One Engine, One Audit Trail

Traceable, SQL-first retrieval for compliance-critical AI systems

Martins Otun

Founder & Principal AI Engineer, Algonix AI Ltd

algonix.co.uk · linkedin.com/in/otun-martins · github.com/otunmartins

THE REGULATOR'S QUESTION

An AI approves a £3M loan. Six months later...

"Why?"

- Which documents were used?
- Which embedding model?
- Which chunk? Which version?
- Who uploaded the document?
- What was the similarity score?
- Could it be reproduced today?

Many AI systems cannot answer these.

Most vector databases were built for search. Search optimises for relevance. Regulated industries need something else — evidence.

THE GAP

Relevance is not evidence

What demos optimise

Relevance. Return the most similar text, fast. Looks impressive in a demo. Says nothing about where the answer came from, who could see it, or whether it can be reproduced.

What regulators require

Evidence. Which document, version, chunk, model, permission, and score produced this answer — and can you reproduce it? Reproducibility, explainability, defensibility.

IN DEPTH

Relevance is not evidence

It helps to name the gap precisely, because it is the whole argument in one line.

A typical RAG demo optimises for relevance. Return the most semantically similar text, quickly, and feed it to the model. It looks impressive. It also says nothing about where the answer came from, who was allowed to see the source, or whether the result can be regenerated next quarter.

A regulated system must optimise for evidence. Which document, which version, which chunk, which model, which permission, and which similarity score produced this answer — and can you reproduce it on demand? The properties that matter are reproducibility, explainability, and defensibility.

The good news: you don't have to choose. Keep the relevance and add the evidence — if the architecture cooperates.

THE CORE ARGUMENT

Your vector store should not be separate from your system of record.

Everything lives inside PostgreSQL — not just vectors, but:

- ✓ metadata
- ✓ permissions
- ✓ document versions
- ✓ audit logs
- ✓ retrieval history
- ✓ user actions
- ✓ citations
- ✓ compliance evidence

One engine. One audit trail.

THE PROBLEM

Why fragmented stacks fail audits

Where is the truth?

Which database owns retrieval? Each store has its own clock, log, retention, and permission model. Reconstructing one answer becomes a forensic exercise across systems that were never meant to align — at exactly the moment an inspection demands it be effortless.

IN DEPTH

Three consequences of splitting the stack

None of these is a bug you can patch — they are properties of splitting the stack.

Provenance is scattered

Reconstructing one answer means correlating records across systems whose clocks, identifiers, logs, and retention were never designed to align.

Permissions live in application code

Access rules get duplicated across services, drift out of sync, and are painfully hard to prove correct to an auditor.

Reproducibility is lost

Without a durable record of the exact chunk, embedding model, prompt version, and similarity, the same answer simply cannot be regenerated.

You discover all three at the worst possible moment — during an inspection.

“Retrieval is the easy part. Proof is the hard part.”

The gap between a good demo and a defensible system.

THE SHIFT

SQL-first: collapse it into one engine

One transaction

retrieval + evidence commit together

One consistency model

ACID, not eventual reconciliation

One backup

and one point-in-time recovery

One permission system

row-level security in the database

One audit trail

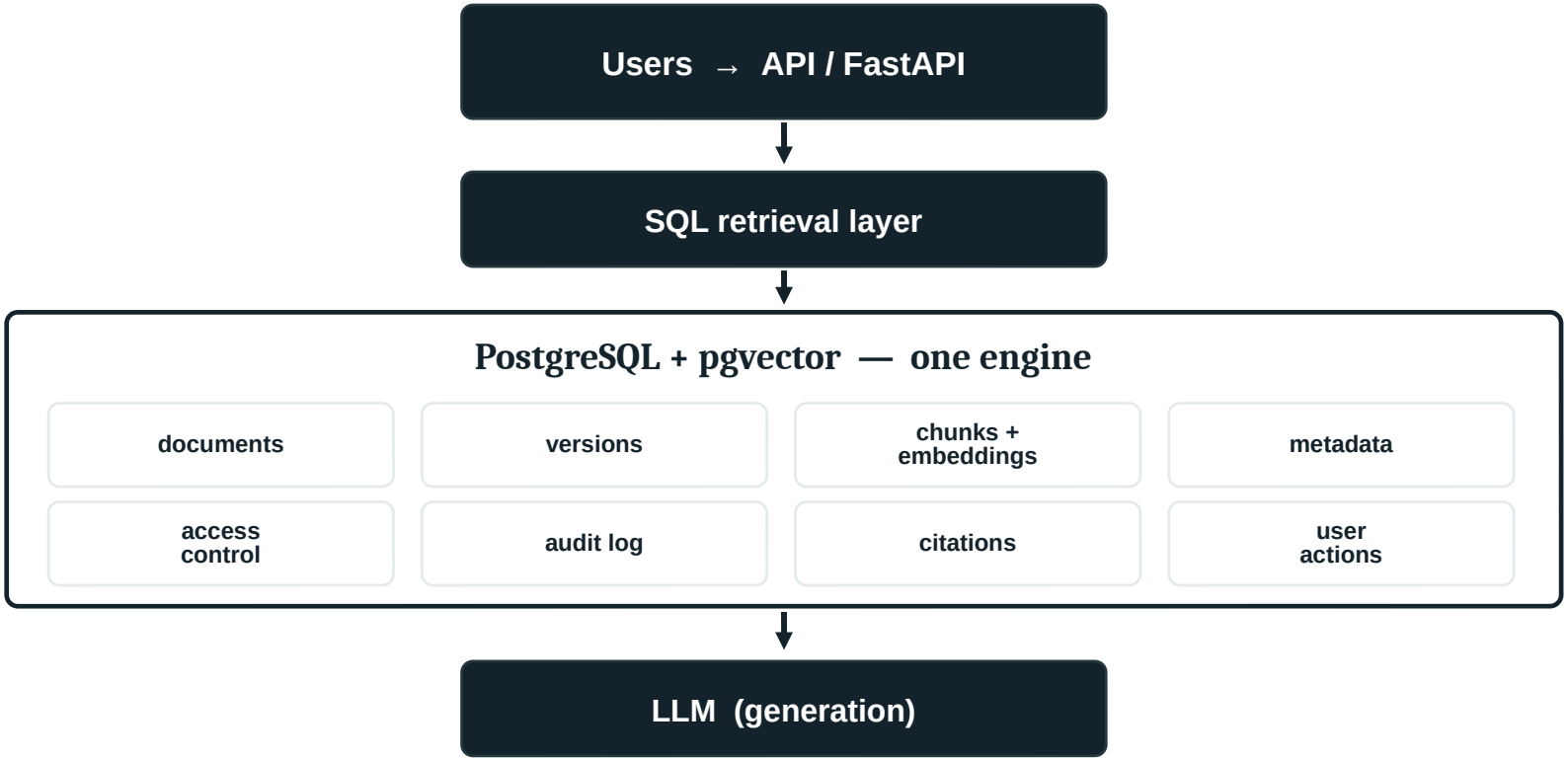
every retrieval, in one place

One source of truth

vectors beside the data they describe

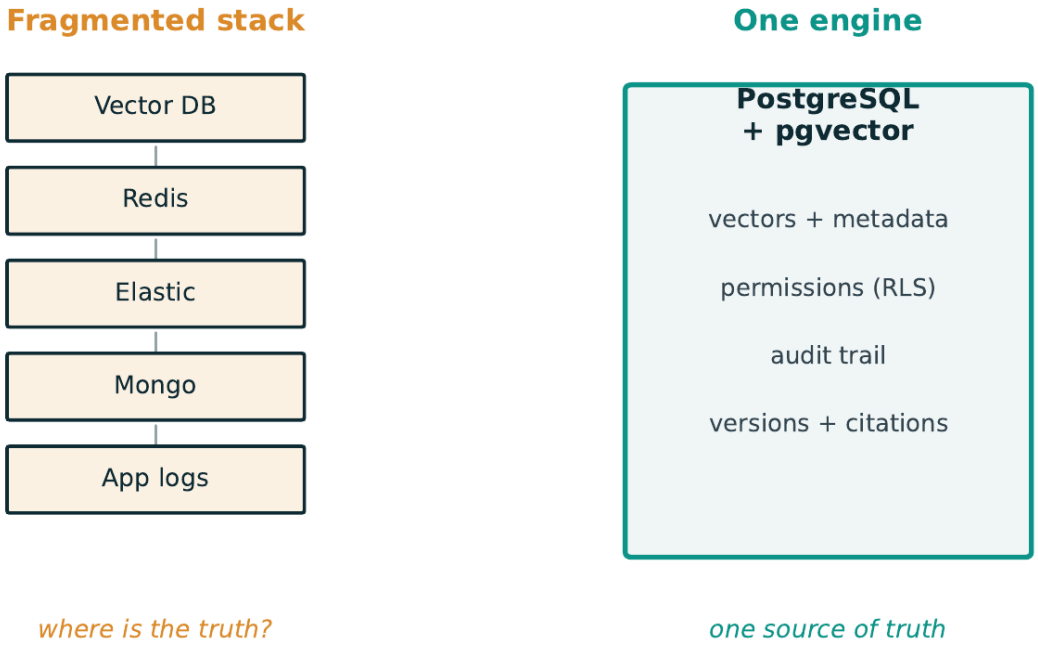
ARCHITECTURE

Everything inside one PostgreSQL engine



ILLUSTRATION

Where does the truth live?



A fragmented stack scatters truth across systems; one engine keeps vectors, permissions, versions, and the audit trail together.

THE OBVIOUS OBJECTION

Why pgvector, not a dedicated vector DB?

The honest answer for compliance-critical systems — not a verdict on Pinecone, Weaviate, or Milvus in general.

	Dedicated vector DB	PostgreSQL + pgvector
Data location	a separate managed service	**same engine as the records**
Transactions	no cross-store ACID	**retrieval + audit commit as one**
Joins with business data	stitched in application code	**native SQL joins**
Provenance & audit	reconcile two systems	**one append-only trail**
Ops surface	another system to secure & back up	**one system**
Where it wins	massive or globally-replicated scale	**co-location, compliance, joins**

Not a knock on dedicated stores. *At extreme scale or global low-latency they're the right tool. For evidence-first systems, co-location is the feature.*

BEFORE / AFTER

Retrieval: text vs. evidence

BEFORE — RELEVANCE

```
# typical RAG — relevance only
docs = vdb.similarity_search(q)
ctx  = "\n".join(d.text
                 for d in docs)
answer = llm.invoke(ctx)

# which chunk? which version?
# who uploaded it? score?
# -> no answer
```

AFTER — EVIDENCE

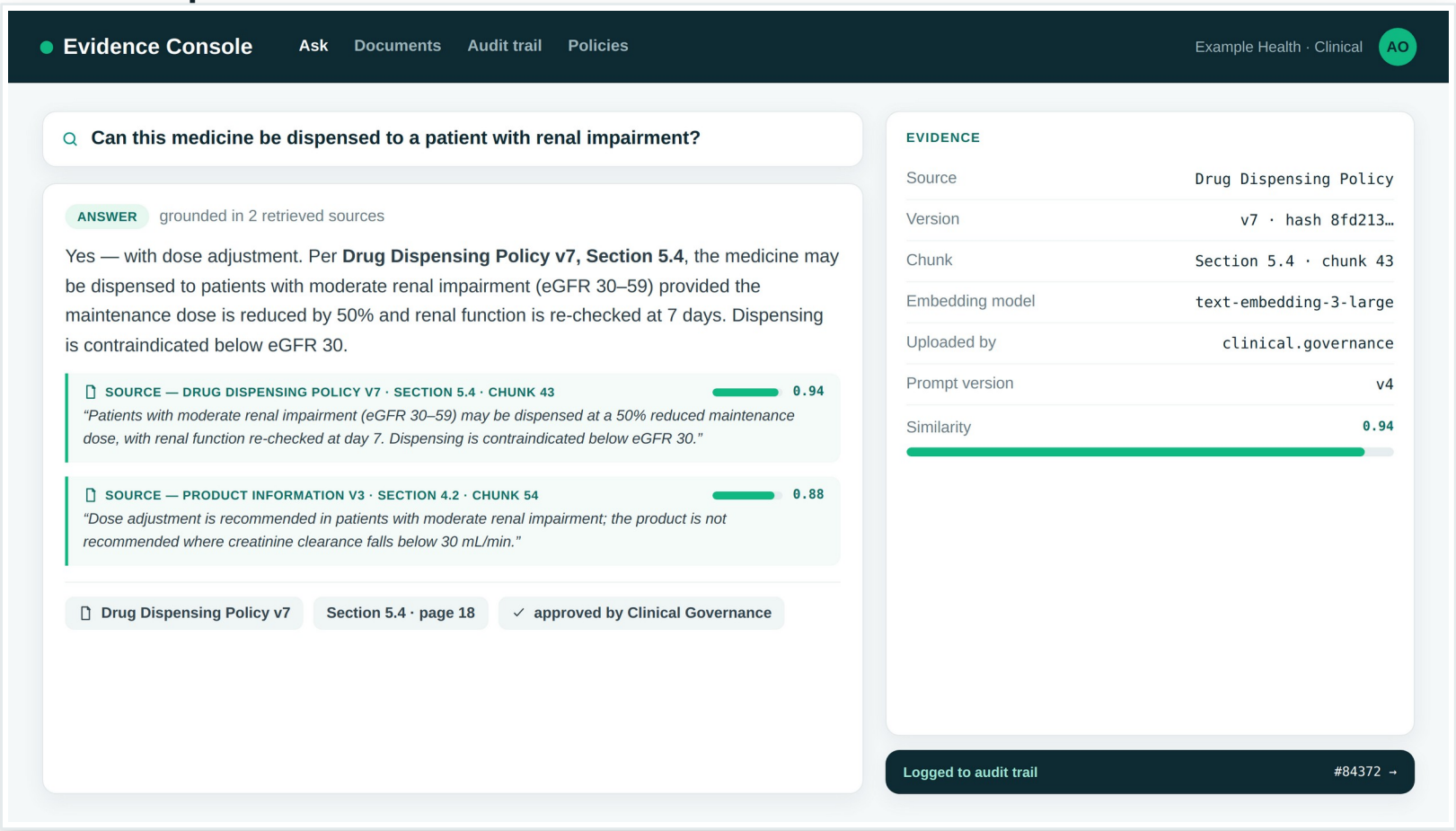
```
-- every field is evidence
SELECT c.id AS chunk_id,
       dv.version,
       1 - (c.embedding <=> :qvec) AS sim,
       d.owner AS uploaded_by,
       dv.approved_by
FROM chunks c
JOIN document_versions dv
  ON dv.id = c.doc_version_id
JOIN documents d
  ON d.id = dv.document_id
ORDER BY c.embedding <=> :qvec
LIMIT 5;
```

✕ Looks impressive. Leaves nothing to audit.

✓ Every column an auditor can defend.

EXAMPLE · HEALTHCARE · RETRIEVAL WITH EVIDENCE

The answer ships with its evidence



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

DATA MODEL

The schema is your audit trail

documents

id · title · owner
classification

→

document_versions

version · content_hash
approved_by · at

→

chunks

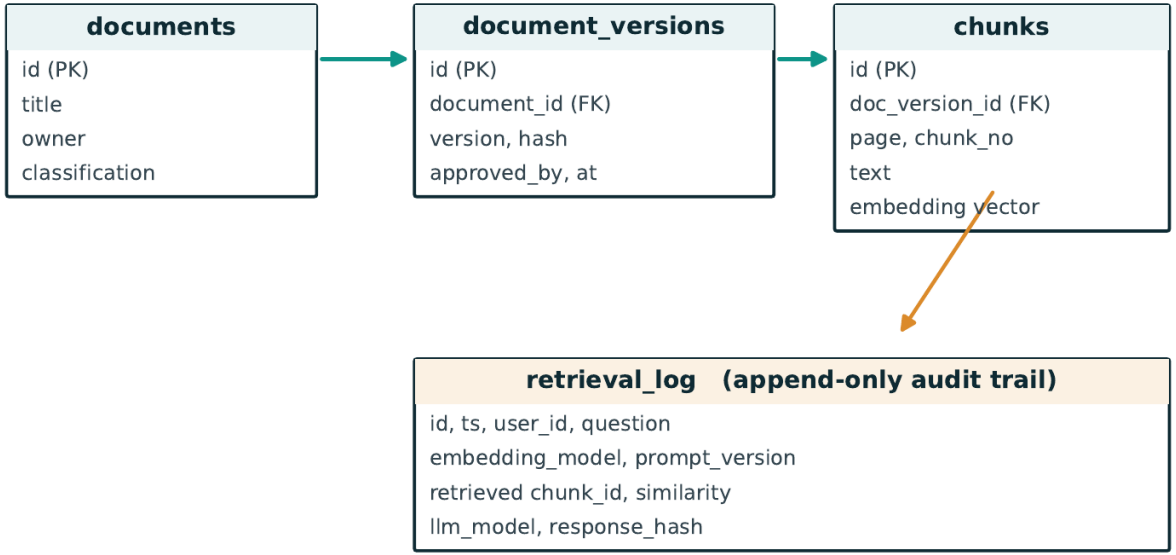
doc_version_id · page
text · embedding

retrieval_log (append-only)

ts · user_id · question · embedding_model · prompt_version · retrieved_chunk · similarity · llm_model · response_hash

ILLUSTRATION

The evidence data model



documents → versions → chunks is a normal hierarchy. retrieval_log is append-only — and becomes the audit trail.

EVIDENCE BY CONSTRUCTION

Retrieval and audit, one transaction

log.sql

```
-- retrieval and its evidence commit together
BEGIN;
  INSERT INTO retrieval_log
    (user_id, question, retrieved_chunk,
     similarity, embedding_model,
     prompt_version, llm_model, response_hash)
  SELECT current_user, :q, chunk_id, similarity,
         :embed, :prompt, :llm, :hash
  FROM retrieve_chunks(:qvec, current_user);
COMMIT;  -- one transaction, one record
```

Atomic

Evidence and answer commit together — or neither does.

Scoped

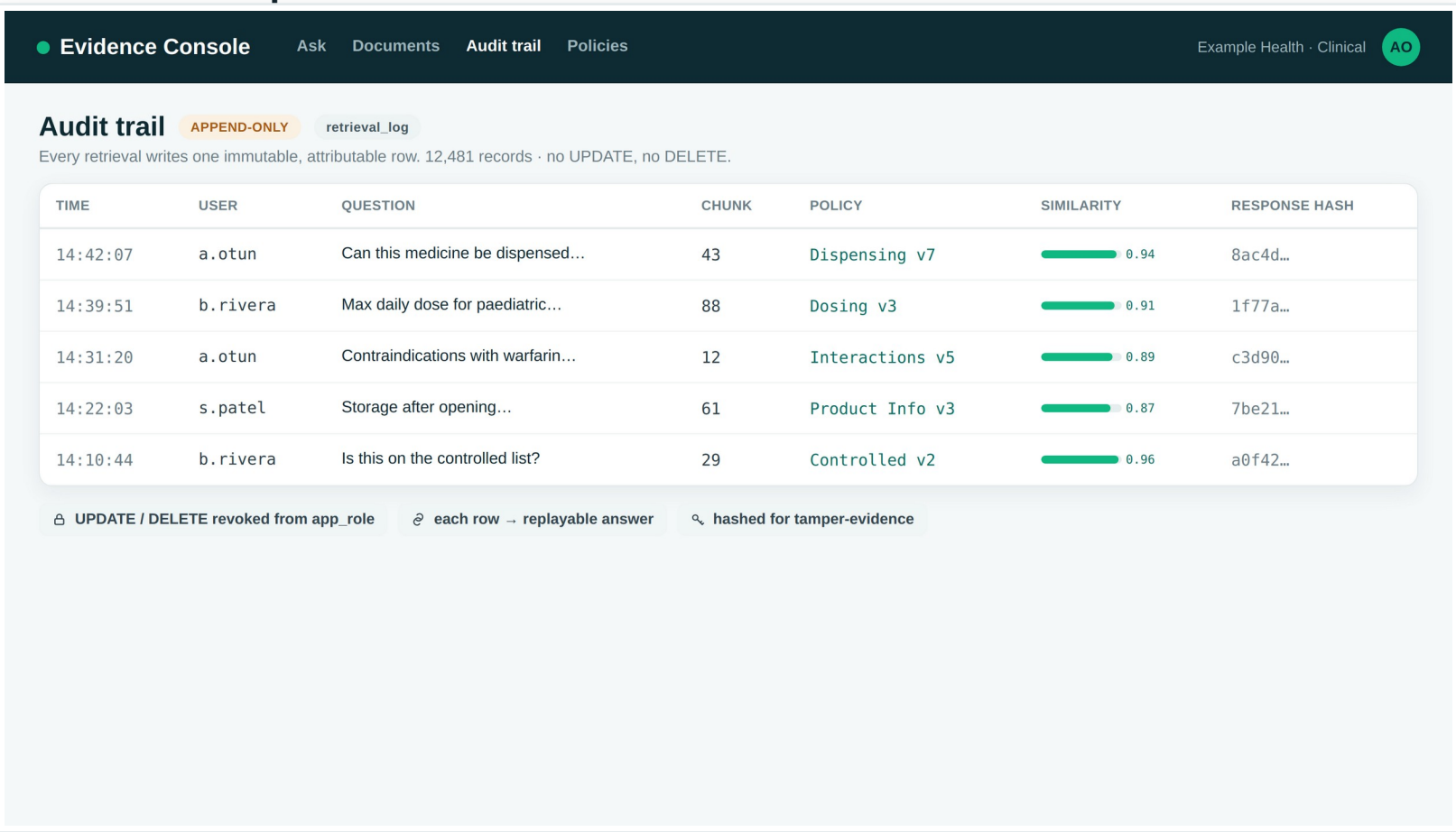
retrieve_chunks runs under the user's permissions.

Permanent

Nothing is retrieved without leaving a record.

EXAMPLE · HEALTHCARE · AUDIT TRAIL

One immutable row per retrieval



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

REPRODUCIBILITY

Six months later, reconstruct the answer

replay.sql

```
-- one lookup, one trail
SELECT *
FROM retrieval_log
WHERE id = 84372;
```

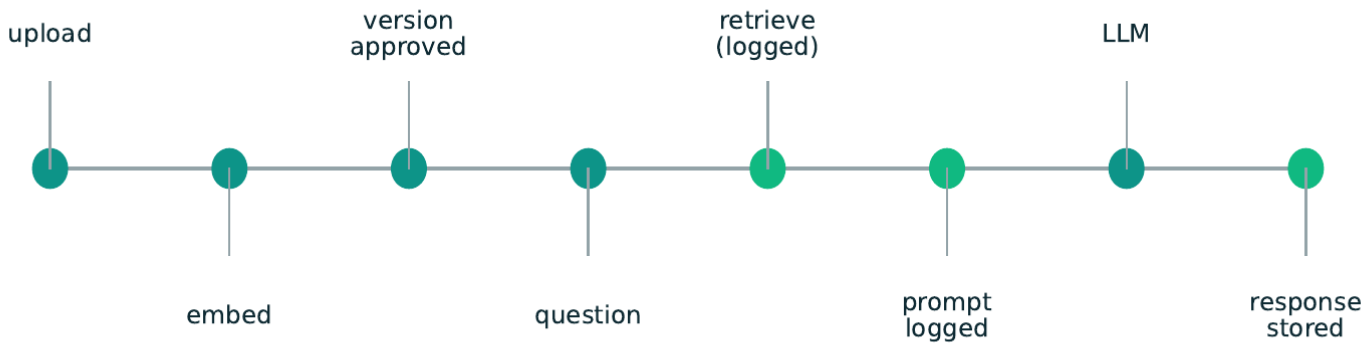
Reproducible: the exact query and embedding replay the same retrieval; the response hash confirms the served answer.

Illustrative audit record

Question	Can this medicine be dispensed?
Retrieved	Drug Policy v7 · §5.4
Similarity	0.94
Embedding	text-embedding-3-large
Prompt	v4
Response hash	8ac4d…

ILLUSTRATION

Each step writes an auditable row

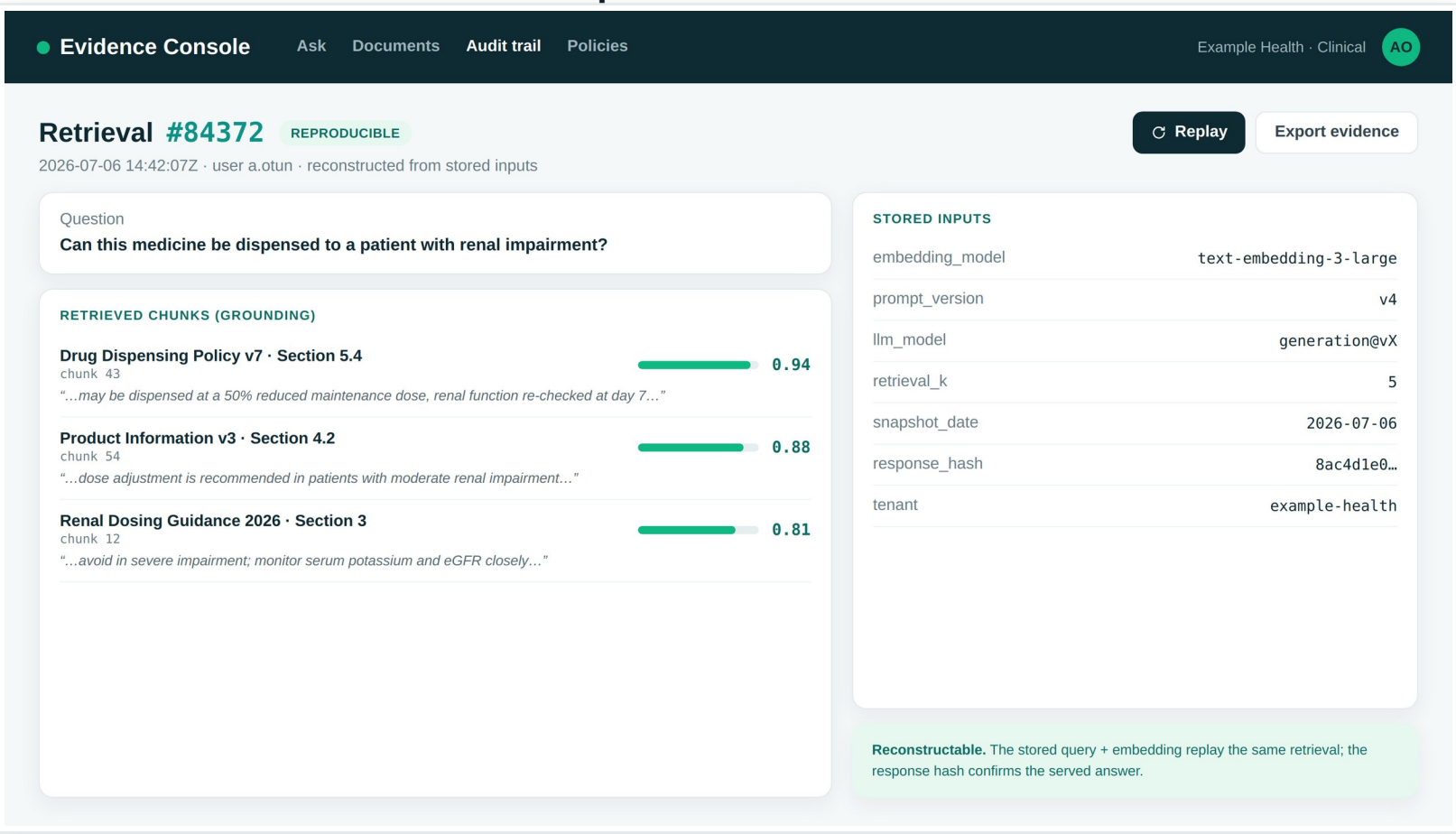


every step writes one attributable, time-stamped row — the answer is reconstructable months later

Upload → embed → approve → retrieve → prompt → generate → store. The answer is reconstructable months later.

EXAMPLE · HEALTHCARE · REPLAY

Reconstruct an answer from its stored inputs



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

“An answer you can’t reconstruct is just a confident guess.”

Reproducibility is not a feature — it is the requirement.

ROW-LEVEL SECURITY

Permissions live in the database

rls.sql

```
ALTER TABLE documents
  ENABLE ROW LEVEL SECURITY;

CREATE POLICY patient_access
  ON documents
  USING (owner = current_user);
```

Retrieval respects permissions

A clinician sees only permitted records — enforced by the database.

No Python filtering

The database enforces access, not fragile app code.

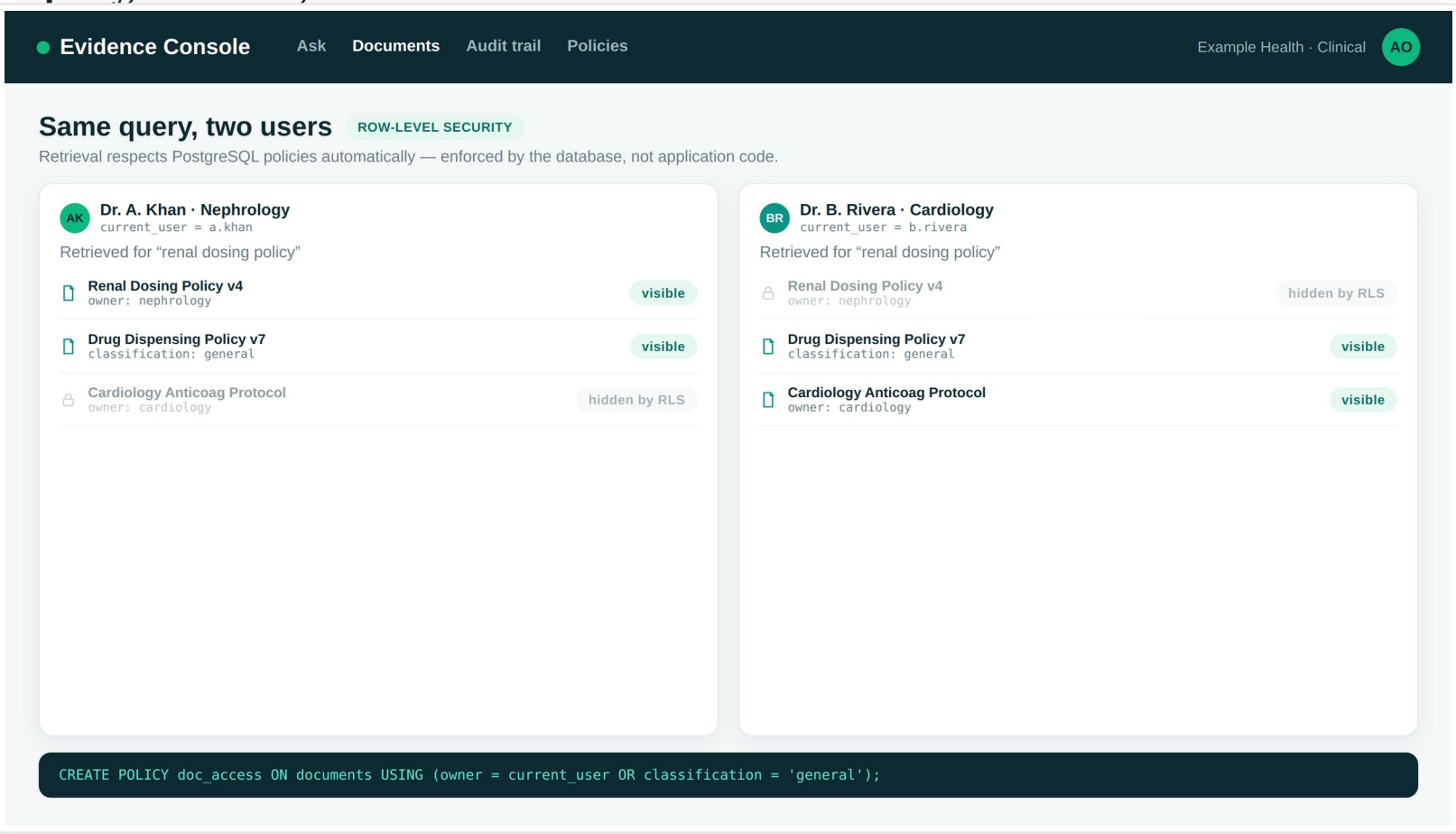
Fewer application bugs

One place to get right — and to evidence.

Note: table owners and superusers bypass RLS unless FORCE ROW LEVEL SECURITY is set. Run the application as a non-owner role.

EXAMPLE · HEALTHCARE · ACCESS CONTROL

Same query, two users, different evidence



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

ISOLATION

Multi-tenant retrieval, isolation enforced

tenant.sql

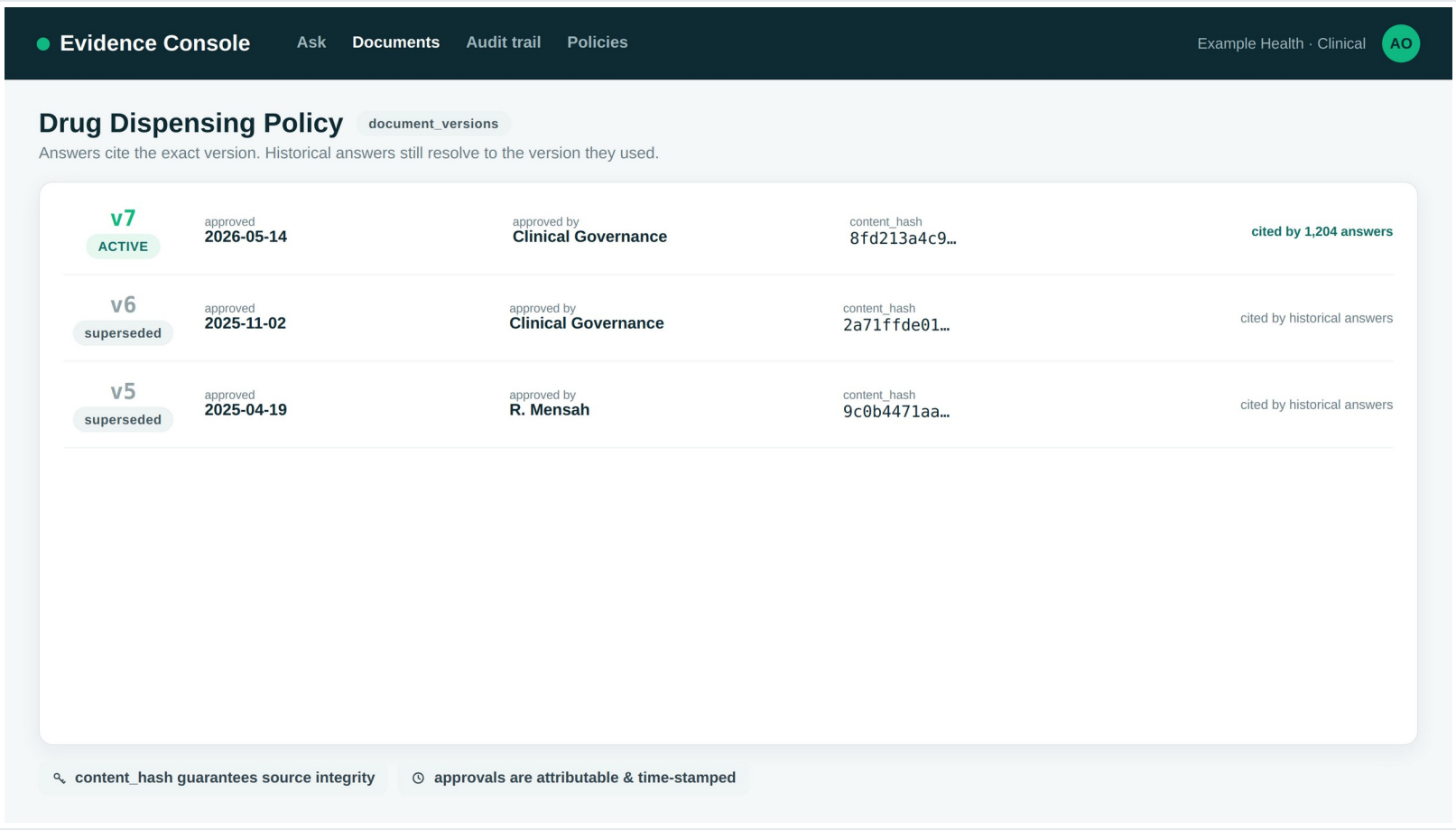
```
SELECT *
FROM chunks
WHERE tenant_id =
   current_setting('app.tenant')::uuid
ORDER BY embedding <=> :qvec
LIMIT 5;
```

Isolation the database enforces

Tenancy becomes a WHERE clause the database enforces — not a filter the application might forget. Semantic search stays scoped to the tenant.

EXAMPLE · HEALTHCARE · PROVENANCE

Answers cite the exact version



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

SAME ARCHITECTURE

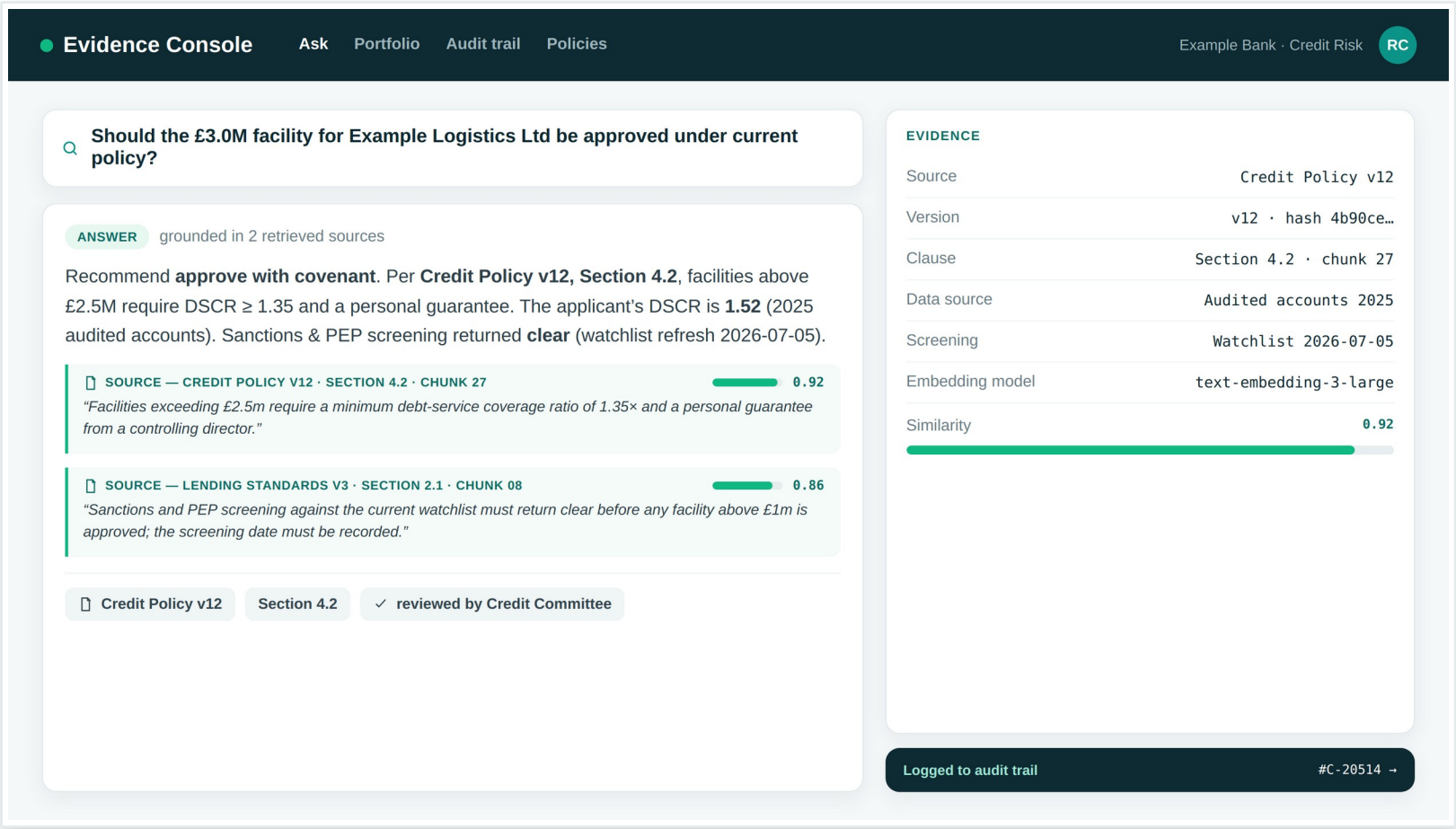
One engine, many regulated industries

The evidence, audit trail, and access control are the same in credit risk, financial crime, and claims. Only the documents change.

All company names, people, documents, figures, and data shown are hypothetical and created for illustration only. They do not represent, and are not derived from, the real data of any company, public or private. Any resemblance to an actual organisation or individual is coincidental.

EXAMPLE · FINTECH · CREDIT RISK

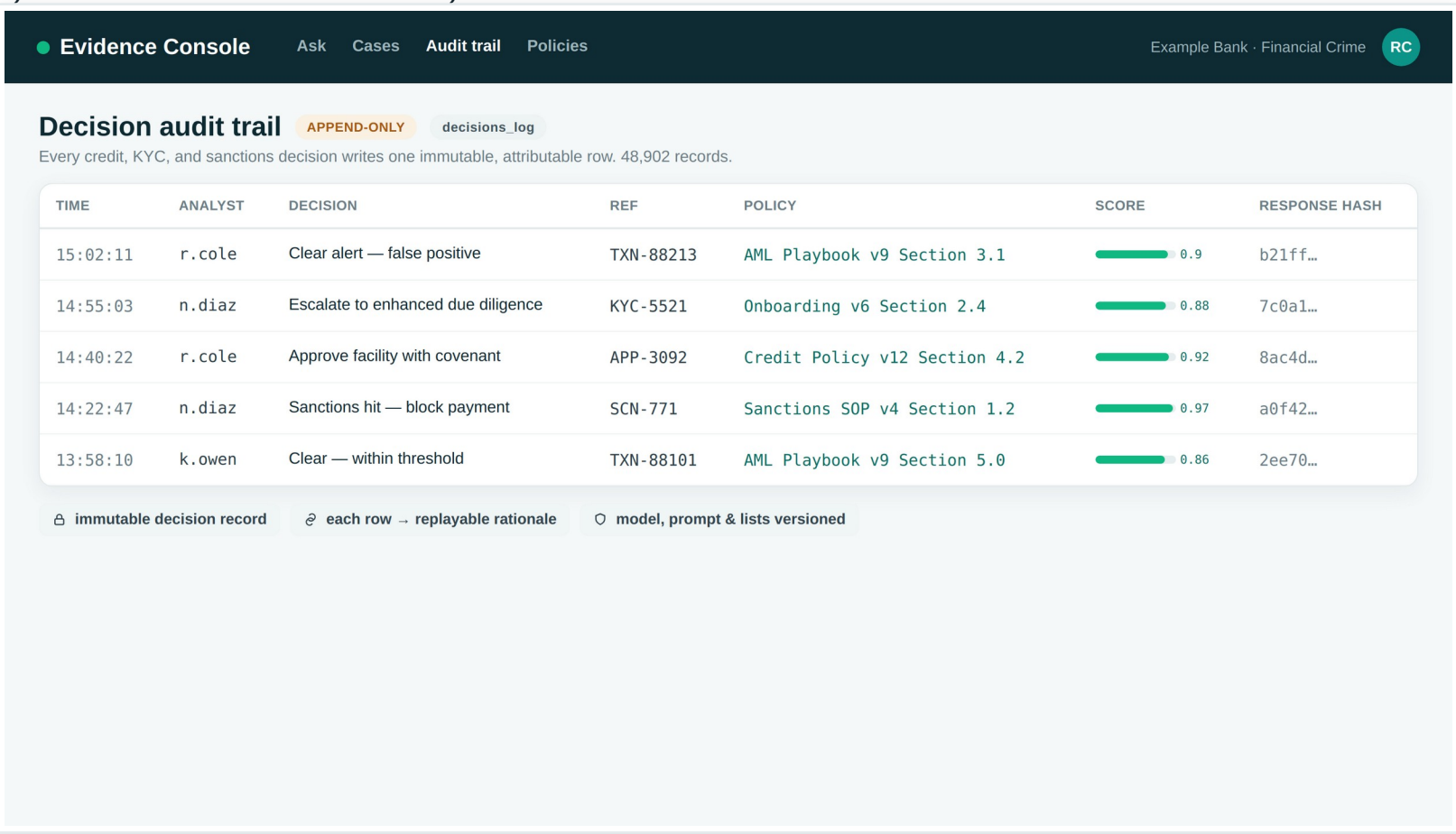
The £3M loan decision — with its evidence



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

EXAMPLE · FINTECH · FINANCIAL CRIME

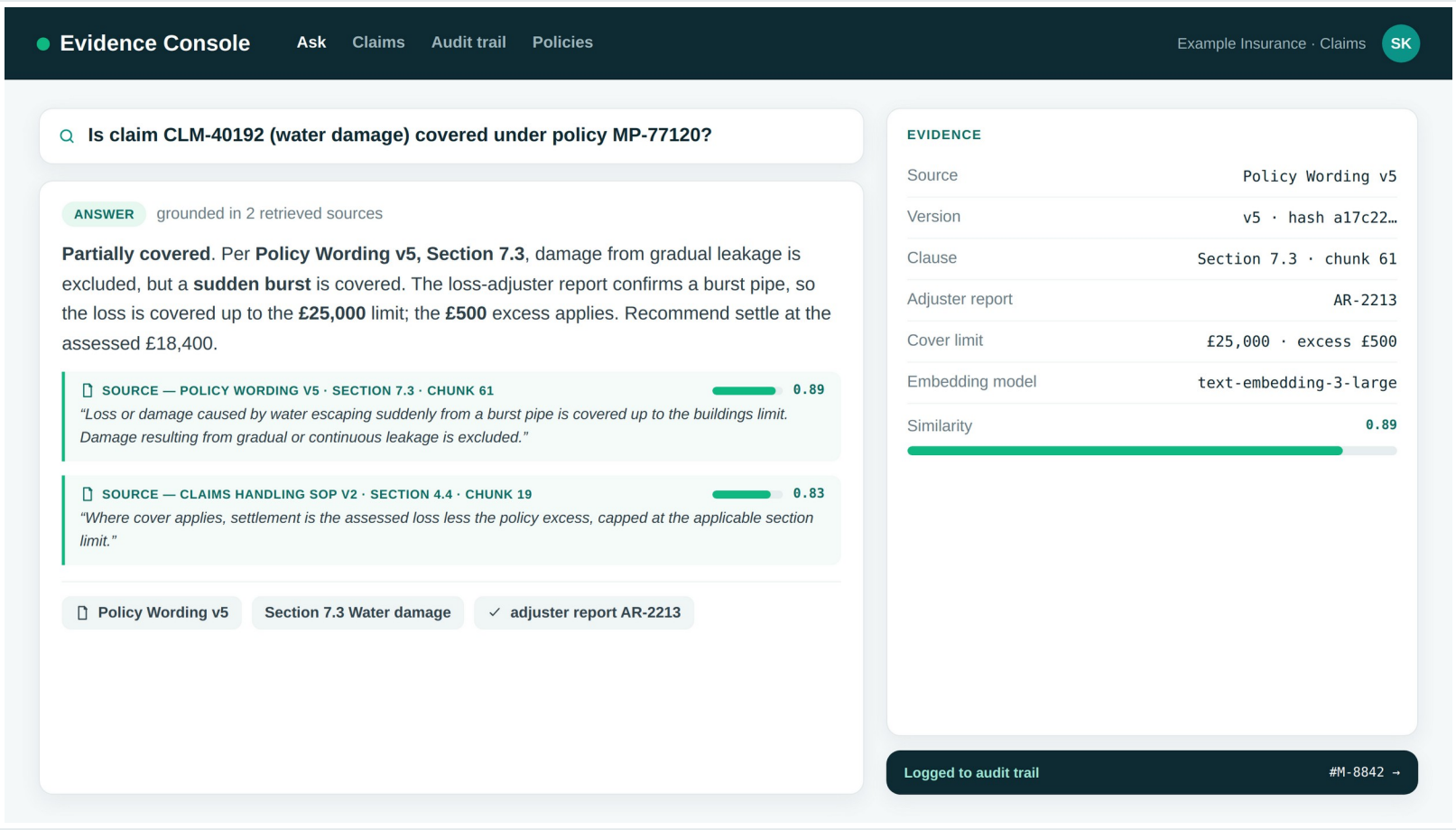
AML, KYC & sanctions decisions, audited



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

EXAMPLE · INSURANCE · CLAIMS

A claims decision you can defend



Hypothetical illustration — company names, documents, and data are invented and do not reflect the real data of any company, public or private.

DATA-INTEGRITY EXPECTATIONS · ALCOA+ [8, 9, 10]

How the data model supports each expectation

Attributable

`user_id · uploaded_by · approved_by`

Contemporaneous

`timestamp written in the transaction`

Original

`stored question · model · prompt version`

Accurate

`retrieved chunk · similarity · response_hash`

Enduring

`append-only log · version content_hash`

Available

`indexed, queryable, RLS-scoped trail`

Supporting these expectations is not the same as being compliant. Compliance is determined by validation, procedures, and qualification of the whole system (e.g. GAMP 5) — not by software alone. Nothing here is legal or regulatory advice.

RETRIEVAL

You don't sacrifice retrieval quality

Dense [2]

pgvector cosine search — captures paraphrase and semantic relatedness.

Lexical (BM25) [3]

Exact terms, rare tokens, IDs and codes that embeddings blur.

Hybrid (RRF) [4]

Reciprocal rank fusion recovers hits either mode alone would miss.

On BM25: PostgreSQL's native ts_rank is term-frequency based, not true BM25. Where ranking quality matters, mature extensions bring real BM25 into the database — no external search cluster.

`ParadeDB pg_search (Tantivy)` `VectorChord-bm25` `Tiger Data pg_textsearch`

Dense retrieval [2] · BM25 [3] · Reciprocal rank fusion [4] · BEIR [6] · BM25 extensions [13, 14]

REPORTED FIGURES · PUBLISHED SOURCES

The numbers: pgvector at scale

~50%

less vector memory with halfvec; similar search quality in AWS tests

AWS Database Blog, 2024 [12]

up to 30×

faster HNSW index build, pgvector 0.7.0 vs 0.5.1

AWS Database Blog, 2024 [12]

up to 67×

faster HNSW build with binary quantization — but AWS report a significant recall drop

AWS Database Blog, 2024 [12]

16 / 64 / 40

HNSW m / ef_construction / ef_search defaults

pgvector documentation [11]

2k / 4k / 64k

indexable dims: vector / halfvec / bit

pgvector 0.7 release notes [11]

tens of M

1536-d vectors on one instance when the index fits in RAM

rule of thumb — not a cited benchmark

Reported by third-party sources, not a benchmark of this system. Validate on your own data and workload.

THE ECOSYSTEM

What vendors and practitioners now report

> *“HNSW is the right default in 2026 — higher recall, lower latency, and it builds on empty tables.”*

paraphrased — practitioner guidance, 2026 [12]

> *Hybrid search can live entirely within PostgreSQL, with no external dependencies to synchronise.*

paraphrased — ParadeDB, 2025 [13]

> *Native BM25 ranking now runs in Postgres — no Elasticsearch sidecar required.*

paraphrased — Tiger Data, 2026 [14]

> *Parallel index builds and quantization delivered large HNSW build speed-ups over prior versions.*

paraphrased — AWS Database Blog, 2024 [12]

“Relevance wins attention. Evidence wins trust.”

AT A GLANCE

Traditional vs. SQL-first

	Traditional	SQL-first
Vector store	separate DB	**PostgreSQL**
Logs	elsewhere	**same database**
Metadata	duplicated	**same tables**
Permissions	in app code	**row-level security**
Reproducibility	hard	**reconstructable**
Backups	multiple	**single**
Consistency	many models	**ACID**

BEING HONEST

When this is overkill — and its limits

Don't build this when…

- consumer search, internal tools, low-stakes assistants — the evidence machinery is pure cost
- scale is extreme (billions of vectors, or a vector workload that must scale independently)
- you need globally replicated, low-latency search on every continent
- the team simply does not want to operate PostgreSQL

Honest limits

- **Append-only ≠ tamper-proof.** binds the app role, not a superuser; add hash-chaining, DB audit logging, WORM backups
- **Reproducible has a boundary.** covers retrieval + stored inputs, not re-running a deprecated model
- **Mind erasure obligations.** keep personal data out of the immutable trail; retain metadata, redact payloads

INDUSTRIES

Where evidence is the requirement

Financial services
loan decisions · AML · KYC

Healthcare
decision support · dispensing

Insurance
claims handling

Legal
case-law retrieval

Government
public-sector records

Pharma / GxP
SOP & policy retrieval

Energy & utilities
operating procedures

Aerospace / mfg
maintenance docs

REFERENCES

References

PEER-REVIEWED LITERATURE

1. **Lewis, P., et al.** (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33, 9459–9474.*

2. **Karpukhin, V., et al.** (2020). Dense passage retrieval for open-domain question answering. *Proceedings of EMNLP, 6769–6781.*

3. **Robertson, S., & Zaragoza, H.** (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval, 3(4), 333–389.*

4. **Cormack, G. V., Clarke, C. L. A., & Büttcher, S.** (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. *Proceedings of ACM SIGIR, 758–759.*

5. **Malkov, Y. A., & Yashunin, D. A.** (2018). Efficient and robust approximate nearest neighbour search using hierarchical navigable small world graphs. *IEEE TPAMI, 42(4), 824–836.*

6. **Thakur, N., et al.** (2021). BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. *NeurIPS Datasets and Benchmarks Track.*

7. **Ji, Z., et al.** (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55(12), 1–38.*

STANDARDS & TECHNICAL DOCUMENTATION

8. **U.S. Food and Drug Administration.** 21 CFR Part 11: Electronic records; electronic signatures. *Code of Federal Regulations, Title 21.*

9. **European Commission.** (2011). EudraLex Volume 4, Annex 11: Computerised systems. *EU Guidelines to Good Manufacturing Practice.*

10. **MHRA.** (2018). ‘GXP’ data integrity guidance and definitions. *Medicines and Healthcare products Regulatory Agency.*

11. **Kane, A., and the pgvector project.** (2024). pgvector v0.7.0 release notes; CHANGELOG 29 April 2024. *github.com/pgvector/pgvector (announced via postgresql.org).*

12. **Amazon Web Services.** (2024). Load vector embeddings up to 67x faster with pgvector and Amazon Aurora. *AWS Database Blog.*

13. **ParadeDB.** (2025). Hybrid search in PostgreSQL: the missing manual. *paradedb.com.*

14. **Green, T. J.** (2025). Introducing pg_textsearch: true BM25 ranking and hybrid retrieval inside Postgres. *Tiger Data.*

15. **TensorChord.** (2025). VectorChord-BM25: native BM25 ranking index for PostgreSQL. *github.com/tensorchord/VectorChord-bm25.*

All performance figures quoted in this talk are attributed to their published sources; none are benchmarks of the system shown.

“Put the truth in one place, and traceability stops being a project.”

CLOSING

Don't optimise only for retrieval.

Optimise for evidence.

The future of enterprise AI isn't just vector search. It's SQL-first retrieval with an audit trail you can trust — one engine, one audit trail.

Martins Otun · founder, ALGONIX AI Ltd · auditable AI for regulated industries

Questions welcome