

# Ask the Data in Plain English, Safely

*The people who need answers should be able to ask for them directly, and the tool that answers should never be able to change the data.*

|          |                                                                                             |
|----------|---------------------------------------------------------------------------------------------|
| PRODUCT  | Whisper-Wind                                                                                |
| AUDIENCE | Operations and business managers, analysts, support engineers, IT and data governance leads |

## The question queue

Every utility has a queue of questions waiting on someone who knows SQL. How many meters on this route missed reads last week? Which accounts were re-billed this month? What changed on this service point before the complaint came in? The people asking are analysts, managers, billing specialists and support engineers. The people who can answer are a small group of database-literate staff who already have full-time jobs.

So the questions become tickets. The answers arrive days later, often as a spreadsheet that raises a follow-up question, which becomes another ticket. During an incident, the delay is worse. A support engineer who needs to check the operational data right now has to find someone with access and the time to write a query.

The data is there. The people are capable. The gap is the language between them.

The cost falls on both sides. The people asking make decisions later than they should, or make them on older numbers. The people answering spend their days on routine lookups instead of the work that genuinely needs their expertise, such as data quality, performance and the next integration.

## Why it's hard

The obvious fix is to let people ask questions in plain English and have a tool turn them into queries. That idea makes data owners nervous, and they are right to be careful.

Utility databases hold billing records, meter data and customer information. A tool that writes queries could, in principle, write one that updates or deletes rows. Even a well-meaning question phrased the wrong way ("clean up the duplicate reads") could turn into a change nobody intended. Any natural-language tool has to answer that concern before it answers anything else.

Answers also have to be trustworthy. A number without context invites the wrong decision. People need to see the result in a form they can check, understand what the answer means, and take it somewhere else for further work.

And not everyone is sitting at a keyboard. Control room operators, lab technicians and field crews often have their hands full. For them, typing a question, or even clicking through several screens, is its own barrier.

## Principles of a good approach

---

**Safe by construction.** The assistant can answer questions. It can never change data. That guarantee should be built into the product, not left to the tool's good behavior.

**Use the live data.** Questions should run against the utility's current data, not a stale extract. That means working with the databases the utility already runs, such as PostgreSQL and Oracle.

**Answers people can use.** Results should arrive as an interactive table with a plain-language explanation of what the answer shows. One click should export the result to CSV for a spreadsheet, a report or a colleague.

**Conversations that persist.** Questions build on each other. Chat history should be saved automatically, and useful conversations should be pinned so they are easy to find again.

**Voice where hands are busy.** Voice navigation should move people between screens and trigger the actions they would otherwise click, across the applications they already use. In a control room, a lab or the field, speaking is faster than reaching for a mouse.

**Fit the enterprise.** Users sign in the way they already do, and the assistant sits alongside the applications they use every day, not in a separate tool with separate rules.

**Keep good data hygiene anyway.** A read-only assistant is one layer. Utilities should still give any query tool only the database access it needs. Good practice is to have several independent protections, and a sound tool should work comfortably within them.

## What it looks like in practice

---

### ILLUSTRATIVE EXAMPLE (HYPOTHETICAL)

A meter operations manager wants to know which routes had the most missed reads last week. Instead of filing a ticket, she types the question into the assistant. A table comes back, sorted by route, with a short explanation of what was counted and over what period. She asks a follow-up in the same conversation: which of those meters also missed reads the week before? The answer narrows the list. She exports it to CSV and sends it to the field supervisor, then pins the conversation to revisit next week.

That afternoon, a support engineer is working an incident: some customers report estimated bills. He asks the assistant which of the affected accounts had reads loaded late in the billing window. The answer comes back in the conversation, with an explanation he can paste into the incident notes. He did not have to wait for a database administrator, and he did not need write access to anything.

Later, someone types, "Delete the duplicate reads from last week." The assistant does not run it. It cannot change data, so it refuses the request. Changes to the data stay with the team that owns it, under its own change process.

In the control room, an operator with a headset and both hands on other work says, "Open the outage dashboard." The screen changes. She asks for the list of affected feeders, and the screen changes again.

None of those people wrote SQL. None of them could have changed a row.

## Questions to ask any natural-language data vendor

---

- Is read-only access guaranteed by the product, or does the tool only promise to behave?
- What happens when a user asks for something that would change data?
- Does it query the live data in the databases we already run?
- Does every answer come with a plain-language explanation? Can we export results to CSV?
- Are conversations saved, and can users pin the useful ones?
- Does it offer voice navigation for people whose hands are busy?
- Does it fit the sign-in and access practices we already have?

## Answers without the wait

---

Whisper-Wind is the conversational layer of the Grid Data family. It turns plain-English questions into safe, read-only queries against the utility's live data in PostgreSQL or Oracle, returns tables with plain-language explanations and CSV export, and adds voice navigation for control rooms, labs and the field.

Learn more about [Whisper-Wind](#), or [talk to our team](#) about shortening your question queue.

### ABOUT PERINIMBLE

PeriNimble builds mission-critical grid data software for energy and utility companies. Founded in 2022 and headquartered in Bloomington, Minnesota, we support more than 15 million meters and over 1 billion interval reads per day in production.

**Built for systems that can't afford failure.** · [perinimble.com](https://perinimble.com) · [perinimble.com/contact](https://perinimble.com/contact)