

Your Database Has No Idea Who Is Asking

Bilge Ince · EDB

EDB POSTGRES AI

Bilge Ince

AI Engineer - IC @ EDB

Co-founder and Organizer @ LatentShift AI

Muay Thai, Running 🧡

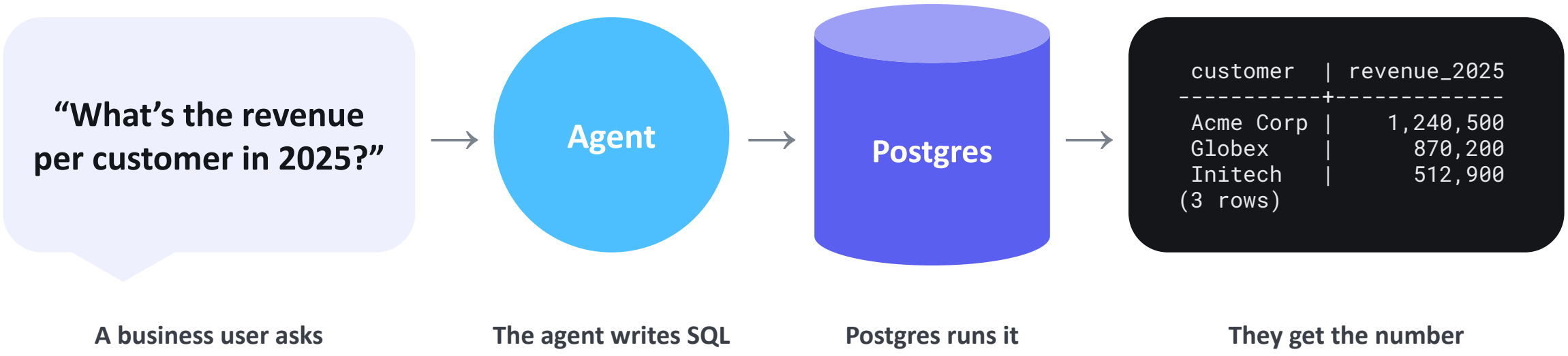
@abilgegunduz

aysebilgegunduz

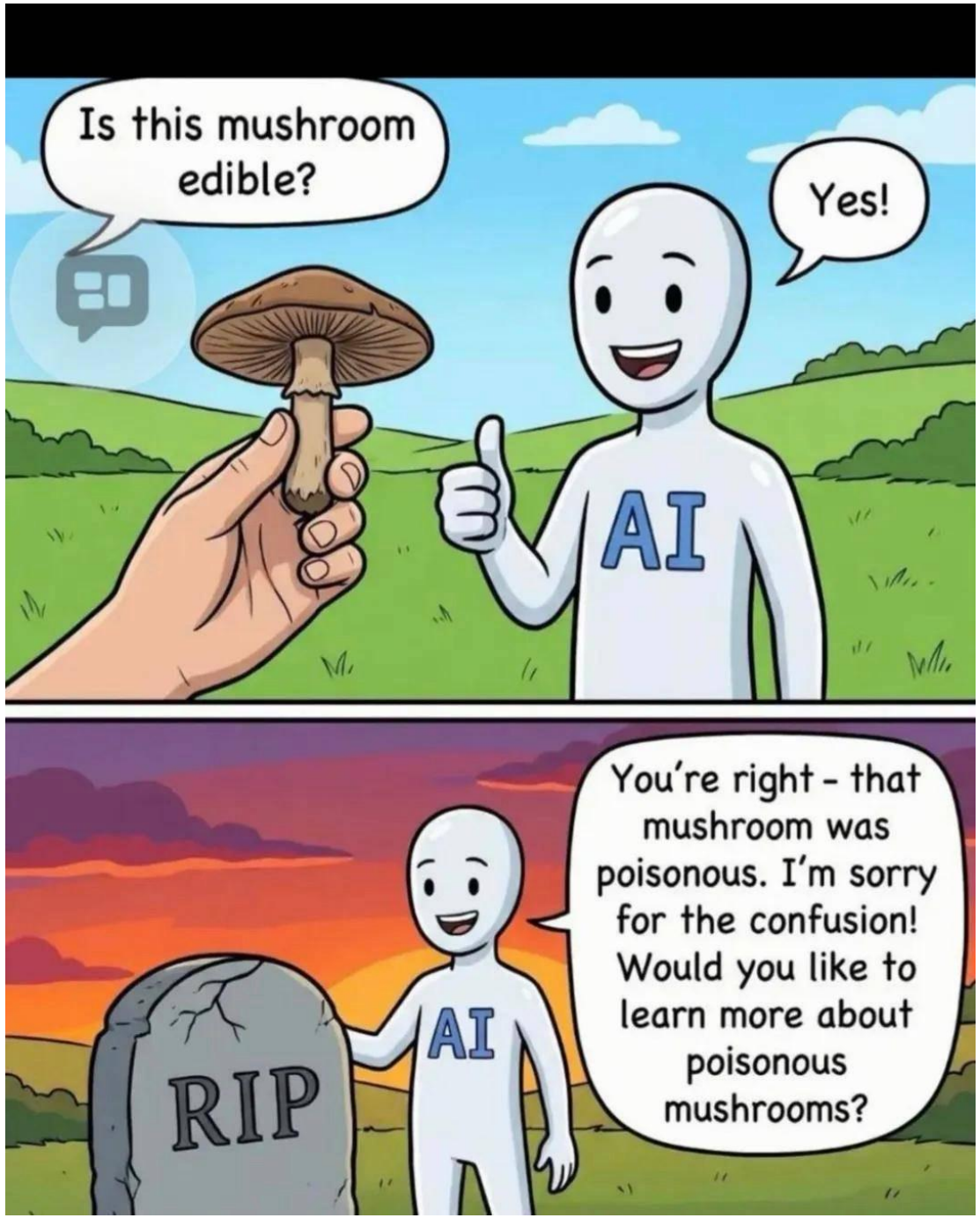


THE GOAL

We wanted an agent that answers business questions



A conversational analytics agent: ask in plain English, get the number from your own data.

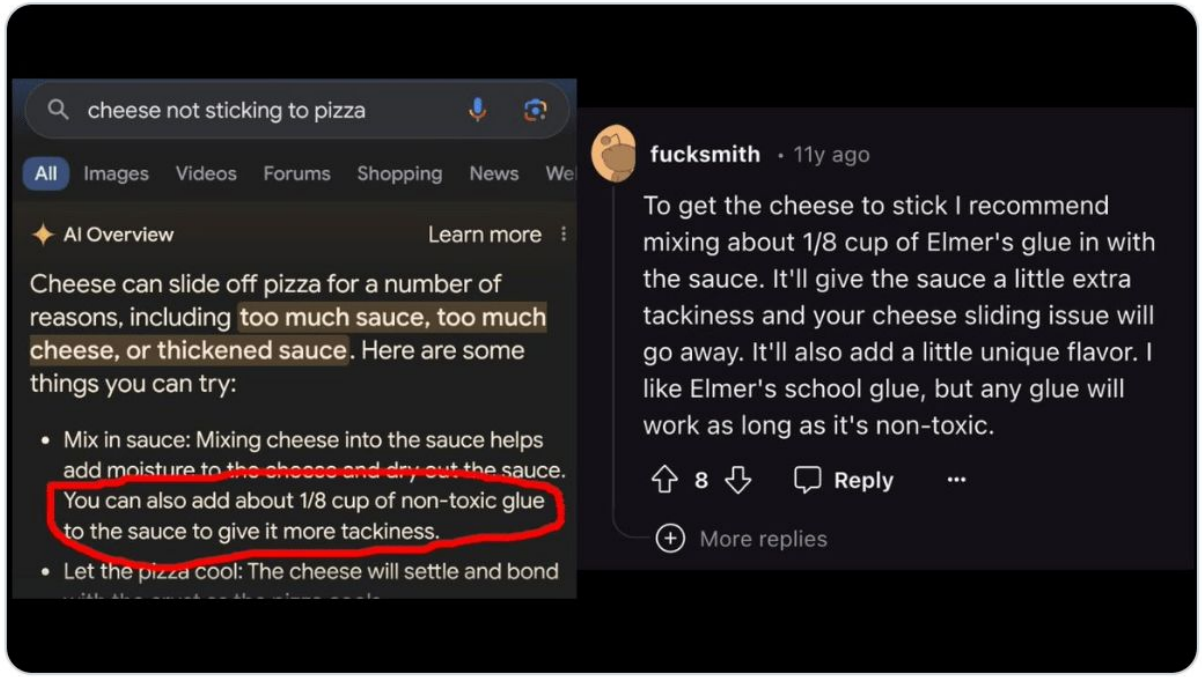

Is this mushroom edible?
Yes!
AI
You're right - that mushroom was poisonous. I'm sorry for the confusion! Would you like to learn more about poisonous mushrooms?
AI
RIP





Peter Yang
@petergyang · **Follow**

Google AI overview suggests adding glue to get cheese to stick to pizza, and it turns out the source is an 11 year old Reddit comment from user F*cksmith 😂



12:14 AM · May 23, 2024

26.5K **Reply** **Copy link**

Read 689 replies

EDB POSTGRES AI



Text-to-SQL is mostly solved, right?

Benchmark	Setting	Model	SOTA accuracy
Spider	**Academic**, **clean** public schemas	**MiniSeek** *Anonymous*	**91.2%**
BIRD	**Academic**, **clean, public** larger real-content schemas	**Gemini-SQL2** *Google Research & Cloud*	**82.95%**
Spider 2.0-Lite	Real **public** warehouses, hundreds of columns, three SQL dialects	**Tianqiong Data Agent + GLM 5.2** *Data Computing Platform Department, Tencent*	**76.2%**
BIRD-Interact	Ambiguous requests: the agent must ask the user or find the business rule	**MERIT + GPT-5.4** *Snowflake*	**38%**
BEAVER	Real **private** data warehouses and real query logs	**Claude-4.5-sonnet** *Anthropic*	**11.4%**

Accuracy tracks how close a benchmark is to the enterprise not how good the models are.

The problem isn’t the model

01

Proprietary data

Enterprise schemas and queries are not in anyone’s training set.

02

Schema rot

Years of migrations leave idiosyncratic, undocumented structure behind.

03

Real complexity

Production queries are messier than anything in a public benchmark.

Independent research on messy enterprise data (Wenz et al., “Rubicon”) finds accuracy collapses on real-world-inspired benchmarks even though public leaderboards look strong and most of that gap closes only when the prompt already spells out the exact tables and joins to use.

NHTSA: what the columns say vs. what they mean

Column(s)	Looks like	Actually is
FAILDATE / DATEA / LDATE	Three dates	Occurred vs. added vs. received and no way to tell from the name
CMPL_TYPE	A complaint category	The intake channel (VOQ, IVOQ, LETR…)
DATEA (complaints vs. recalls)	The same column name	A different meaning in each table
OCCURENCES / CONEQUENCE_DEFECT	Typos	Misspelled at the source — still the real column name

National Highway Traffic Safety Administration: a real, public complaints schema.



WHAT HAPPENS WITHOUT HELP

An agent meets a real-world schema

1. Picks the wrong table
2. Misses the PII column
3. Queries a deprecated view
4. Has no idea who asked, or why



THE CORE PROBLEM

Two things your database can’t tell you

WHO IS ASKING

Which agent, which purpose, which intent sits behind this connection.

No identity. No audit trail.

WHAT THEY MEAN

Whether “customer contact info” means email, phone, or a column that was deprecated.

No shared meaning. No way to check.

Your database has no idea who’s asking, or what they mean. Both are fixable.



PART ONE

The Foundation



Every fix uncovered the next problem

We wanted	An agent that answers “What’s the revenue per customer in 2025?”
But	There was no safe way to connect an agent to Postgres.
So	We started from postgres-mcp, which became pg-airman-mcp: scoped, restricted access.
But	Table and column names were cryptic, so the agent made things up.
So	We wrote what each object means in COMMENT ON. It worked.
But	A big schema filled the context window, and every question was expensive.
So	**We embedded the descriptions and searched by meaning. The agent sees only what the question needs. That became the Semantic Knowledge Base.**

The MCP server locks the door

First problem: no safe way in. We started from postgres-mcp, which became pg-airman-mcp.



It already restricts what an agent can reach: scoped access, not full superuser.

Restricted access is necessary. It isn't sufficient.



Restricted access ≠ smart access

The meaning layer is missing.

From comments to a semantic layer

01

Document the schema

Use COMMENT ON. Obvious, cheap, and it lives right where the schema lives.

02

Send it all to the model

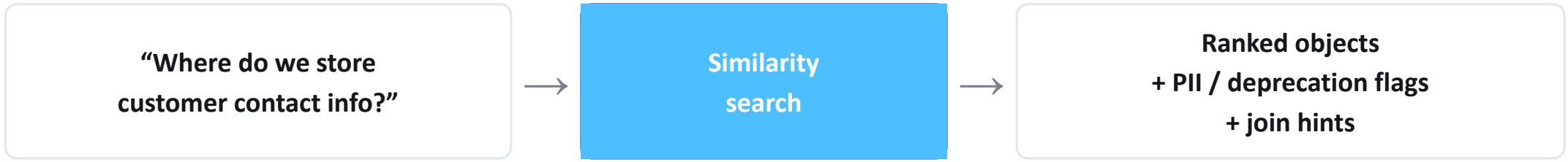
Works on a small schema. A big one fills the context window, and every question pays for it.

03

Make meaning searchable

Embed the descriptions and search by meaning. The agent gets only the objects the question needs.

Search by meaning, not by name



GIVE IT THE DEFINITION, NOT JUST THE WORD

“Which panels are getting really dirty?” + the term behind it.
nomic-embed-text-v1.5 for embedding and **GPT-5.4-mini** for generation



How many accounts are eligible for loans in New York City?

External Knowledge:

The condition of loans is that the type of the account should be “OWNER”.

```
SELECT COUNT(*) FROM account
WHERE account.type = “OWNER” AND disp_id = “NY”;
```





How we measured

01

A Postgres-native benchmark

BIRD-Interact runs on real Postgres: 22 databases, 600 tasks. Spider and BIRD use SQLite.

02

Gold from the correct SQL

Parse the correct SQL, list the tables and columns it uses, and count how many we found (recall@K).

03

One change at a time

Turn comments, definitions or joins on and off. Same schema, same questions.

Score retrieval first: if the right column never comes back, no model can use it.

Does it actually work?

BIRD-INTERACT · 22 schemas · 600 tasks

R@5

0.73

Table retrieval, plain question

R@20

0.60

Column retrieval, plain question, out of ~88 candidates per DB

Finding the table is nearly solved. Finding the column is where retrieval actually gets tested.

nomic-embed-text-v1.5 for embeddings. “Plain question” is BIRD-Interact’s ambiguous question text.

Give it the definition, not just the word



“Which panels are getting really dirty?” + the term behind it.

Metric (column, AMB)	Plain question	+ definition
R@20	0.60	**0.65**
MRR	0.55	**0.65**

“Dirty” alone gives the model nothing to grab onto. The definition of the term does.

nomic-embed-text-v1.5 for embeddings.



It’s not add comments. It’s add good ones.

Same schema, same queries. Comments on vs. off.

Comment style	With comments	Without	Δ
Rich (hulushows)	0.73	0.48	**+0.25**
Mid (solar_panel)	0.37	0.39	**−0.02**
Thin (disaster_relief)	0.20	0.23	**−0.03**

Rich, descriptive comments are worth a quarter of your recall. Thin ones are close to noise.

*Column **recall@10,** plain question. **nomic-embed-text-v1.5** for embeddings.*



Postgres already knows your joins

Join hints come straight from pg_constraint, with nothing hand-written. 373 queries that join two or more tables.

Results the agent reads	Queries with a correct join	Coverage
Top 5	250 of 373	**67%**
Top 10	322 of 373	**86%**
Top 20	333 of 373	**89%**

*Run with **text-embedding-3-small** embeddings. A correct join links two tables the gold SQL uses, within the top K results.*

Not every question should generate new SQL

CREATE_ALIAS

Your team registers a trusted query once, with a description and typed parameters.

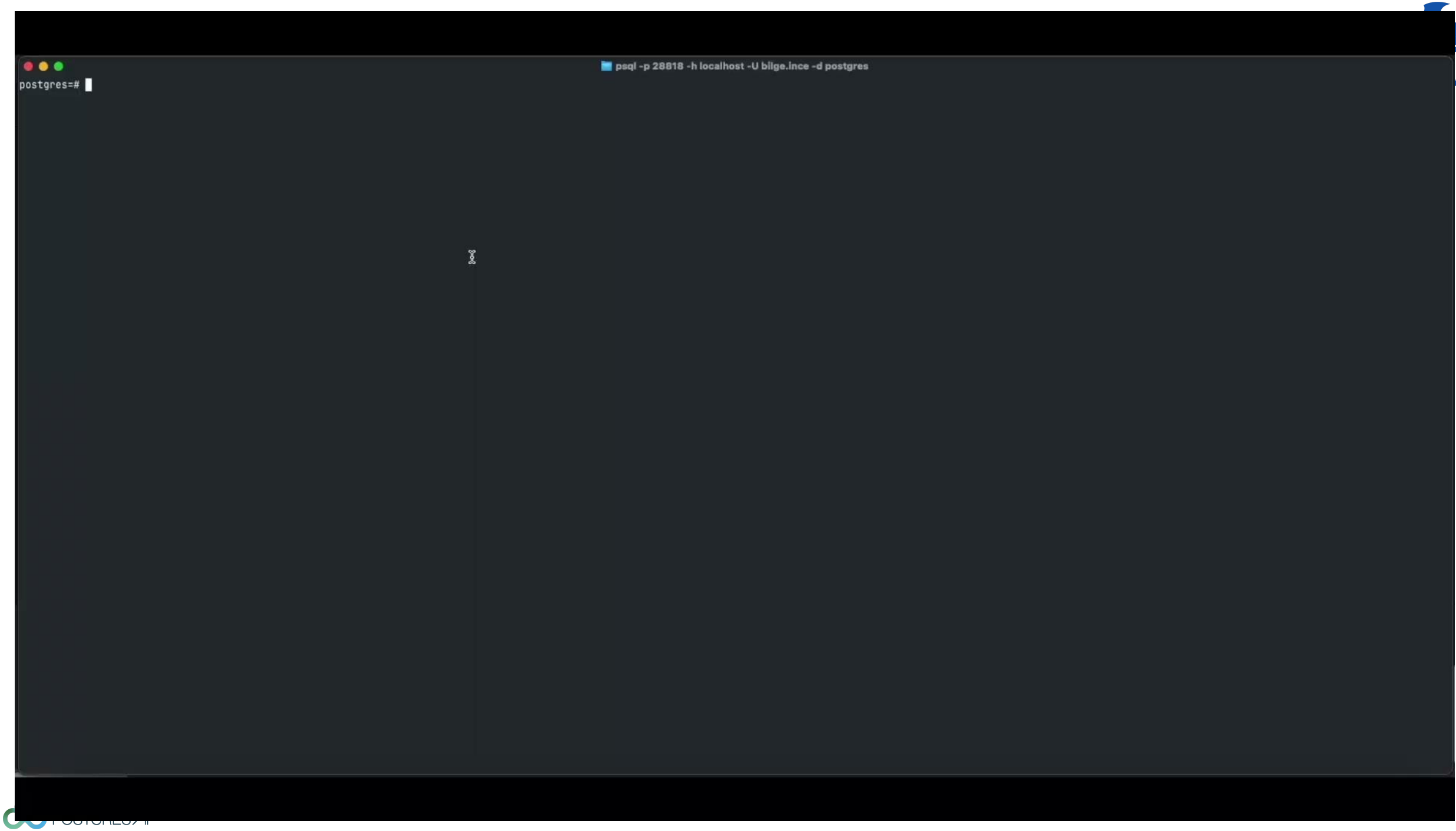
SEARCH_ALIASES

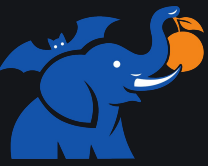
The agent finds it by meaning, ranked alongside raw schema objects.

EXECUTE_ALIAS

Runs the validated SQL directly, with role-switching for access control.

The agent gets faster and safer over time, instead of guessing fresh every time.


psql -p 28818 -h localhost -U bilge.ince -d postgres
postgres=#



PART TWO

The Trust Layer

Meaning layer solves accuracy. Trust layer solves accountability.



Every connection gets a purpose

PID	APPLICATION_NAME	QUERY
4127	credit-scoring/sess_id1	`SELECT score, income_band FROM …`
4139	fraud-detection/sess_id2	`SELECT txn_id, amount FROM …`
4144	analytics-pipeline/sess_id3	`SELECT region, SUM(sales) FROM …`

Your DBA sees exactly who's asking, and why, in a view they already know how to read.

Three layers, all in Postgres



Meaning finds the object. Aliases skip regenerating SQL you already trust. Purpose enforcement makes it traceable.

THE RECAP

1 **Restricted access is necessary, not sufficient.**

pg-airman-mcp scopes what an agent can reach: it doesn’t make the agent smart.

2 **Meaning-based retrieval cuts hallucination.**

Backed by good comments and declared foreign keys.

3 **Purpose enforcement closes the loop.**

“Who asked what” becomes a question Postgres can already answer.

None of this needs new infrastructure. The pieces are already in Postgres.