A Governed Geographic Intelligence Platform

Why trust in geographic data has to be built into the schema — not enforced by discipline — and how a four-layer architecture gets us there.

JEFF FRANZEN · JUNE 2026 · FOR DISCUSSION — INDEPENDENT RESEARCH

Most organizations do not lack geographic data. They lack a way to know, for any given number on a map, where it came from, how old it is, who owns it, and whether it is safe to put in front of an AI or an executive. This brief describes a platform that answers those questions by construction. The governing principle is one sentence: *nothing enters the trusted system without source, date, owner, validity, currency, and use restrictions* — and here those rules are enforced by database constraints and foreign keys, not by a habit someone has to remember.

> *"Nothing is considered trusted merely because it loaded successfully."*
>
> — CORE RULE, BOTH AGENTS OBEY

The Problem Is Not Data. It Is Trust.

A spreadsheet that loaded without an error is not the same as a number you can defend in front of leadership. Most data systems conflate the two: if the import succeeded, the data is "in," and its provenance lives in someone's memory or a file name. That works until the person leaves, the file is overwritten, or current chapter boundaries get silently applied to a three-year-old disaster. The failure is quiet, and quiet failures are the dangerous kind.

The platform inverts the default. Data is untrusted until it carries its papers. A row cannot exist in a topic table unless it points to a registered source. A county FIPS code cannot be stored in a way that drops its leading zero. A second "current" chapter assignment for the same county is refused outright. Trust is not a review step at the end — it is the shape of the database.

Three Priorities, In Order

Every source is judged on three things, and the order is not negotiable: **accuracy** first, **currency** second, **speed** third. Speed matters — a slow answer helps no one in a disaster — but a fast wrong answer is worse than a slow right one. The architecture optimizes for trust and treats latency as a problem to solve afterward, with caching and serving views, never by cutting a corner on validation.

The Architecture: Four Layers, One Gate

Data moves in one direction, from least trusted to most governed, and only the final layer is ever exposed to an AI, an API, or the public.

1 RAW ARCHIVE
Original Files, Exactly As Received
Nothing is edited here. This is the audit trail — the ability to reproduce or dispute any later number by going back to what actually arrived.

2 CLEAN DATA
Standardized Parquet / GeoParquet
Typed, deduplicated, and documented. Column names normalized, geometries validated, but still file-based and easy to inspect locally with DuckDB.

3 DATABASE
Postgres + PostGIS — The Canonical Store
The single source of truth, with constraints that reject bad data at the door. This is where the geography spine, the topic tables, and the full governance catalog live.

4 AI / API / MAP · GATED
The Only Layer Anything Outside Ever Sees
Claude, Codex, dashboards, Vercel apps, ArcGIS, and reports read from a serving layer of approved views — never from the raw or canonical tables underneath.

Why The Spine Is Two Tables

The geography spine is the master join — it connects every county to its chapter, region, and division. The obvious design is one table holding the polygon, the assignment, and the dates together. We split it, and the reason is a real operational fact: **a county's shape almost never changes, but its chapter assignment does** — every boundary realignment moves counties between chapters.

So the polygon lives once in `county_boundary`, and the assignment lives in `geography_assignment` as slowly-changing history with effective dates. The manual's single spine is rebuilt as a view over both — same answer, but a realignment no longer duplicates the full polygon, and a stored "current" flag can no longer contradict the dates because there isn't one.

The payoff: historical correctness by construction.

FUNCTION	`geography_at(fips, date)` returns the assignment that was valid on a given date.
EFFECT	A 2023 disaster joins to the chapter that owned the county *then* — not the one that owns it today.
VERIFIED	Proven on live Postgres: the 2023 event resolves to the old chapter, current to the new.

The Governance Catalog

Four schemas separate data by trust level. The catalog is not paperwork bolted on — it is what makes the AI gate real.

SCHEMA	HOLDS	PURPOSE
gov	source_registry, field_catalog, ingestion_log, validation_results, dataset_inventory	Provenance, glossary, run history, test results.
geo	county_boundary, geography_assignment, tract_county_crosswalk, the spine view	The master join and its temporal history.
topic	fema_mobile_homes_county, and future subject tables	Facts joined by FIPS / GEOID — never with geography copied in.
serving	Approved, AI-safe views only	The single surface the AI / API role may read.

The AI Gate, Made Real

Every field carries an `approved_for_ai` flag. On its own, a flag is a label — and a label is not security. It becomes a gate the moment a dedicated `ai_reader` role is granted read access to the serving schema *only* and revoked from everything else. Then the AI literally cannot reach an unapproved field, because the database will not let it. That role is the next thing to wire, and it is what turns a governance promise into an enforced boundary.

Three Roles That Keep Each Other Honest

Claude Code builds — schema, ingestion, tests, documentation. **Codex challenges** — an adversarial reviewer hunting bad joins, stale data, exposure risks, and silent failure modes, reviewing only a committed schema against a real load. **The gatekeeper approves** — source authority, sensitivity, public versus internal use, and final publishing. No single agent can both build and bless its own work.

Where This Stands, And The Ask

The scaffold is built and proven against a live database: every schema applies clean, every guard rejects bad data, and the historical join resolves correctly. What remains before the first real load are three gatekeeper decisions — pick one canonical cloud Postgres (recommended: Supabase), confirm PostGIS as the canonical geometry store, and authorize wiring the `ai_reader` role.

> *"We are not trying to collect everything. We are building a system that knows what data exists, where it came from, how current it is, how trustworthy it is, and whether AI can touch it."*
>
> — THE ONE-SENTENCE VERSION