ARDUA

Private AI Analytics: EntailDB + Ardua AI

A technical white paper on governed conversational analytics

Executive summary

Conversational analytics puts a business's data in front of a model: a live schema, a generated query, the rows that come back. That exposure is exactly what makes institutions cautious about AI in the first place. EntailDB answers business questions from connected databases and shows the evidence behind every answer. Ardua AI decides which model may see what, before it sees it, and keeps a record of why. Together, as Private AI Analytics, a database's declared sensitivity is checked before the first call to any model — not discovered afterward — while each product keeps doing the part it already does well. This paper explains the mechanism, states plainly what it does and does not cover, and shows the engineering evidence behind the claims.

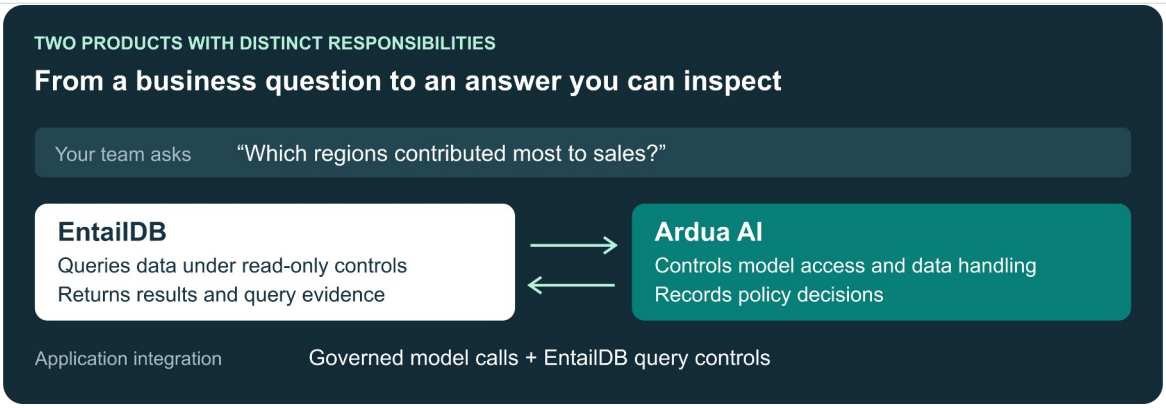
The problem

A fixed report cannot anticipate the next question. Conversational analytics fixes that by letting a model see a database's structure, generate a query, and interpret the result — which means the model's input necessarily includes a description of what data exists, and its context necessarily includes what the query returned. Handled carelessly, that is a data-egress problem wearing a chat interface: a schema for a restricted database, or the rows from a sensitive query, can end up as a string sent to whatever provider the application happens to be configured with, with no record of the decision and no way to enforce a different rule for a different database.

The response is not to avoid conversational analytics — it is to govern the path data takes before it reaches a model, and to keep evidence of what happened. That requires two different kinds of engineering: one that understands databases, queries, and how to keep an answer honest; and one that understands identity, policy, and how to enforce a boundary at the moment an action actually happens. Building both inside a single system tends to produce a system that does neither well. Ardua's architecture keeps them as two independently deployable products with one narrow, well-defined connection between them.

Two products, one controlled path

EntailDB owns everything about the database: connectors, read-only query controls, schema profiling, query evidence, and the mechanisms that keep an answer’s basis inspectable. Ardua AI owns everything about the AI execution: identity and scope, policy evaluation, provider selection, permission to act, and the audit record. Neither reimplements the other’s concerns — Ardua AI depends on EntailDB’s own accuracy mechanisms rather than building parallel ones, and the dependency runs only one way (technical detail in the appendix). EntailDB stays deployable on its own, exactly as it is today, whether or not Ardua AI is present.



EntailDB application integration: EntailDB controls queries, Ardua AI governs model calls.

How EntailDB earns trust in what it returns

A conversational analytics tool is only as trustworthy as its guarantee that it cannot change the data it explains and that its explanations are not quietly wrong.

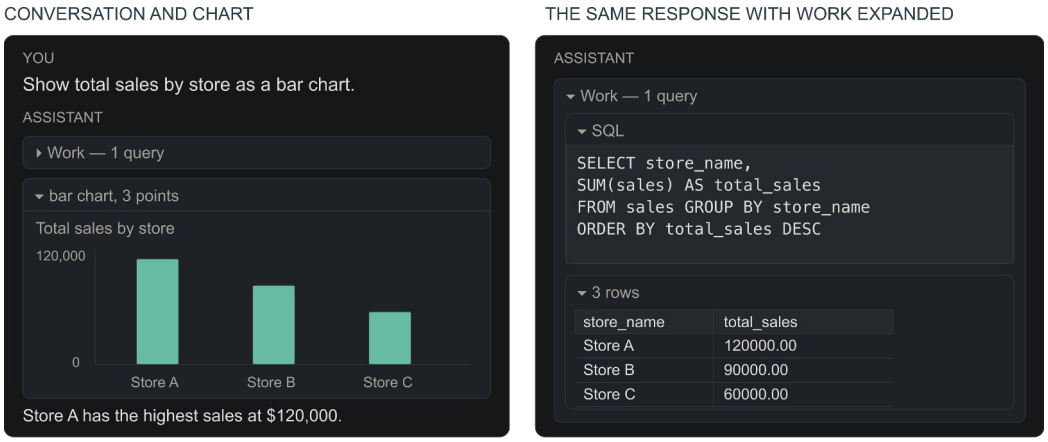
Read-only is enforced, not assumed. EntailDB’s connectors go through an allowlist gate: a statement must actually parse as a SELECT, checked after stripping comments, string literals, and quoted identifiers so a keyword hidden inside a string can’t fool the check, and evaluated only on top-level keywords so a subquery isn’t mistaken for a second, separate statement. That gate exists in its current form because an earlier, simpler keyword blocklist was demonstrably bypassed — a statement written as a common table expression reads as a SELECT to a naive scan, but executes as a delete:

```
WITH d AS (SELECT ...)
DELETE FROM d OUTPUT deleted.id
```

The response was not a patch to the blocklist; it was a structural allowlist plus enforcement of an actual read-only session at the database layer itself, so a bypassed application check is not the only thing standing between a query and the data.

Answers are checked against a live profile, not a stale assumption. EntailDB builds and refreshes a profile of a database’s structures and relationships, and supplies it to the model as a generated, current fact rather than a paragraph someone wrote once and never revisited. What actually closes the failure is more precise than “live facts replace stale ones”: in a controlled test, putting the generated fact in the prompt did not by itself stop a model from reciting a stale, hand-pasted statistic sitting next to it — the two competed, and the stale one kept winning. What eliminated the failure was a separate, explicit instruction naming which source is authoritative, which took the affected models from 20 fabricated recitations out of 20 attempts, and 9 out of 20 on a newer model, to zero on both. The profiler’s real contribution is upstream of that measurement: it removes the reason anyone hand-writes a statistic into a prompt in the first place, so there is nothing stale left to compete with the live source. Analytical traps that would otherwise produce a confidently wrong number — joins that silently inflate totals, hierarchies that leak rows, open-ended intervals — are checked structurally rather than left to the model to notice.

A partial answer says so. Previews are bounded at the query cursor, not by rewriting the SQL, and a bounded result reports its count as genuinely unknown rather than presenting a partial set as if it were complete. Every returned answer stays inspectable: the Work panel shows the SQL that ran and the rows that came back, so a reviewer is checking evidence, not trusting an assertion.



Two views of the same response, recreated from the current interface with fictional data. Expand Work to inspect SQL and returned rows.

How Ardua AI decides what a model may see

Ardua AI treats every model — local or cloud — as an untrusted inference engine. A model may propose an action; it is never the thing that authorizes one.

Classification travels with the data, not with the request. Data and requests carry a classification level along an ordered scale from public through internal, confidential, restricted, to the most sensitive tier, plus independent caveats for things like personal data or a hard no-external-egress requirement. When information from more than one source combines, the result inherits the most restrictive classification and caveats among its sources — never a milder one. An unlabeled source is treated as unknown provenance and handled restrictively by default, not assumed safe.

Enforcement happens at the point of the action, not only at the door. A single local policy decision point evaluates a request; separate enforcement points then check the actual action immediately before it happens — which provider is about to be called, which tool is about to run, what is about to leave the local boundary — because a decision made once at the start of an execution can be stale by the time new data has entered the conversation. If no model that satisfies the applicable policy is available, the request is refused. It is not silently sent to whatever provider is configured as a default.

Decisions are recorded without duplicating the sensitive data itself. A separate, append-only decision record captures which rule applied, which provider was used or refused, and why — enough to reconstruct a decision after the fact without routinely copying restricted source data into the audit trail alongside it.

The integration mechanism

Here is the specific problem EntailDB and Ardua AI solve together. By the time a conversational analytics workload has produced a queryable result, the first call to a model has usually already happened — that call is what generates the query, and its prompt already contains a profile of the database's schema. A database is not just its rows; describing its structure to a model is itself a disclosure. Governing only the query's returned rows leaves that first call, and everything it already reveals about a restricted database, completely ungoverned.

The integration closes that gap by moving classification to the front of the sequence. Each EntailDB connection carries an explicit classification and any relevant caveats, set at the connection level — a mixed-sensitivity database is labeled at its most restrictive content. The moment an execution selects that connection, its classification is resolved into the execution's context and evaluated by policy before any provider is chosen — not after a query runs, not after a result comes back. The rule is a strict sequence and nothing in the implementation is permitted to reorder it:

```
selected classified source  ->  initial policy context
  ->  policy decision  ->  provider eligibility
```

never the reverse — a provider chosen first, a query run second, and the source's sensitivity discovered only afterward. If the selected source is at the most restrictive levels, or carries a no-external-egress caveat, the entire execution is confined to an approved, private-processing path from that first decision onward, before the model has been shown anything about the schema at all.

This division of labor is deliberate and narrow. EntailDB attaches the classification and produces the evidence a decision needs; it does not decide what is allowed to happen with that classification. That decision, and the authority to enforce it, belongs entirely to Ardua AI. A capability behind a policy boundary is not the policy boundary itself — collapsing the two would mean trusting a database connector to also be a security control, a weaker guarantee than a dedicated enforcement point checking every action as it happens.

Two integration options, stated honestly

Not every deployment of these two products enforces the same things, and the honest description says which is which rather than claiming uniform coverage.

Governed inference with client-enforced actions. Inside the EntailDB application itself, EntailDB controls which queries run and Ardua AI controls what reaches a model, from which provider, and under what data-handling rule. Ardua AI's coverage in this mode is real and enforced for the model call itself; it does not extend to auditing what the client application does with a result afterward.

Platform-governed execution. Where the integration runs through Ardua AI's own execution platform, Ardua AI additionally authorizes the data-access action itself through its own enforcement points, in addition to EntailDB's controls, giving the platform direct authority over both the model call and the action that produced its input.

Both are legitimate deployment shapes. What is not legitimate, under this architecture's own discipline, is describing either one as "fully governed" without naming which of the two it is.

Deployment posture

The architecture is local-first, not local-only. Local models are preferred where they are capable, economical, and private, or where policy requires it; approved cloud models remain available where policy permits and the work benefits from them. The reference deployment target is Apple Silicon hardware running local inference, with cloud providers integrated as a permitted, policy-gated option rather than a default. EntailDB's independent deployability is preserved throughout — it runs standalone today, with its own provider settings, and gains governed model-handling only when explicitly integrated with Ardua AI. Standalone provider settings, used without that integration, do not enforce Ardua AI's data-handling policy.

What this does not claim

A governed-execution architecture is credible in proportion to what it admits it does not yet do. This integration does not construct a comprehensive graph of every place a piece of data has ever been — it uses a bounded reference that identifies a source, its classification, and its immediate parents, which is sufficient to make a correct decision at connection granularity without lineage down to individual rows or columns. It does not declassify data automatically; classification may be raised by automated means but is never silently lowered. It is not, at this stage, built for hostile multi-tenant isolation at internet scale. These are stated boundaries of the current architecture, not gaps discovered after the fact.

Conclusion

Governing conversational analytics means governing the moment before a model sees anything, not auditing the moment after. EntailDB's job is to make a database's structure and results trustworthy to inspect; Ardua AI's job is to decide, before a model is ever called, what that trustworthy result is allowed to reach and to keep a record of why. Together, as Private AI Analytics, they give an organization a controlled path to ask real questions of real data — with the query, the decision, and the evidence all available to check afterward. Organizations evaluating this path for their own data can ask to see the mechanism run against their own schema before deciding whether to rely on it.

ARDUA, INC.

Appendix — Technical notes

Dependency direction. Ardua AI may call into EntailDB's fidelity library to drive a governed analytics workload — supplying the provider object and policy-wrapped tools itself. EntailDB never imports, calls, or requires anything from Ardua AI. The relationship is one-directional by construction, which is what lets EntailDB remain independently deployable: removing Ardua AI leaves EntailDB fully functional; removing EntailDB removes one governed capability from Ardua AI, not the platform itself.