ALCHEMY LOOP

AI LLMQUERY; BUYER'S COMPARISON

Thirteen questions before you buy

Ask every vendor the same thirteen questions, including us. Most of what separates these products does not appear in a feature list, and several of the differences only surface after the data has already left your building.

Priced per seat, most tools in this category look cheaper until you count the seats. More importantly, they differ on things a demonstration will not show you: what leaves your network, whether one question can reach two systems, and what the product does when it is not sure.

01 What exactly is sent to the AI provider?

Why it matters. "Your data is secure" is not an answer. Ask for the list. Then ask them to show you, in the running product, what left the machine.

> No row returned to answer a question is ever transmitted, and no record about any matter, borrower, property or loan is sent to the AI provider. The query runs on your machine. The figures are assembled on your machine.

What is sent: your question, the database structure, your administrator's column notes, and your firm's own calculation conventions and settled definitions.

There are four places a value can travel, and we name all four rather than claim none: the codes of a coded column, a follow-up carrying the earlier query, the in-app assistant if you switch it on, and the repair of a query that failed. Turn on the request trace and read exactly what was sent. The in-app assistant is recorded too, in its own block, with your own guidance and the briefing written out in full.

Our product is **Servicer data safe**, and you can prove it yourself.

Evidence to ask for. The actual outbound request, from the running product, with every transmitted field named. A policy document is not evidence when the payload itself can be shown.

02 Can one question reach more than one database?

Why it matters. Nearly every product connects to several databases. Far fewer can answer a single question using two of them at once. Ask specifically: "what did we bill on our open files", where the files are in the case system and the billing is in the accounting system. Watch whether they open two tabs.

> One question, several systems, one figure. The matter comes from the case system, the money from the accounting system, joined on the number your firm actually uses to link them.

Two systems are combined only after somebody at the firm confirms, per pair and per kind of record, that the same record cannot be in both — because if it can, a combined total double-counts and looks entirely normal. Absent that confirmation each system answers on its own.

Every comparison between databases happens in the local process. Nothing about one database is sent to another.

Evidence to ask for. One question whose answer genuinely requires two independent databases, run in front of you. Several saved connections do not qualify, and neither do two browser tabs.

---

03 Is this a language model pointed at a database, or something more?

Why it matters. Most products in this category are a prompt wrapped around your schema. That works on a tidy demonstration database and degrades sharply on a real one with 500 tables, undocumented codes and three tables that all look like invoices.

> A multi-layer retrieval and reasoning architecture, not a wrapper. Question interpretation, schema retrieval, query decomposition, local computation and verification are separate stages, and several of them never call a model at all.

Ask any vendor what happens on a 500-table schema, and what their product does when two tables look equally plausible.

Evidence to ask for. The path from question to answer drawn out — planning, validation, execution, checking — with the stages that call a model marked, and what happens when two tables look equally plausible.

---

04 How does it learn what your firm means?

Why it matters. “Active”, “open”, “referred” and “closed” mean something specific in your firm and something different in the firm down the road. Ask how that knowledge is captured, who owns it, where it is stored, and what happens to it when the vendor changes their model.

Four kinds of knowledge, all yours, all stored in your own installation as readable files:

- **Business rules** — how your firm wants things counted.
- **Settled definitions** — a phrase decided once, with a record of who decided it and when.
- **Column notes** — what the codes in your database actually stand for, discovered by reading the data rather than guessing.
- **Settled questions** — a question a DBA has answered in SQL, which then runs verbatim with no AI call at all.

> Nothing is learned quietly. A correction somebody types in conversation does not become a standing rule by itself — a rule is written deliberately, checked, and stays visible, editable and removable. That is a real difference worth putting to every vendor: does a correction become a governed rule, temporary context for that conversation, or memory nobody can inspect?

Ask a competitor whether their equivalent survives a model change, and whether you can read it.

Evidence to ask for. Where the knowledge is stored, who approves it, and whether you can read, edit and remove it yourself.

05 Does a second AI check a rule before it becomes permanent?

Why it matters. A rule you write today steers every answer from now on. If the sentence still admits two readings, you will not find out from the product — you will find out months later when two people get different numbers from the same question. Ask any vendor what checks a rule before it is saved.

> When your leads switch it on, a second, different AI model is asked the same question at the moment a rule is being written. If the two models agree, the rule is tight. If they disagree, the sentence is still ambiguous and the person is told — while they are holding the pen.

Measured on our own data. Asked “how many milestones are overdue” with no rule in place, one model built its own query and returned 424; the other used the view named for the question and returned 41. Both were defensible. Given one sentence of business language, both produced character-identical SQL.

The two models must differ — a model checking its own work is not a check, and the settings screen refuses that pairing rather than pretending. It runs at rule-writing time only, never at answering time, so it costs nothing on an ordinary question.

Evidence to ask for. Whatever checks a rule before it is saved. If nothing does, ask what the product does when the sentence still admits two readings.

06 How was the accuracy figure measured?

Why it matters. This is where the numbers stop being comparable. A score on a tidy academic benchmark, on a vendor's own curated questions, or on a demonstration database tells you very little about your schema. Ask which benchmark, whether the databases were real, and whether you can run it yourself.

> Measured against BIRD-SQL, the cross-domain research benchmark built on large, messy, real schemas with inconsistent values and undocumented codes — deliberately not the tidy invented ones.

46% on day one, on a database it had never seen with nothing configured. **72%** once set up the way a firm sets it up. **85%** of everyday questions.

The scope stated precisely: 120 questions, stratified across all eleven databases of the BIRD-SQL Mini-Dev set and three difficulty bands, one generated query per question, no partial credit. This is our own internal evaluation against a public benchmark. It is not a leaderboard submission and not an independent certification, and we do not present it as either.

The harness ships with the product. You can run it against a copy of your own database and get the same tables we publish.

Scores from different benchmarks are not comparable with each other, including ours. If a vendor quotes a number, ask which test produced it.

Evidence to ask for. The test set, the scoring method, the model version and the failed cases — and the same set run twice.

07 What does it do when it is not sure?

Why it matters. The dangerous failure is not an error message. It is a confident, plausible, wrong number — and you will not catch it, because it looks exactly like a right one.

> It refuses, and says why. A total it cannot verify is not offered as an answer:

- Records whose status cannot be read, where any of them might be voided — refused, with the count.
- A figure where some records have nothing to match on — refused, naming how many.
- A total computed from a result the query limited before adding it up — flagged, with what was left out.
- Two systems never confirmed combinable — each answers separately.

Ask a competitor to show you their product declining to answer.

Evidence to ask for. The product refusing. Ask them to produce a question their system declines to answer, and watch what the user is told.

08 Can it write to your database?

Why it matters. Several products in this space are administration workbenches with a chat box attached — inline editing, imports, schema changes. That is a different risk profile from a reporting tool, and it is worth knowing which one you are buying.

> No. Read-only, enforced at four independent layers, and proven by 38 adversarial attacks in our own published test report — which states plainly that it is our testing rather than an independent assessor's, and what an assessor should target instead.

Evidence to ask for. Which controls are enforced in code and which by database permission. An instruction in a prompt telling the model not to write is not a control.

09 Where does it run, and what does the vendor see?

Why it matters. Ask whether your schema passes through the vendor's infrastructure, whether they hold an account for you, and what telemetry leaves. Ask what happens to your data if you stop paying.

> It installs on a workstation or a shared folder and listens only on the local machine. There is no cloud service of ours in the path, no account with us, and no telemetry. We receive nothing from your installation.

The AI provider account is yours, so the commercial and retention terms are yours to set.

Evidence to ask for. What leaves the installation, including telemetry and logs, and what becomes of your configuration if you stop paying.

10 Can somebody say which tables a question is about?

Why it matters. On a real schema there are three tables that look like invoices: the live one, the archive, and a staging copy from a migration nobody finished. A product that picks the wrong one answers correctly from the wrong data, and the figure looks ordinary. Nobody re-words their way out of that, because the words were never the problem.

> Type **@** and name the table. A list of your own tables appears as you type; choose one and the question is pointed at it. Nothing appears until you type the **@**, so it stays out of the way of people who never want it.

Beside each table it offers the ones it is joined to, and says *why* — a key your database declares, or a join that queries run here have actually used. Not "often used together", which would be a claim about usage frequency that we do not measure.

It is a preference, not a restriction. If answering genuinely needs another table it is still used, and the answer names what it added. If what you typed matched no table, the answer says so and tells you what it read instead — so nobody is left believing a question was narrowed when it was not. A table your firm has withheld is never offered and cannot be named.

Evidence to ask for. Ask their product a question where two similarly named tables could both answer it, and see whether you can say which one you meant — and whether the answer tells you which it used.

11 Does the built-in help know what you are doing?

Why it matters. Every vendor has help. The question is whether it is a documentation site beside a question box, or guidance that can see the screen you are on — and what it is given about your firm in order to do that.

> The help knows what you are doing. When somebody is settling a definition, correcting an assumption the product made, or looking at a question the safeguards refused, the built-in assistant is briefed on that specific situation — which item they are on, what their own preparation measured about it, which answers are on offer, and how many of their questions rest on it being settled. Eight different situations brief it, and it is told how to help in each.

It is briefed without being handed records. A briefing carries the wording somebody is drafting, the rules and settled definitions the firm already holds on that subject, table and column names, and counts from their own preparation run — and no rows, and no value read from a record. On a refused question it is told only which category was recognized, never the question itself, because that is the thing being kept inside the building; if somebody pastes it in, the assistant is instructed to stop them.

Ask any vendor what happens to their built-in help when the product updates. Ours is generated from the product and refuses to load against a build it does not describe: help that names a screen which has moved is worse than no help, because it is authoritative and wrong.

Evidence to ask for. Open the help from inside a task, and see whether it knows what the task is. Then ask how it is kept current, and what happens to it the day the product changes.

12 Can it answer questions on a system that keeps its fields as rows?

Why it matters. Most databases give each piece of information its own column — sale date, property state, loan number. Some do the opposite. They store one row per answer, tagged with a form and a field number, and put every answer in a single text column. It is called EAV or entity-attribute-value, and it exists so a firm can add its own questions to a form without anyone changing the database. This market runs on a handful of platforms — and **at least one of the major case management platforms is built this way**. Firms running it usually do not know, because nothing on screen says so.

What it costs you is that the database stops describing itself. There is no *sale_date* column to find; there is a column called something like *answertext* holding four hundred different kinds of answer, and a number beside it saying which question was asked. Most reporting tools are built to read column names. Here there is nothing to read.

> We work it out from the data: which column identifies the file, which names the question, which holds the answer, and which says *which form* it was on. That last one decides whether a figure is right — field 72 on the foreclosure form and field 72 on the title form are different questions, and a product that merges them returns a number built from two unrelated answers with nothing looking wrong. Measured on a corpus built from a real firm's export: 13,108 answers across 300 files and four forms, counted, filtered and listed correctly.

What we do not claim: recognizing the shape is not the same as knowing what field 54 *asks*. That lives in the platform's own field dictionary, and until a firm supplies it, we can count and filter but cannot put the question in business language. We say so rather than guess.

Evidence to ask for. Ask every vendor directly how their product handles an EAV entity-attribute-value database — what setup is needed first, who has to do it, and whether the answer stays right when two forms use the same field number. If your case management system is one of these, it is the question that decides whether anything else on this list matters.

13 What does it cost at your size, in year two?

Why it matters. Per-seat pricing is the one that moves. Work out the cost for everyone who will eventually want access — not the three people in the pilot — and ask what a second database costs.

> $750 a month. Unlimited users. Unlimited databases. Adding the tenth person costs nothing. Adding a second system costs nothing.

Stated plainly: you also pay your own AI provider directly for the questions you ask, because the account is yours. That is usually modest, it is visible to you rather than marked up by us, and a settled question costs nothing at all because no AI call is made.

** Your most-asked questions can cost nothing at all using our product! **

Most firms ask the same handful of questions over and over — the month-end set, the standing client reports, the figures somebody pulls every Monday morning. Your DBA can settle those as approved queries. A settled question then runs verbatim, with no AI call of any kind: no provider fee, no waiting for a model, and the same figure to the penny every time it is asked.

That matters twice over. On cost, your highest-volume questions — usually the bulk of what a team asks between one month-end and the next — stop consuming provider tokens entirely, so the AI bill falls as the firm settles more of its routine. On trust, a settled answer cannot drift: it is SQL a person at your firm reviewed and approved, not a fresh interpretation each time.

Worth putting to any vendor: can our DBA approve a query, so it bypasses the AI completely, and what does that question cost us afterwards?

Evidence to ask for. A written list of what the price includes and excludes: seats, questions, credits, databases, environments, connectors, provider fees, support and overages.

If you only run three tests

Thirteen questions will tell you a great deal. Three demonstrations will settle it.

- Ask to see the outbound request. Not a description of it — the actual payload, from the running product, with every field named.
- Ask one question that needs two databases at once, and look at what the answer says about coverage, and about records that could be sitting in both.
- Run an accuracy set twice. Not for the score — for whether the same question gives the same answer. A product that answers differently on the second run will do that to you in production, and nobody will be watching.

The third one is the test almost nobody runs, and it is the one that exposes the most. Ask for the differences between the two runs in writing.

How to use this

Send the thirteen questions to every vendor on your list, including us, and compare the answers side by side. Where a vendor cannot answer one of them in writing, that is itself an answer.

Figures for AI LLMQuery are from our own published testing data sheet and security documents, which ship inside the product and are available on request. Competitor pricing examples above are illustrative per-seat ranges for the category rather than a quotation from any named vendor; obtain current pricing directly.