

Where Correctness Lives

A working engineer’s history of schema — lost, rebuilt, and still missing a floor.

Michael P. Laing

`mllaing@post.harvard.edu`

ORCID 0009-0004-5593-5759

Written with Anthropic Claude Fable 5.1

August 2026; revised September 28, 2026

Companion essay to “Measure-Relational Algebra: The Core” (September 2026; doi:10.5281/zenodo.22999911). Canonical reference: realql.org.

This essay is partly a history and partly a confession. I spent a career inside the systems it describes — first in the relational discipline, then, enthusiastically, in the movement that dismantled it. A practitioner’s view is always myopic: you work all day inside one organization’s constraints, and even with unusual license to experiment — which I had, and used — you see the elephant mostly from where you’re standing. So take this as one engineer’s reconstruction, offered with dates and reasons, and correct it where your part of the elephant differed.

What Codd actually moved

The relational model is remembered as tables, keys, and joins. That memory undersells it. Codd’s deep move, in 1970, was jurisdictional: he relocated correctness. Before the relational model, data integrity lived in application code — every program that touched a file was responsible for not corrupting it, and the record of what the data *meant* lived in the heads and habits of the programmers. Codd declared that meaning and integrity were the *system’s* job. Types, domains, keys, constraints, dependencies: declared once, at the boundary, enforced for every reader and writer, forever. Normalization theory went further — it made whole classes of error *unrepresentable*. You cannot commit the update anomaly, because the schema forbids its existence.

Everything people later found bureaucratic about the SQL world — the DBAs, the change-control boards, the migration reviews — was the institutional shadow of that single idea: **the data model is where correctness lives, so changing it is a serious act.** Around this a genuine professionalism accreted through the 1980s and 1990s: composability of queries, transactional guarantees, a shared algebra that meant a question asked of the data had one defensible answer. It was slow, and it was often infuriating, and it was a floor under everything.

The trade, and why the cool kids were right

Around 2008–2012 that floor was deliberately taken up. The Dynamo paper (2007) and BigTable paper (2006) described what Amazon and Google had built to survive their own scale; Cassandra, MongoDB, Riak, and DynamoDB productized the ideas; Brewer’s CAP theorem supplied the intellectual cover. The trade was explicit and, at the time, honest: give up global consistency, joins, and enforced schema; receive horizontal scale, always-on availability, cloud-native operation, and — the underrated term — *developer velocity*. Schema-on-write became schema-on-read: store whatever arrived, decide what it means when you look at it.

I want to be precise about the sociology, because I lived it. The people who moved were not cowboys rejecting rigor; they were good engineers making a rational trade under real constraints. The web’s workloads genuinely did not fit a single-node ACID database. Product cycles genuinely were being lost to migration review. The new systems were fun, fast, and *good enough*, and “good enough, this quarter” is how working engineers are actually paid to think. I became an expert in Cassandra and DynamoDB myself, and I’d make the same call again under the same constraints. The myopia was not stupidity; it was the ordinary condition of practitioners, who see the system in front of them and not the four-decade arc.

But notice what the trade actually discarded. Not tables — those came back immediately, as “wide columns” and “documents.” What was discarded was the jurisdiction: correctness moved *out of the data model and back into every application that touches the data*. Schema-on-read means every reader re-derives, or guesses, what the bytes mean. Twenty services touch the data; twenty implementations of its meaning drift apart. The relational floor was replaced by vigilance — each team’s discipline, each on-call engineer’s memory — which is to say, by the pre-1970 arrangement with better tooling. Meanwhile data volume exploded, pipelines lengthened into chains of services and streams and ML feature stores, and *similarity* quietly joined equality as a basis for connecting records: fuzzy matching, embeddings, entity resolution — operations with no place at all in the old algebra, performed with no declared semantics in the new stores.

The pendulum came back — for syntax

Here is where an honest history must push back on its own nostalgia, because the story since roughly 2015 is substantially a *re*-schematization, and any database researcher will say so. NewSQL systems (Spanner, CockroachDB) restored transactions at scale. The lakehouse movement put enforced tables back over object storage. Schema registries now police the shape of every message on a stream. dbt turned transformation into declared, versioned, tested models; “data contracts” — the current enthusiasm — are producer-consumer schema agreements with teeth; validation frameworks assert distributions and ranges in CI. The industry, having relocated correctness into application vigilance and discovered at scale what that costs, has spent a decade hauling it back into structure. Data contracts are normalization’s ghost, and their popularity is the market conceding Codd’s point.

So “schema faded” overstates the case — for *shape*. Types, ranges, nullability, referential structure: rebuilt, or being rebuilt, tool by tool.

But look at what all of it validates. A field is a float. The float is between 0 and 500. It is never null. It arrived from *this* upstream table, transformed by *that* job — lineage tools will

draw you the graph. Nowhere in the modern stack — not the contract, not the registry, not the lakehouse table, not the lineage graph — is there a slot for what the number *knows*: that it is a measurement, made by an instrument, with an error model; that it shares a calibration with ten thousand siblings; that it has already been averaged, or thresholded, or collapsed, and by whom. Lineage tracks where a value came from. Nothing tracks its epistemic status — “epistemic” being the philosopher’s word for *what is actually known*: not the number, but the state of knowledge behind the number, and how good it is. The active, ingestion-to-use schema was rebuilt for syntax and never existed for uncertainty — not in the SQL golden age either. This is the sense in which the disorder is real and even understated by its critics: the industry did not lose schema for what a value knows. It has never had one.

Why the floor matters more now

Three things make the missing floor a current problem rather than a historical grievance.

The chains got long. A number now routinely passes from sensor to stream to lake to feature store to model to dashboard to decision, crossing a dozen systems, each of which faithfully preserves its *value* and silently discards everything about its *standing*. Each hop is another chance to treat an estimate as a fact, and there is no statistician stationed between hops.

The operations got curved. The interesting steps in modern pipelines — thresholds, ratios, exponentials, learned nonlinear models, similarity itself — are precisely the operations for which computing on a collapsed value is not an approximation of the correct computation but a different computation, wrong with a sign. The accompanying paper proves where the boundary sits: collapse commutes with an operation, for every dependence structure among the inputs, exactly when the operation is affine — or the stored value was exact to begin with. Engineer’s translation: affine means a straight-line combination of its inputs, sums and scalings and weighted averages, $a*x + b$ and nothing that bends. Everything the modern stack finds interesting is on the far side of that line.

And the failure mode is now documented at civilization scale. In 2026, *Nature* published the demonstration that forty years of apparent saturation in Earth’s geomagnetic response to the solar wind is reproduced, quantitatively, by uncertain measurements stored as exact values and pushed through nonlinear analysis — no physical mechanism required, and no statistical evidence left that the saturation is real — an error that had roughly halved our estimate of extreme-storm risk, corrected not by better instruments but by finally carrying the uncertainty. Nobody in that story lacked discipline. The uncertainty was documented; the corrective statistics were a century old. What was missing was structural: the data model in which four decades of careful people worked had no place to put what the measurements knew, so it went nowhere.

The proposal, stated as history

Read this way, the proposal in the accompanying paper is not radical; it is the third verse of a song the industry already knows.

Verse one, 1970: correctness for *values* moves from application vigilance into the data model. Verse two, 2010: it moves back out, for good reasons, and the industry spends the next fifteen

years discovering the price and hauling it back. Verse three: extend the returning motion one level deeper. Let the schema declare not only what shape a value has but what *kind of knowledge* it is — a measure, with its dependence carried, with collapse as an explicit audited operation instead of a silent default at ingestion. The mathematics says exactly where the system may collapse early for free (the affine parts — which is why such a system need not be slow) and prices the error everywhere else. Correctness for *epistemic status* moves, at last, from the analyst’s vigilance into the structure — the only place vigilance has ever successfully been retired.

I watched the floor come up, and I helped. The view from inside was limited — it always is — but the arc is visible now. The industry has spent fifty years arguing with Codd about where correctness lives and has ended, each time, by agreeing with him. There is one floor left to lay.

The paper: M. P. Laing, “Measure-Relational Algebra: The Core,” *realql.org*, September 2026, <https://doi.org/10.5281/zenodo.22999911>. *The motivating case:* N. Sivadas et al., *Nature* **655**, 1143–1147 (2026). *Companion essays:* “The Cost of Writing Down a Number” and “The Half-Trace,” *realql.org*.