travel and fail-safe" https://docs.cloud.google.com/bigquery/docs/time-travel
- Sethi et al., "Presto: SQL on Everything", ICDE 2019, DOI 10.1109/ICDE.2019.00196, copy at https://trino.io/Presto_SQL_on_Everything.pdf
- Martin Traverso, Dain Sundstrom and David Phillips, "We're rebranding PrestoSQL as Trino", Trino blog, 27 December 2020 https://trino.io/blog/2020/12/27/announcing-trino
- AWS What's New, "AWS announces general availability of Amazon Aurora MySQL zero-ETL integration with Amazon Redshift" https://aws.amazon.com/about-aws/whats-new/2023/11/aws-general-availability-amazon-aurora-mysql-zero-etl-integration-redshift/
- Google Cloud documentation, "Introduction to federated queries" https://docs.cloud.google.com/bigquery/docs/federated-queries-intro
- Amazon, "Amazon Web Services Makes Amazon Redshift Available to All Customers", 15 Feb 2013 https://press.aboutamazon.com/2013/2/amazon-web-services-makes-amazon-redshift-available-to-all-customers; Jeff Barr, "Amazon Redshift: Now Broadly Available", AWS News Blog, 15 Feb 2013 https://aws.amazon.com/blogs/aws/amazon-redshift-now-broadly-available/
- Google Cloud, "Understand slots" https://docs.cloud.google.com/bigquery/docs/slots

Chapter 8. Tables, Snapshots and the Lakehouse

- Armbrust et al., "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores", PVLDB 13(12): 3411 to 3424, 2020 https://vldb.org/pvldb/vol13/p3411-armbrust.pdf
- same paper, sections 2 and 7
- Databricks press release https://www.databricks.com/company/newsroom/press-releases/databricks-open-sources-delta-lake-for-data-lake-reliability
- Apache Incubator, Iceberg project status https://incubator.apache.org/projects/iceberg.html Armbrust et al., CIDR 2021
- The Apache Software Foundation, "The Apache Software Foundation Announces Apache Hudi as a Top-Level Project", 4 June 2020 https://news.apache.org/foundation/entry/the-apache-software-foundation-announces64
- Apache Incubator, Hudi project status https://incubator.apache.org/projects/hudi.html
- Prasanna Rajaperumal and Vinoth Chandar, "Hudi: Uber Engineering's Incremental Processing Framework on Apache Hadoop", Uber Engineering blog https://www.uber.com/blog/hoodie/
- Apache Iceberg, "Iceberg Table Spec" https://iceberg.apache.org/spec/
- Apache Iceberg documentation 1.12.0, "Configuration", table behavior properties https://iceberg.apache.org/docs/latest/configuration/
- Apache Iceberg documentation, "Maintenance" https://iceberg.apache.org/docs/latest/maintenance/

- Apache Iceberg documentation, "Branching and Tagging" https://iceberg.apache.org/docs/latest/branching/
- Armbrust, Ghodsi, Xin and Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics", CIDR 2021 https://vldb.org/cidrdb/2021/lakehouse-a-new-generation-of-open-platforms-that-unify-data-warehousing-and-advanced-analytics.html
- Databricks blog, "Announcing Delta Lake 3.0 with New Universal Format and Liquid Clustering", 29 June 2023 https://www.databricks.com/blog/announcing-delta-lake-30-new-universal-format-and-liquid-clustering
- Databricks documentation, "Read Delta tables with Iceberg clients" (UniForm) https://docs.databricks.com/aws/en/delta/uniform
- Apache Incubator, XTable project status https://incubator.apache.org/projects/xtable.html
- Apache XTable (incubating), project site https://xtable.apache.org/
- Databricks press release, "Announcing Delta Lake 3.0 with New Universal Format" https://www.databricks.com/company/newsroom/press-releases/announcing-delta-lake-30-new-universal-format-offers-automatic
- Apache Iceberg documentation, "Spark Queries" https://iceberg.apache.org/docs/latest/spark-queries/
- Delta Lake GitHub release v3.0.0 https://github.com/delta-io/delta/releases/tag/v3.0.0
- Delta Lake GitHub release v3.2.0 https://github.com/delta-io/delta/releases/tag/v3.2.0

Chapter 9. Anatomy of a Platform

- Snowflake press release, "Snowflake Unveils Polaris Catalog and Emphasizes Commitment to Interoperability with AWS, Google Cloud, Microsoft Azure, Salesforce, and More", 3 June 2024 https://www.snowflake.com/en/news/press-releases/snowflake-unveils-polaris-catalog-and-emphasizes-commitment-to-interoperability-with-aws-google-cloud-microsoft-azure-salesforce-and-more-2/
- Matei Zaharia, Ali Ghodsi, Reynold Xin, Arsalan Tavakoli-Shiraji and Patrick Wendell, "Open Sourcing Unity Catalog", Databricks blog, 13 June 2024 https://www.databricks.com/blog/open-sourcing-unity-catalog
- Apache Iceberg Table Spec https://iceberg.apache.org/spec/
- Armbrust, Ghodsi, Xin and Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics", CIDR 2021 https://vldb.org/cidrdb/2021/lakehouse-a-new-generation-of-open-platforms-that-unify-data-warehousing-and-advanced-analytics.html
- Oracle Technology Network License Agreement https://www.oracle.com/downloads/licenses/standard-license.html

Chapter 10. Catalog, Semantics and Policy

- ISO, ISO/IEC 9075:1992 catalog page https://www.iso.org/standard/16663.html read through the Internet Archive copy of 20 October 2026
- Information Technology, Database Language SQL (Proposed revised text of DIS 9075), July 1992, section 4.32 http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt
- Apache Hive, Design https://cwiki.apache.org/confluence/display/Hive/Design
- AWS What's New, "Introducing AWS Glue", 14 August 2017 https://aws.amazon.com/about-aws/whats-new/2017/08/introducing-aws-glue-a-simple-flexible-and-cost-effective-extract-transfer-and-load-etl-service
- AWS What's New, "Use Apache Spark and Hive on Amazon EMR with the AWS Glue Data Catalog", 14 August 2017 https://aws.amazon.com/about-aws/whats-new/2017/08/use-apache-spark-and-hive-on-amazon-emr-with-the-aws-glue-data-catalog
- AWS Glue Developer Guide, Data discovery and cataloging in AWS Glue https://docs.aws.amazon.com/glue/latest/dg/catalog-and-crawler.html
- AWS Glue Developer Guide, AWS Glue REST APIs for Apache Iceberg specifications https://docs.aws.amazon.com/glue/latest/dg/iceberg-rest-apis.html
- Apache Iceberg, rest-catalog-open-api.yaml https://github.com/apache/iceberg/blob/main/open-api/rest-catalog-open-api.yaml
- Apache Iceberg GitHub release apache-iceberg-1.12.0 https://github.com/apache/iceberg/releases/tag/apache-iceberg-1.12.0; rest-catalog-open-api.yaml at that tag https://github.com/apache/iceberg/blob/apache-iceberg-1.12.0/open-api/rest-catalog-open-api.yaml
- Snowflake press release, 3 June 2024 https://www.snowflake.com/en/news/press-releases/snowflake-unveils-polaris-catalog-and-emphasizes-commitment-to-interoperability-with-aws-google-cloud-microsoft-azure-salesforce-and-more-2/
- Apache Incubator, Polaris project status https://incubator.apache.org/projects/polaris.html
- Apache Polaris documentation, Access control https://polaris.apache.org/in-dev/unreleased/managing-security/access-control/; Apache Polaris home page https://polaris.apache.org/

- Zaharia et al., "Open Sourcing Unity Catalog", Databricks blog, 13 June 2024 https://www.databricks.com/blog/open-sourcing-unity-catalog
- Google Cloud, Looker documentation, What is LookML? https://docs.cloud.google.com/looker/docs/what-is-lookml
- Microsoft Learn, Semantic models in the Power BI service https://learn.microsoft.com/en-us/power-bi/connect-data/service-datasets-understand
- dbt Labs documentation, dbt Semantic Layer https://docs.getdbt.com/docs/use-dbt-semantic-layer/dbt-sl
- Snowflake press release, "Snowflake, Salesforce, dbt Labs, and More, Revolutionize Data Readiness for AI with Open Semantic Interchange Initiative", 23 September 2025 https://www.snowflake.com/en/news/press-releases/snowflake-salesforce-dbt-labs-and-more-revolutionize-data-readiness-for-ai-with-open-semantic-interchange-initiative/
- OpenLineage documentation https://openlineage.io/docs/
- Bitol home page https://bitol.io/; Open Data Contract Standard https://bitol-io.github.io/open-data-contract-standard/latest/
- Hu et al., NIST SP 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations https://doi.org/10.6028/NIST.SP.800-162; publication page https://csrc.nist.gov/pubs/sp/800/162/upd2/final
- Snowflake documentation, Understanding row access policies https://docs.snowflake.com/en/user-guide/security-row-intro
- Google Cloud, BigQuery documentation, Introduction to column-level access control https://docs.cloud.google.com/bigquery/docs/column-level-security-intro
- Databricks documentation, Row filters and column masks https://docs.databricks.com/aws/en/data-governance/unity-catalog/filters-and-masks/
- Apache Iceberg documentation, Configuration https://iceberg.apache.org/docs/latest/configuration/

Chapter 11. Security and Resilience on a Shared Platform

- Snowflake documentation, Planning for the deprecation of single-factor password sign-ins https://docs.snowflake.com/en/user-guide/security-mfa-rollout
- AWS, Shared Responsibility Model https://aws.amazon.com/compliance/shared-responsibility-model/
- Microsoft Learn, Shared responsibility in the cloud https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility
- OASIS, Security Assertion Markup Language (SAML) v2.0 https://www.oasis-open.org/standard/saml/
- OpenID Foundation, OpenID Connect Core 1.0 incorporating errata set 2 https://openid.net/specs/openid-connect-core-1_0.html
- IETF, RFC 7644, System for Cross-domain Identity Management: Protocol https://www.rfc-editor.org/rfc/rfc7644
- Google Developers Blog, "Google BigQuery brings Big Data analytics to all businesses" https://developers.googleblog.com/2012/05/google-bigquery-brings-big-data.html
- Snowflake documentation, Understanding row access policies https://docs.snowflake.com/en/user-guide/security-row-intro
- Snowflake documentation, Tag-based masking policies https://docs.snowflake.com/en/user-guide/tag-based-masking-policies
- Google Cloud, BigQuery documentation, Introduction to column-level access control https://docs.cloud.google.com/bigquery/docs/column-level-security-intro
- Mandiant (Google Cloud Threat Intelligence), "UNC5537 Targets Snowflake Customer Instances for Data Theft and Extortion" https://cloud.google.com/blog/topics/threat-intelligence/unc5537-snowflake-data-theft-extortion
- same report, section Introduction
- same report, section Campaign Overview
- same report, section Outlook and Implications
- Snowflake documentation, Tri-Secret Secure https://docs.snowflake.com/en/user-guide/security-encryption-tss
- AWS What's New, "AWS KMS launches External Key Store", 29 November 2022 https://aws.amazon.com/about-aws/whats-new/2022/11/aws-kms-external-key-store/
- Google Cloud, Cloud External Key Manager https://docs.cloud.google.com/kms/docs/ekm
- HHS Office for Civil Rights, Guidance on HIPAA and Cloud Computing https://www.hhs.gov/hipaa/for-professionals/special-topics/health-information-technology/cloud-computing/index.html read through the Internet Archive copy of 9 November 2026; no issue date on the page
- DICOM PS3.3, section C.7.6.1 General Image Module https://dicom.nema.org/medical/dicom/current/output/chtml/part03/sect_C.7.6.html

- DICOM PS3.15, section E.3.1 Clean Pixel Data Option https://dicom.nema.org/medical/dicom/current/output/chtml/part15/sect_E.3.html
- Zaharia, Armbrust, Weis et al., "Introducing Delta Sharing: An Open Protocol for Secure Data Sharing", Databricks blog, 26 May 2021 https://www.databricks.com/blog/2021/05/26/introducing-delta-sharing-an-open-protocol-for-secure-data-sharing.html
- Snowflake documentation, About Secure Data Sharing https://docs.snowflake.com/en/user-guide/data-sharing-intro
- AWS What's New, "AWS Clean Rooms is now generally available", 21 March 2023 https://aws.amazon.com/about-aws/whats-new/2023/03/aws-clean-rooms-generally-available/
- Apache Iceberg documentation, Configuration https://iceberg.apache.org/docs/latest/configuration/
- Apache Iceberg documentation, Spark Procedures https://iceberg.apache.org/docs/latest/spark-procedures/
- Snowflake documentation, Understanding and using Time Travel https://docs.snowflake.com/en/user-guide/data-time-travel
- Snowflake documentation, Understanding and viewing Fail-safe https://docs.snowflake.com/en/user-guide/data-failsafe
- Snowflake documentation, Introduction to replication and failover across multiple accounts https://docs.snowflake.com/en/user-guide/account-replication-intro
- AWS Clean Rooms User Guide, "Analysis rules in AWS Clean Rooms" https://docs.aws.amazon.com/clean-rooms/latest/userguide/analysis-rules.html

Chapter 12. Seven Cards: Six Platforms and an Open Stack

- Microsoft Fabric Blog, "Prepare your data for AI innovation with Microsoft Fabric, now generally available", 15 Nov 2023 https://www.microsoft.com/en-us/microsoft-fabric/blog/2023/11/15/prepare-your-data-for-ai-innovation-with-microsoft-fabric-now-generally-available/
- Snowflake release notes, "June 10, 2024: Apache Iceberg tables: General Availability" https://docs.snowflake.com/en/release-notes/2024/other/2024-06-10-iceberg-tables
- AWS What's New, 3 Dec 2024 https://aws.amazon.com/about-aws/whats-new/2024/12/amazon-s3-tables-apache-iceberg-tables-analytics-workloads/
- Databricks press release, "Databricks Eliminates Table Format Lock-in and Adds Capabilities for Business Users with Unity Catalog Advancements", 11 Jun 2025 https://www.databricks.com/company/newsroom/press-releases/databricks-eliminates-table-format-lock-and-adds-capabilities
- Apache Incubator, Polaris project incubation status https://incubator.apache.org/projects/polaris.html
- Amazon S3 User Guide, "Working with Amazon S3 Tables and table buckets" https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-tables.html
- AWS Glue Developer Guide, "Connecting to the Data Catalog using AWS Glue Iceberg REST endpoint" https://docs.aws.amazon.com/glue/latest/dg/connect-glu-iceberg-rest.html
- AWS Lake Formation Developer Guide, "Data filtering and cell-level security in Lake Formation" https://docs.aws.amazon.com/lake-formation/latest/dg/data-filtering.html
- Amazon Athena User Guide, "Query Apache Iceberg tables" https://docs.aws.amazon.com/athena/latest/ug/querying-iceberg.html
- Amazon Athena User Guide, "Query Linux Foundation Delta Lake tables" https://docs.aws.amazon.com/athena/latest/ug/delta-lake-tables.html
- Amazon Athena pricing https://aws.amazon.com/athena/pricing/
- Amazon S3 User Guide, "Protecting data with encryption" https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingEncryption.html
- AWS KMS Developer Guide, "External key stores" https://docs.aws.amazon.com/kms/latest/developerguide/keystore-external.html
- Amazon Redshift Database Developer Guide, "Amazon Redshift and PostgreSQL" https://docs.aws.amazon.com/redshift/latest/dg/c_redshift-and-postgres-sql.html
- Amazon Redshift Management Guide, "Billing for Amazon Redshift Serverless" https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-billing.html
- Amazon S3 pricing https://aws.amazon.com/s3/pricing/
- Microsoft Learn, "Azure Data Lake Storage overview" https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction
- Microsoft Learn, "Customer-managed keys for account encryption" https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview

- Microsoft Learn, "Serverless SQL pool in Azure Synapse Analytics" https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/on-demand-workspace-overview
- Microsoft Learn, "Learn about Microsoft Purview Unified Catalog" https://learn.microsoft.com/en-us/purview/unified-catalog
- Microsoft Tech Community, Microsoft Security Blog, "A next step in the evolution of the Microsoft data governance strategy", 12 November 2026 https://techcommunity.microsoft.com/blog/microsoft-security-blog/a-next-step-in-the-evolution-of-the-microsoft-data-governance-strategy/4572227
- Azure bandwidth pricing https://azure.microsoft.com/en-us/pricing/details/bandwidth/
- Microsoft Learn, "What is Azure Databricks?" https://learn.microsoft.com/en-us/azure/databricks/introduction/
- Google Cloud, "Overview of BigQuery storage" https://docs.cloud.google.com/bigquery/docs/storage_overview
- Google Cloud, BigQuery Iceberg managed tables https://docs.cloud.google.com/bigquery/docs/iceberg-tables
- Google Cloud, Iceberg REST catalog endpoint https://docs.cloud.google.com/bigquery/docs/blms-rest-catalog; Lakehouse introduction https://docs.cloud.google.com/biglake/docs/introduction
- Google Cloud, "Introduction to BigQuery row-level security" https://docs.cloud.google.com/bigquery/docs/row-level-security-intro; "Introduction to column-level access control" https://docs.cloud.google.com/bigquery/docs/column-level-security-intro
- Google Cloud, "BigQuery IAM roles and permissions" https://docs.cloud.google.com/bigquery/docs/access-control
- Google Cloud, "Customer-managed Cloud KMS keys" https://docs.cloud.google.com/bigquery/docs/customer-managed-encryption
- Google Cloud, "Cloud External Key Manager" https://docs.cloud.google.com/kms/docs/ekm
- Google Cloud, "Introduction to SQL in BigQuery" https://docs.cloud.google.com/bigquery/docs/introduction-sql
- Google Cloud, "Understand reservations" https://docs.cloud.google.com/bigquery/docs/reservations-intro
- Cloud Storage pricing https://cloud.google.com/storage/pricing
- Databricks on AWS, "High-level architecture" https://docs.databricks.com/aws/en/getting-started/high-level-architecture; Databricks on Google Cloud, same page https://docs.databricks.com/gcp/en/getting-started/high-level-architecture
- Databricks, "What is Unity Catalog?" https://docs.databricks.com/aws/en/data-governance/unity-catalog/
- Databricks, "Access Databricks tables from Apache Iceberg clients" https://docs.databricks.com/aws/en/external-access/iceberg
- Databricks, "Unity Catalog managed tables" https://docs.databricks.com/aws/en/tables/managed
- Databricks, "Row filters and column masks" https://docs.databricks.com/aws/en/tables/row-and-column-filters
- Databricks, "Customer-managed keys for encryption" https://docs.databricks.com/aws/en/security/keys/customer-managed-keys
- Databricks, "Unity Catalog metric views" https://docs.databricks.com/aws/en/metric-views/; metric views queried by people and AI tools, AI/BI dashboards, Genie: Microsoft Learn, "What is Azure Databricks?" https://learn.microsoft.com/en-us/azure/databricks/introduction/
- Databricks SQL reference, "ANSI_MODE" https://docs.databricks.com/aws/en/sql/language-manual/parameters/ansi_mode
- Databricks pricing https://www.databricks.com/product/pricing
- Snowflake, "Key concepts and architecture" https://docs.snowflake.com/en/user-guide/intro-key-concepts
- Snowflake, "Apache Iceberg tables" https://docs.snowflake.com/en/user-guide/tables-iceberg
- Snowflake, "Query Iceberg tables with an external engine through Snowflake Horizon Catalog" https://docs.snowflake.com/en/user-guide/tables-iceberg-query-using-external-query-engine-snowflake-horizon
- Snowflake, "Snowflake Open Catalog overview" https://docs.snowflake.com/en/user-guide/opencatalog/overview
- Snowflake, "Overview of semantic views" https://docs.snowflake.com/en/user-guide/views-semantic/overview
- Snowflake, "Understanding row access policies" https://docs.snowflake.com/en/user-guide/security-row-intro
- Snowflake, "Understanding Column-level Security" https://docs.snowflake.com/en/user-guide/security-column-intro
- Snowflake, "Tri-Secret Secure" https://docs.snowflake.com/en/user-guide/security-encryption-tss
- Snowflake, "Understanding compute cost" https://docs.snowflake.com/en/user-guide/cost-understanding-compute
- Snowflake, "Understanding overall cost" https://docs.snowflake.com/en/user-guide/cost-understanding-overall
- Microsoft Learn, "What is Microsoft Fabric" https://learn.microsoft.com/en-us/fabric/fundamentals/microsoft-fabric-overview
- Microsoft Learn, "OneLake, the unified data lake" https://learn.microsoft.com/en-us/fabric/onelake/onelake-overview
- Microsoft Learn, "Use Iceberg tables with OneLake" https://learn.microsoft.com/en-us/fabric/onelake/onelake-iceberg-tables
- Microsoft Learn, "Understand Microsoft Fabric licenses and capacity" https://learn.microsoft.com/en-us/fabric/enterprise/licenses

- Microsoft Learn, "OneLake consumption" https://learn.microsoft.com/en-us/fabric/onelake/onelake-consumption
- Microsoft Learn, "Customer-managed keys for Fabric workspaces" https://learn.microsoft.com/en-us/fabric/security/workspace-customer-managed-keys
- Microsoft Learn, "What is Fabric Data Warehouse?" https://learn.microsoft.com/en-us/fabric/data-warehouse/data-warehousing
- Apache Polaris https://polaris.apache.org/
- Apache Polaris 1.2.0 documentation, "Access control" https://polaris.apache.org/releases/1.2.0/managing-security/access-control/
- Apache Iceberg, "Configuration" https://iceberg.apache.org/docs/latest/configuration/
- Apache Iceberg, "Spark Configuration" https://iceberg.apache.org/docs/latest/spark-configuration/
- Apache Iceberg, rest-catalog-open-api.yaml https://github.com/apache/iceberg/blob/main/open-api/rest-catalog-open-api.yaml
- Trino documentation, "Iceberg connector" https://trino.io/docs/current/connector/iceberg.html
- Trino documentation, "SQL language" https://trino.io/docs/current/language.html
- Apache Spark, "ANSI Compliance" https://spark.apache.org/docs/latest/sql-ref-ansi-compliance.html
- Apache Spark, "Migration Guide: SQL, Datasets and DataFrame" https://spark.apache.org/docs/latest/sql-migration-guide.html
- Google Cloud, Looker documentation, What is LookML? https://docs.cloud.google.com/looker/docs/what-is-lookml

Chapter 13. What Work Costs

- Google Cloud, BigQuery pricing https://cloud.google.com/bigquery/pricing
- Google Cloud, BigQuery documentation, Best practices for controlling costs https://docs.cloud.google.com/bigquery/docs/best-practices-costs
- Snowflake documentation, Understanding compute cost https://docs.snowflake.com/en/user-guide/cost-understanding-compute
- Snowflake documentation, Understanding overall cost https://docs.snowflake.com/en/user-guide/cost-understanding-overall
- Databricks, Pricing https://www.databricks.com/product/pricing
- Microsoft Learn, Understand Microsoft Fabric licenses and capacity https://learn.microsoft.com/en-us/fabric/enterprise/licenses
- AWS documentation, Billing for Amazon Redshift Serverless https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-billing.html
- AWS, Amazon S3 pricing https://aws.amazon.com/s3/pricing/
- FinOps Foundation, "FOCUS 1.0 available" https://www.finops.org/insights/focus-1-0-available/
- FinOps Foundation, "Introducing FOCUS 1.4" https://www.finops.org/insights/introducing-focus-1-4/
- FOCUS https://focus.finops.org/
- FOCUS Specification v1.4, Introduction https://focus.finops.org/docs/specification/v1-4/sections/introduction
- FOCUS Specification v1.4, Columns https://focus.finops.org/docs/specification/v1-4/columns
- Google Cloud, Understand slots https://docs.cloud.google.com/bigquery/docs/slots
- FOCUS Specification v1.0 and v1.4 https://focus.finops.org/focus-specification/v1-0/

Chapter 14. Location, Transfer and Jurisdiction

- Regulation (EU) 2016/679 (GDPR), OJ L 119, 4.5.2016, EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679
- EDPB, Guidelines 05/2021 on the Interplay between the application of Article 3 and the provisions on international transfers as per Chapter V of the GDPR, version 2.0 https://www.edpb.europa.eu/system/files/2023-02/edpb_guidelines_05-2021_interplay_between_the_application_of_art3-chapter_v_of_the_gdpr_v2_en_0.pdf
- same, EDPB Guidelines 05/2021 v2.0
- EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:62018CJ0311
- EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32023D1795
- Court of Justice of the European Union, Press Release No 106/25 https://curia.europa.eu/site/upload/docs/application/pdf/2025-09/cp250106en.pdf
- Official Journal notice C/2025/6610, EUR-Lex https://eur-lex.europa.eu/eli/C/2025/6610/oj/eng

- Pub. L. 115-141, govinfo https://www.govinfo.gov/content/pkg/PLAW-115publ141/html/PLAW-115publ141.htm
- AWS News Blog, "Opening the AWS European Sovereign Cloud" https://aws.amazon.com/blogs/aws/opening-the-aws-european-sovereign-cloud/
- Microsoft On the Issues, "Microsoft completes landmark EU Data Boundary", 26 Feb 2025 https://blogs.microsoft.com/on-the-issues/2025/02/26/microsoft-completes-landmark-eu-data-boundary-offering-enhanced-data-residency-and-transparency/
- HHS Office for Civil Rights, Guidance on HIPAA and Cloud Computing https://www.hhs.gov/hipaa/for-professionals/special-topics/health-information-technology/cloud-computing/index.html; (read via Web Archive copy of 9 Nov 2026 https://web.archive.org/web/20261109195103/https://www.hhs.gov/hipaa/for-professionals/special-topics/health-information-technology/cloud-computing/index.html
- same, HHS Guidance on HIPAA and Cloud Computing
- eCFR, 45 CFR 164.514 https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164/subpart-E/section-164.514
- Court of Justice of the European Union, Press Release No 117/15 https://curia.europa.eu/jcms/upload/docs/application/pdf/2015-10/cp150117en.pdf
- Regulation (EU) 2016/679, Art. 48, EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679
- 18 U.S.C. 2703, Legal Information Institute https://www.law.cornell.edu/uscode/text/18/2703
- US Department of Justice, CLOUD Act Resources https://www.justice.gov/criminal/cloud-act-resources
- 45 CFR 164.514, Legal Information Institute https://www.law.cornell.edu/cfr/text/45/164.514
- EDPB Guidelines 05/2021 on the interplay between the application of Article 3 and the provisions on international transfers as per Chapter V of the GDPR, version 2.0 https://www.edpb.europa.eu/system/files/documents/2023-02/edpb_guidelines_05-2021_interplay_between_the_application_of_art3-chapter_v_of_the_gdpr_v2_en_0.pdf

Chapter 15. Migration and the Price of Leaving

- Amit Zavery, "Cloud switching just got easier: Removing data transfer fees when moving off Google Cloud", Google Cloud Blog, 11 Jan 2024 https://cloud.google.com/blog/products/networking/eliminating-data-transfer-fees-when-migrating-off-google-cloud
- Google Cloud, Exit Google Cloud with free data transfer https://cloud.google.com/exit-cloud
- Sebastien Stormacq, "Free data transfer out to internet when moving out of AWS", AWS News Blog, 5 Mar 2024 https://aws.amazon.com/blogs/aws/free-data-transfer-out-to-internet-when-moving-out-of-aws/
- same, AWS News Blog post with dated updates
- Microsoft Learn, Cancel and delete your Azure subscription (page dated 7 Jun 2026) https://learn.microsoft.com/en-us/azure/cost-management-billing/manage/cancel-azure-subscription
- Microsoft Azure, Bandwidth pricing https://azure.microsoft.com/en-us/pricing/details/bandwidth/
- Regulation (EU) 2023/2854, OJ L, 22.12.2023, EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32023R2854
- same, Art. 2
- same, Art. 25
- same, Art. 29
- same, Art. 31
- same, Art. 34
- European Parliament Legislative Observatory, procedure 2025/0360(COD) https://oeil.europarl.europa.eu/oeil/en/procedure-file?reference=2025/0360(COD)
- CMA case page https://www.gov.uk/cma-cases/cloud-services-market-investigation
- CMA, Cloud services market investigation, Summary of final decision, 31 July 2025 https://assets.publishing.service.gov.uk/media/688b20e6ff8c05468cb7b120/summary_of_final_decision.pdf
- CMA, Update on cloud actions, covering March to October 2026, 16 Oct 2026 https://assets.publishing.service.gov.uk/media/6ad1cb60e5db84746b4796d1/Update_on_cloud_actions_-_covering_March_to_October_2026.pdf
- CMA case page, Microsoft's business software ecosystem https://www.gov.uk/cma-cases/microsofts-business-software-ecosystem
- Apache Iceberg Table Spec https://iceberg.apache.org/spec/; Apache Iceberg, Spark Procedures https://iceberg.apache.org/docs/latest/spark-procedures/
- Apache Iceberg, REST Catalog OpenAPI specification https://github.com/apache/iceberg/blob/main/open-api/rest-catalog-open-api.yaml

- Apache Iceberg, Releases https://iceberg.apache.org/releases/; Spark Procedures https://iceberg.apache.org/docs/latest/spark-procedures/
- Regulation (EU) 2023/2854 https://publications.europa.eu/resource/celex/32023R2854
- Apache Spark, SQL math functions https://spark.apache.org/docs/latest/api/sql/math-functions/; Databricks, round and bround functions https://docs.databricks.com/aws/en/sql/language-manual/functions/bround

Chapter 16. Choosing per Workload

- "Announcing Delta Lake 3.0 with New Universal Format and Liquid Clustering", Databricks blog, 29 Jun 2023 https://www.databricks.com/blog/announcing-delta-lake-30-new-universal-format-and-liquid-clustering
- Apache Incubator, XTable project status https://incubator.apache.org/projects/xtable.html
- Transaction Processing Performance Council, TPC-DS https://www.tpc.org/tpcds/

Chapter 17. A Platform Ledger and Exit Test in 90 Days

- Apache Iceberg, Configuration (table properties) https://iceberg.apache.org/docs/latest/configuration/
- Apache Iceberg Table Spec, Snapshot References and Snapshot Retention Policy https://iceberg.apache.org/spec/
- Google Cloud, "Exit free data transfer request" https://cloud.google.com/exit-cloud; Amit Zavery, "Cloud switching just got easier: Removing data transfer fees when moving off Google Cloud", Google Cloud Blog, 11 Jan 2024 https://cloud.google.com/blog/products/networking/eliminating-data-transfer-fees-when-migrating-off-google-cloud
- Sebastien Stormacq, "Free data transfer out to internet when moving out of AWS", AWS News Blog, 5 Mar 2024, updated 8 Oct 2026 https://aws.amazon.com/blogs/aws/free-data-transfer-out-to-internet-when-moving-out-of-aws/
- Regulation (EU) 2023/2854 (Data Act), EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32023R2854
- Apache Iceberg, Spark Procedures, expire_snapshots https://iceberg.apache.org/docs/latest/spark-procedures/
- From Tally to Token (Every Record Counts, Book 1), chapter 15

Epilogue: The Format Outlives the Platform

- Dmitriy Ryaboy, "Announcing Parquet 1.0: Columnar Storage for Hadoop", Twitter Blog (archived) https://web.archive.org/web/20130910203755/https://blog.twitter.com/2013/announcing-parquet-10-columnar-storage-for-hadoop
- Apache Incubator, Polaris project status https://incubator.apache.org/projects/polaris.html
- Apache Polaris, project site https://polaris.apache.org/
- Matei Zaharia et al., "Open Sourcing Unity Catalog", Databricks blog, 13 June 2024 https://www.databricks.com/blog/open-sourcing-unity-catalog
- John R. Lange, "Reports of Casualty Insurance: Loss Reserve Schedules", Proceedings of the Casualty Actuarial Society, Vol. XXI, 1934 https://www.casact.org/abstract/reports-casualty-insurance-loss-reserve-schedules
- NAIC, Schedule P instructions (Blanks Working Group materials) https://content.naic.org/sites/default/files/call_materials/547_Property_2025_Schedule_P_Phase2_012726.docx
- Book thread

Appendix A. Templates: Ledger, Metric Passport, Platform Card, Exit Inventory

- Apache Iceberg Table Spec, Snapshot References and Snapshot Retention Policy https://iceberg.apache.org/spec/

Appendix B. Timeline

- MIT DSpace record https://dspace.mit.edu/handle/1721.1/47936
- Crossref record, DOI 10.1145/318898.318923 https://doi.org/10.1145/318898.318923
- Database Engineering, March 1986, Vol. 9 No. 1 http://sites.computer.org/debull/86MAR-CD.pdf
- Crossref record, DOI 10.1147/sj.271.0060 https://doi.org/10.1147/sj.271.0060
- Library of Congress catalog record, LCCN 91041284 https://lccn.loc.gov/91041284
- White paper text https://staff.icar.cnr.it/manco/Teaching/2006/datamining/articoli/olapcoddwp.pdf

- Library of Congress catalog record, LCCN 95050557 https://lccn.loc.gov/95050557
- Crossref record, DOI 10.1145/945445.945450 https://doi.org/10.1145/945445.945450
- USENIX, OSDI '04 technical paper page https://www.usenix.org/legacy/event/osdi04/tech/dean.html
- VLDB 2005 paper https://www.vldb.org/archives/website/2005/program/paper/thu/p553-stonebraker.pdf
- Crossref record, DOI 10.14778/1920841.1920886 https://doi.org/10.14778/1920841.1920886
- James Dixon's blog https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/
- Kulkarni and Michels, "Temporal features in SQL:2011", SIGMOD Record 41(3), 2012, DOI 10.1145/2380776.2380786 https://doi.org/10.1145/2380776.2380786
- Google Developers Blog, "Google BigQuery brings Big Data analytics to all businesses" https://developers.googleblog.com/2012/05/google-bigquery-brings-big-data.html
- AWS What's New, "Announcing Amazon Redshift" https://aws.amazon.com/about-aws/whats-new/2012/11/28/announcing-amazon-redshift
- AWS What's New https://aws.amazon.com/about-aws/whats-new/2013/02/14/amazon-redshift-available-to-all-customers/
- Dmitriy Ryaboy, "Announcing Parquet 1.0: Columnar Storage for Hadoop", Twitter Blog (archived) https://web.archive.org/web/20130910203755/https://blog.twitter.com/2013/announcing-parquet-10-columnar-storage-for-hadoop
- Snowflake press release, "Snowflake Redefines the Data Warehouse for the Cloud Era", 23 June 2015 https://www.snowflake.com/?p=1440
- The Apache Software Foundation, "The Apache Software Foundation Announces Apache Arrow as a Top-Level Project" https://news.apache.org/foundation/entry/the_apache_software_foundation_announces87
- Apache Incubator, Iceberg project status https://incubator.apache.org/projects/iceberg.html
- Databricks press release https://www.databricks.com/company/newsroom/press-releases/databricks-open-sources-delta-lake-for-data-lake-reliability
- AWS What's New, "Amazon Redshift introduces RA3 nodes with managed storage" https://aws.amazon.com/about-aws/whats-new/2019/12/amazon-redshift-announces-ra3-nodes-managed-storage
- EUR-Lex, CELEX 62018CJ0311 https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:62018CJ0311
- "Prepare your data for AI innovation with Microsoft Fabric, now generally available", Microsoft Fabric Blog, 15 Nov 2023 https://www.microsoft.com/en-us/microsoft-fabric/blog/2023/11/15/prepare-your-data-for-ai-innovation-with-microsoft-fabric-now-generally-available/
- Amit Zavery, "Cloud switching just got easier: Removing data transfer fees when moving off Google Cloud", Google Cloud Blog, 11 Jan 2024 https://cloud.google.com/blog/products/networking/eliminating-data-transfer-fees-when-migrating-off-google-cloud
- Sebastien Stormacq, "Free data transfer out to internet when moving out of AWS", AWS News Blog, 5 Mar 2024, updated 8 Oct 2026 https://aws.amazon.com/blogs/aws/free-data-transfer-out-to-internet-when-moving-out-of-aws/
- Mandiant, "UNC5537 Targets Snowflake Customer Instances for Data Theft and Extortion", Google Cloud Threat Intelligence blog, 10 Jun 2024 https://cloud.google.com/blog/topics/threat-intelligence/unc5537-snowflake-data-theft-extortion
- Matei Zaharia et al., "Open Sourcing Unity Catalog", Databricks blog, 13 Jun 2024 https://www.databricks.com/blog/open-sourcing-unity-catalog
- CMA case page https://www.gov.uk/cma-cases/cloud-services-market-investigation
- Regulation (EU) 2023/2854 (Data Act), EUR-Lex https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32023R2854
- Apache Incubator, Polaris project status https://incubator.apache.org/projects/polaris.html
- USENIX, OSDI '04 invitation https://usenix.org/events/osdi04/invitation.html
- Databricks newsroom https://www.databricks.com/company/newsroom/press-releases/databricks-open-sources-unity-catalog-creating-industrys-only-open
- ISO, ISO/IEC 9075-2:2011 https://committee.iso.org/standard/53682.html
- The Apache Software Foundation, news.apache.org https://news.apache.org/?p=7401

Appendix C. Failure Library

- GitLab, "Postmortem of database outage of January 31", 10 Feb 2017 https://about.gitlab.com/blog/postmortem-of-database-outage-of-january-31/
- FCA press release, "TSB fined £48.65m for operational resilience failings" https://www.fca.org.uk/news/press-releases/tsb-fined-48m-operational-resilience-failings
- OCC News Release 2020-101 https://www.occ.gov/news-issuances/news-releases/2020/nr-occ-2020-101.html

- Public Health England, "PHE statement on delayed reporting of COVID-19 cases", 4 Oct 2020, updated 5 Oct 2020 https://www.gov.uk/government/news/phe-statement-on-delayed-reporting-of-covid-19-cases
- Atlassian, "Post-Incident Review on the Atlassian April 2022 outage", 29 Apr 2022 https://www.atlassian.com/blog/atlassian-engineering/post-incident-review-april-2022-outage
- Data Protection Commission, "Data Protection Commission announces conclusion of inquiry into Meta Ireland" https://www.dataprotection.ie/en/news-media/press-releases/Data-Protection-Commission-announces-conclusion-of-inquiry-into-Meta-Ireland
- Mandiant, "UNC5537 Targets Snowflake Customer Instances for Data Theft and Extortion", Google Cloud Threat Intelligence blog, 10 Jun 2024 https://cloud.google.com/blog/topics/threat-intelligence/unc5537-snowflake-data-theft-extortion
- Google Cloud, "Sharing details on a recent incident impacting one of our customers", 24 May 2024 https://cloud.google.com/blog/products/infrastructure/details-of-google-cloud-gcve-incident

Appendix D. Glossary

- Apache Iceberg Table Spec, Snapshot References and Snapshot Retention Policy https://iceberg.apache.org/spec/
- Kulkarni and Michels, "Temporal features in SQL:2011", SIGMOD Record 41(3), 2012, DOI 10.1145/2380776.2380786 https://doi.org/10.1145/2380776.2380786

THE SERIES

Every Record Counts

Every Record Counts is a series about data: where each record was born, what it means, who can change it, what it costs and who reads it now, from the tally stick to the training set. This is its third book. Each book stands on its own.

Also by the author:

Every Record Counts	**From Tally to Token**: the record, its readers, and what each era broke
Every Record Counts	**The Model as Reader**: data for training, retrieval and agents
Every Record Counts	**Data on Every Desk**: files, spreadsheets and the PC era
Every Record Counts	**What a Value Means**: types, encodings and formats
Every Pixel Matters	**From Bit to Pixel**: the whole stack, from the bit to the pixel
Every Pixel Matters	**From Code to Glass**: the rendering layer
Every Pixel Matters	**From Silicon to Screen**: who controls each layer, and what trust costs
Medallion	**From Bronze to Gold**: why data layers decide what AI can do

Juan Cardena

Enterprise AI & Agentic Systems Architect · blog.jcardena.com

Download the latest edition and the templates at blog.jcardena.com/en/ebooks/from-warehouse-to-lakehouse.

Every Record Counts · Book 3 · From warehouse to lakehouse.