Catastrophe — wrong-date fallback (music-time)

Key takeaways

1. A valid schema can still be false about the world; domain checks catch it.
2. Visualization is part of the data model — sometimes the plot changes it.
3. Rows are not things: write down what one row means before aggregating.

music-time

Rows were valid; meanings were wrong.

59	9	9,027	292
commits	**active days**	**Python LOC**	**tests**
mostly Claude Code	2026-02-19 -> 2026-05-06	42 files	regex-heavy normalization

Scrapes Wikipedia, Archive.org, MusicBrainz, LivePhish, and phish.in; renders 30+ years of jam length as space-filling-curve plots.

Dated case study — repository state as of 2026-05-06.

Scrape once; reinterpret often

The architecture is a protection against re-fetching the world.

Fetch and cache raw source pages.

Wikipedia / Archive.org / MusicBrainz

Parse cache → SQLite; normalize + dedup.

One row must mean one performance.

Render sunflower + streamgraph views.

Plots expose semantic mistakes.

Decoupling matters: schema changes should not trigger ~18,000 network fetches.

Repo by the numbers — code volume

Component	LOC	Notes
gdtimings/ (Dead scrapers + analyzer)	3,409	Wikipedia + Archive.org + MusicBrainz
phishtimings/ (Phish)	1,550	LivePhish.com + phish.in
viz/ (rendering)	1,803	Sunflower / streamgraph plots
tests/	2,265	292 test cases, all passing
Total Python	9,027	across 42 files

- Test ratio ≈ 25% — heavy regex normalization needs heavy regression coverage.

Repo by the numbers — time + cost

- Build time
  - 59 commits, 2026-02-19 → 2026-05-06
  - 9 distinct active development days (burst pattern, not daily)
- Sessions
  - 36 Claude Code session directories under ~/.claude/projects/…
  - Almost all sessions co-authored by Claude Opus 4.6 (1M context) — one late commit on Opus 4.7
- Cost caveat — and itself a lesson
  - ccusage reports $0: music-time used the older nested-jsonl session format that ccusage does not read
  - Unmeasured ≠ small; if you want to measure spend later, instrument it from day 1

Catastrophe — wrong-date fallback

Task	Record how long each song was played at the concert where it happened.	**The key field is concert date, not release date.**
Fault	MusicBrainz discs named only “Disc 1” / “Disc 2” had no show date, so the fallback used `release.date`.	**A valid date answered the wrong question.**
Signal	A 1970 concert reissued in 1996 moved into 1996; 127 of 151 fallback rows were wrong.	**Domain checks found what schema checks could not.**

A schema can be internally valid and still false about the world.

source: music-time/HISTORY.md @ 2026-02-22

Lesson — schema validity ≠ real-world validity

- How the bug was finally caught
  - A domain check flagged 58 “Playing in the Band” performances dated 1997–2026
  - Impossible: Jerry Garcia (the band’s lead guitarist) died in 1995; the Dead stopped touring then
  - The check was a domain constraint, not a type or schema constraint
- Why the fallback looked reasonable
  - `release.date` IS a real date — just answering a different question (“when did the CD ship?”)
  - Without domain context, “use any date when one is missing” looked like sensible code
  - Type checkers verify a field is a date; they cannot verify it’s the RIGHT date
- Lesson — bake domain constraints into the test suite
  - When an agent proposes a “reasonable default,” ask: under what REAL condition is this default wrong?
  - Add the constraint as a test: “no Dead performances after 1995”
  - One assertion catches the entire class of bug; the analyst’s mental constraint catches none

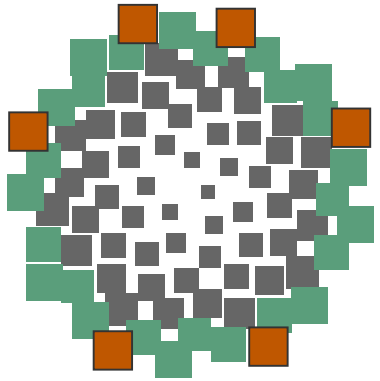
Story #1 — fractal intuition gets tested

Layout	30 years of concerts become one page-sized “sunflower”: each song is a tile on a golden-angle spiral, shortest jams at the center, longest on the rim; tile area ∝ song length.	**Position encodes duration rank, not the year.**
Question	Each tile is drawn as a tiny Gosper-curve fractal. Which way should it be rotated — random, fixed 0°, or matched to the curve’s hexagonal symmetry?	**An open rendering choice with no obvious answer.**
Proposal	The agent argued: snap every tile to 60° increments so they interlock cleanly, like a honeycomb.	**Plausible — the Gosper region looks hexagonal.**

source: music-time/HISTORY.md @ 2026-02-23 (tile alignment experiment)

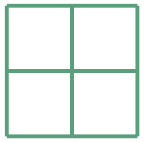
Lesson — cheap A/B tests beat clever math

golden-angle sunflower

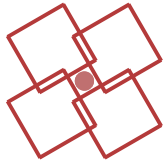


center = shortest jams · rim = longest

aligned 0°



hex 60°



Four strategies A/B-tested. Aligned (0°) won — hex rotation produced MORE collisions, not fewer.

- Why the intuition failed
  - The Gosper region is only roughly hexagonal — radius varies 0.18–0.77 by sector
  - Golden angle ≈ 137.5° is irrational, so adjacent tiles never share an orientation
- Lesson — ask for the A/B test FIRST
  - An optimization from a mathematical property? Verify the property empirically first
  - Two minutes of brute-force comparison beats two hours of “of course it works”

Story #2 — when the plot drives the schema

Layout	Each performance is a square tile; longer song → bigger square. The first scheme used 4 quartile bins: short, medium, long, and “epic” (top 25%).	**Statistically defensible binning.**
Fault	A 47-minute “Playing in the Band” (1972-05-21) landed in the same “epic” bin as ordinary 15-minute jams — 2.2× the next-longest tile, but drawn no larger.	**The legendary outlier vanished into the crowd.**
Signal	Nothing in the data was wrong. A human looked at the plot and said, “that can’t be right.”	**Only the rendering exposed the problem.**

source: music-time/HISTORY.md @ 2026-02-23; commit 3367c0d

Lesson — visualization is part of the schema

- The fix
  - Added a 5th bin: “Gigantous” (≥25 min)
  - Tiles in this bin are 3.5× base size and rendered in distinctive white
  - Power-law sizing compresses bins below so the outlier doesn’t blow up the page layout
- What the agent did, and didn’t, propose
  - The agent added the new enum value, threshold, and visual treatment in one commit — when asked
  - It did NOT propose the new bin on its own; the trigger was a human noticing “the plot looks wrong”
  - The data was correct all along; only the rendering exposed the problem
- Lesson — design the plot alongside the schema
  - Visualization is not downstream of the schema — sometimes it CHANGES the schema
  - Treat plots as first-class consumers of your data model, not an afterthought
  - Schedule “look at the rendering” as a regular checkpoint alongside “look at the data”

Story #3 — when 8 tapes became 8 concerts

Task

Count how often each song was played across 30 years of Grateful Dead concerts.

One row should mean one performance.

Fault

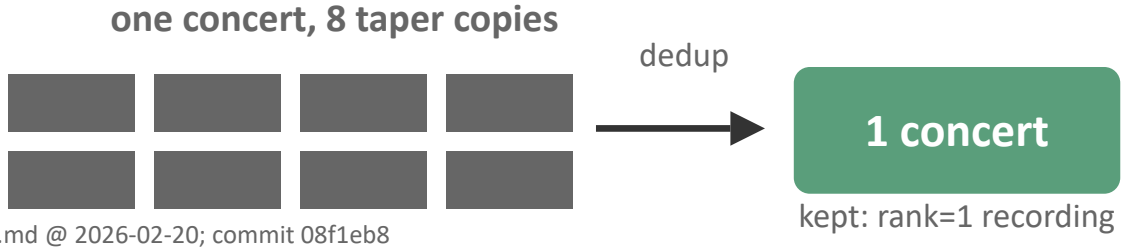
Archive.org stores every taper's recording as its own item. First-pass code counted each item as a performance — so a show with 8 tapers counted eight times.

The rows were tapes; the question wanted concerts.

Signal

Averages went noisy, year-over-year trends grew spurious spikes, and the most-taped shows dominated every statistic.

Row-by-row correct; wrong at the granularity boundary.



source: music-time/HISTORY.md @ 2026-02-20; commit 08f1eb8

Lesson — count THINGS, not rows

- The fix — pick one canonical recording per show
  - Quality rank: official release (500) > soundboard (300) > audience (100)
  - `ROW_NUMBER() OVER (PARTITION BY concert_date ORDER BY quality_rank DESC)`; keep rank=1
- Why this is hard to notice
  - Every row is a genuine recording of a real concert — nothing is fake, and no exception is raised
  - Only a sanity check (“why are there 8 Bertha performances on 1972-05-21?”) exposes it
- Lesson — multiplicity in source data multiplies your statistics
  - “How many rows?” and “how many THINGS?” are different questions; SQL won’t tell you which
  - Write down what one row represents in plain English, then check your queries agree

Where did the project stall?

- Wikipedia markup chaos (2026-02-19)
  - ~220 official-release Wikipedia pages each had idiosyncratic HTML
  - Cost: ~one day of reverse-engineering before the pattern crystallized
- Title normalization explosion (2026-02-22)
  - Song count dropped from 7,537 to 996 once cleaning was complete
  - tests/test_normalize.py grew to 92 cases — a cleanup tax that took longer than estimated
- What did NOT stall
  - Adding Phish took 5 days, not 5 weeks, because the architecture was designed to take a second band

Key takeaways — music-time

1. Domain-grounded sanity checks beat schema-grounded validation.
2. A cheap A/B test beats clever math: verify a plausible optimization empirically.
3. Visualization design is part of the data model — sometimes the plot changes the schema.
4. Rows are not things. Write down what one row means before trusting aggregate statistics.