

AGENT READY POSTGRES

Taking AI agents to production: effective, seamless, and governed

Jigar Mody · Bilge Ince

EDB · PGConf EU 2026



THE PROBLEM

MOST AI PROJECTS STALL BEFORE PRODUCTION

Studies blame the systems around the model more than the model itself.

5%

reach production

60% of organizations evaluated custom enterprise AI tools. 20% reached a pilot. 5% reached production.¹

40%+

canceled by the end of 2027

Gartner's prediction for agentic AI projects. The causes: escalating costs, unclear business value, and weak risk controls.²

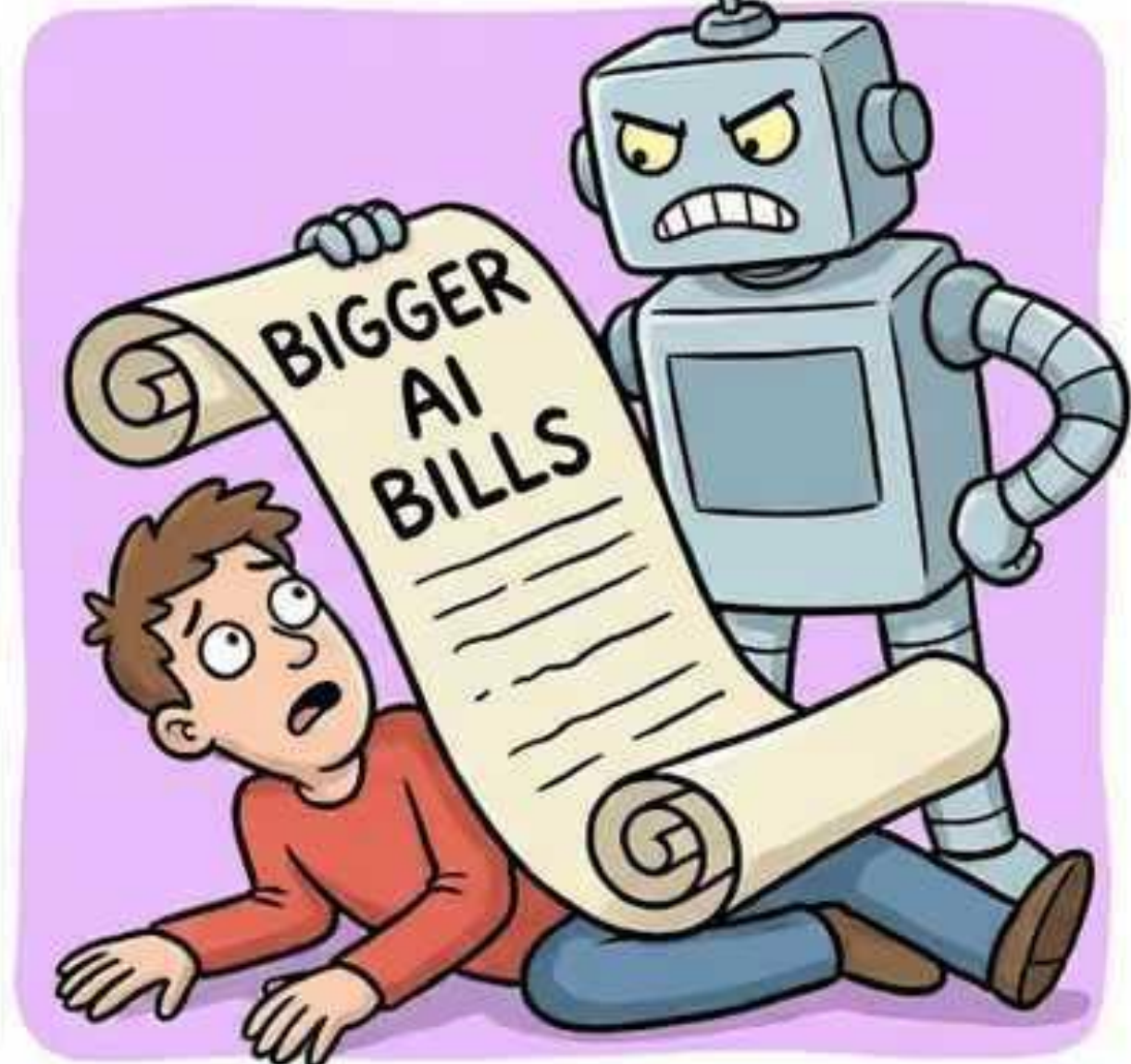
1 in 5

has mature agent governance

Deloitte's 2026 survey: only one company in five has a mature governance model for AI agents.³

THE QUESTION How do we make agents effective, with a seamless path to production, while keeping governance?

¹ MIT NANDA, The GenAI Divide: State of AI in Business 2025 (custom enterprise AI tools). ² Gartner press release, June 25, 2025 (prediction). ³ Deloitte, State of AI in the Enterprise, 2026.





THE TALK IN ONE SLIDE

TWO THINGS AGENTS NEED FROM POSTGRES

Agents are now real database clients. We see a two-fold opportunity for Postgres.

A · AGENT MEMORY

Better outcomes

LLM calls are stateless, so agents forget everything between calls. Memory in Postgres lets agents keep context, send fewer tokens, and reuse what they learned across sessions.

B · AGENT-RESILIENT POSTGRES

Governed adoption

Agents act at machine speed, retry often, and write SQL that nobody reviewed. Governance has to hold even when the agent is wrong, confused, or attacked.

ROADMAP FOR THE NEXT 40 MINUTES

01

Why Postgres

02

Agent Hub

03

Agent Memory

04

Semantic KB

05

Governance

06

What is next

01

WHY AGENTS NEED POSTGRES

Agents are a new kind of database user.



THE SHIFT

AGENTS ARE THE APPLICATIONS OF THE AI WORLD

Apps follow a script. Agents decide. The database stays the system of record.

01

They never get tired

An agent can run 25 reasoning steps for one question, call tools in parallel, and retry a failed call again and again.

02

They are non-deterministic

The same question can take a different path on every run. The SQL an agent sends was generated seconds ago, and nobody reviewed it.

03

Models change. The system of record stays.

The model is probabilistic and the agent is dynamic. The database is the system of record: it knows what is true now, and what changed.

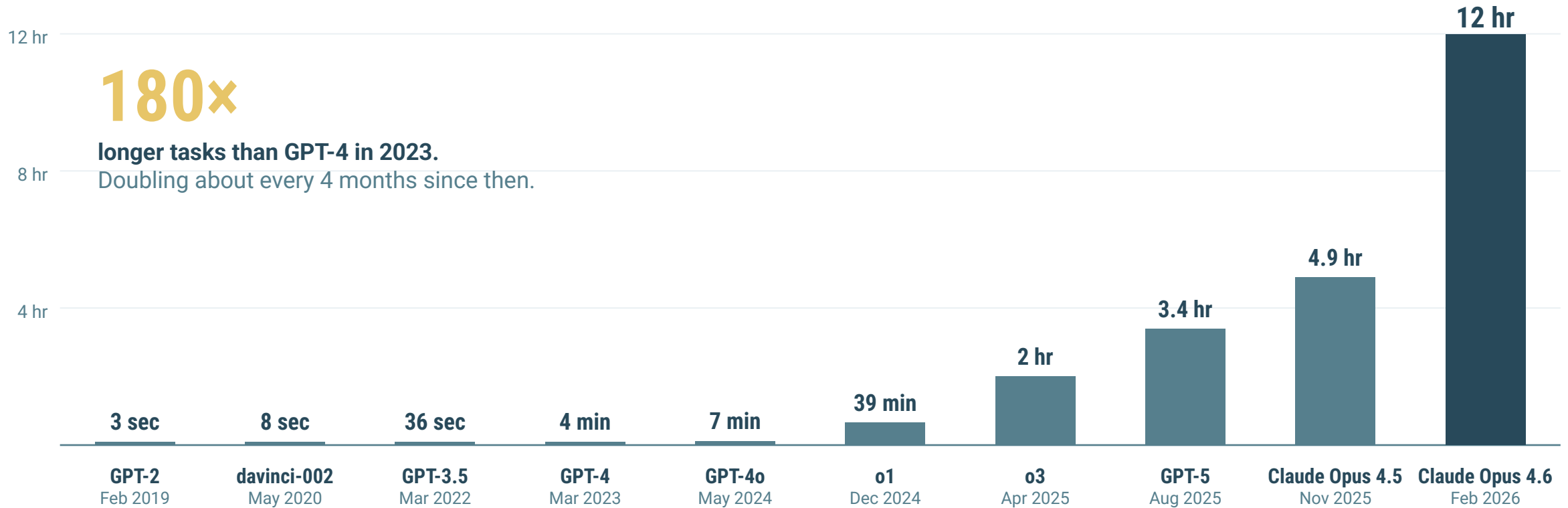
SO WHAT The database is the right place to make agents effective and governed: it is the system of record when everything else is dynamic.



THE PACE OF CHANGE

FROM SECONDS TO HOURS IN SIX YEARS

How long a software task an AI agent can finish half the time, measured in human-expert time.



Source: METR, Task-Completion Time Horizons of Frontier AI Models (Time Horizon 1.1, data as of Sept 2026). 50% time horizon on software, ML, and cybersecurity tasks. Not every model was measured. Claude Mythos Preview (early) omitted: METR rates results above 16 hours as unreliable.



POSTGRES ALREADY HAS THE PRIMITIVES

IT'S JUST POSTGRES

Developers already run this database¹ and already use AI to work with SQL.² Agents can ship on the same stack.

01

Memory and transactions

Agent memory lives in Postgres. An action and its memory commit together, or roll back together, and raw turns are never lost.

02

Governance by purpose

Purpose is declared and bound to a role at session start, enforced on every call with your GRANTS and RLS, and recorded.

03

Meaning layer

The Semantic Knowledge Base helps agents find the right tables, joins, and approved queries by meaning. Agents propose, people approve.

04

Extensible

aidb is a Postgres extension. There is no extra service to deploy, secure, or back up.

PRINCIPLE Start where your team already is.

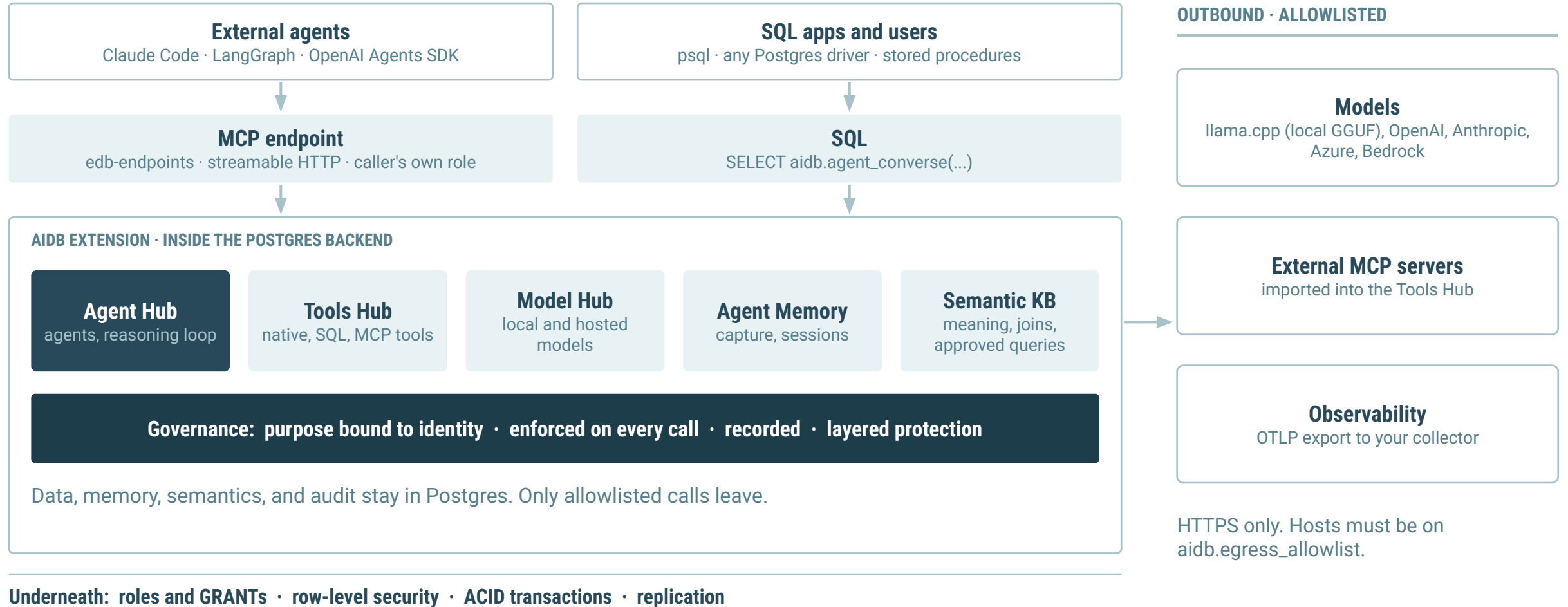
¹ Stack Overflow Developer Survey 2026: PostgreSQL is the most used database (57.9%), the most admired (62.3%), and the most desired (43.9%). ² Same survey: SQL is the second most used language (58.4%), and 29.8% of respondents used AI tools or agents to work with it.



ARCHITECTURE

EDB PG AI TURNS POSTGRES INTO AN AGENT RUNTIME

One extension. Agents, tools, models, memory, and semantics, all governed by Postgres.



02

AGENT HUB: IT'S JUST POSTGRES

From pilot to production on the Postgres you already run.



FROM PROMPT TO PRODUCTION AGENT IN ONE CALL

One definition holds the instructions, model, tools, and budgets. It lives in `aidb.agents`.

CREATE_AGENT

```
-- 1. Define the agent once
SELECT error FROM aidb.create_agent(
  'sales_agent',                                -- a name
  'Answer sales questions. Never guess.',      -- instructions
  'claude_sonnet',                             -- any Model Hub model
  tools => ARRAY[
    'catalog_list_objects',                    -- native tool
    'late_orders',                            -- your SQL tool
    'search_issues'],                         -- from your MCP server
  delegates => ARRAY['forecast_agent'],        -- agents it can hand off
  to
    purpose => 'sales_analytics',              -- policy scope: one role
    max_iterations => 10,                      -- budget: reasoning steps
    budget_strategy => 'summarize');           -- what to do at the limit

-- 2. Talk to it: each call is one turn
SELECT message, conversation_id
FROM aidb.agent_converse('sales_agent',
  'How many EMEA orders are late?');
```

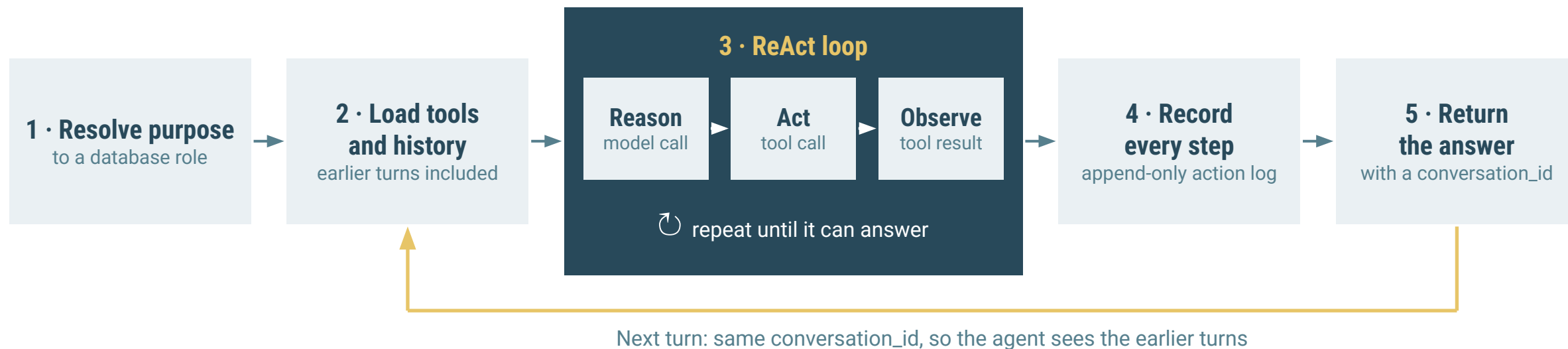
WHAT THE DEFINITION HOLDS

- **Instructions and model.** Any Model Hub model: OpenAI, Anthropic (also on Azure and Bedrock), or a local GGUF model through llama.cpp.
- **Tools.** Names from the Tools Hub catalog.
- **Delegates.** Other agents. The model sees them as ordinary tools.
- **Budgets.** Iterations, input and output tokens, timeout, and what to do when a limit is hit.
- **Output type.** An optional schema for structured answers.



ASK IN SQL, GET AN ANSWER BACK

Each call is one conversation turn. Inside it, the agent reasons, acts, and observes until it can answer.



CONVERSATION TURNS

TURN 1

Q How many EMEA orders are late?
A 42 orders. Most of them ship with one carrier.

TURN 2

Q Which carrier is it?
A Carrier B: 31 of the 42.



TOOLS HUB: ONE CATALOG, ONE DISPATCH

Three kinds of tools. One function to call them: `aidb.run_tool(name, arguments)`.

NATIVE TOOLS · 85

Built into aidb: catalog discovery, database health checks, Semantic KB search, pipelines, models, and AI primitives such as chunking, OCR, and embeddings.

```
SELECT
aidb.run_tool('catalog_list_objects',
  '{"schema": "sales"}');
```

SQL TOOLS

Any SQL statement becomes a tool. aidb decides if the tool is read-only from the command tag. It never trusts the caller's claim.

```
SELECT
aidb.create_sql_tool('late_orders',
  'Late orders for a region',
  'SELECT id, customer FROM sales.orders
  WHERE region = ${region}
  AND ship_by < now()',
  aidb.tool_params(aidb.tool_param(
    'region', 'TEXT', 'Sales region')));
```

MCP TOOLS

Import tools from any external MCP server, with an optional filter. Header secrets are hidden from normal users.

```
SELECT
aidb.import_mcp_tools('github',

  'https://mcp.example.com/mcp',
  tool_filter =>
  ARRAY['search_issues']);
```

03

AGENT MEMORY: BETTER ANSWERS, FEWER TOKENS

Capture fast. Think later. Never lose the raw data.

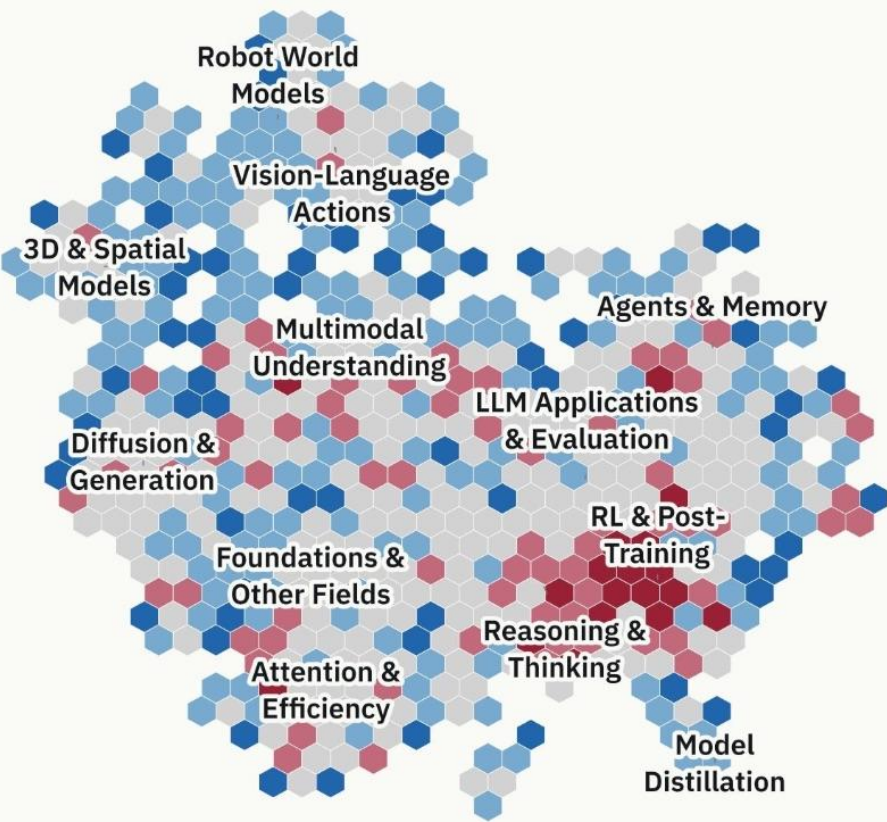
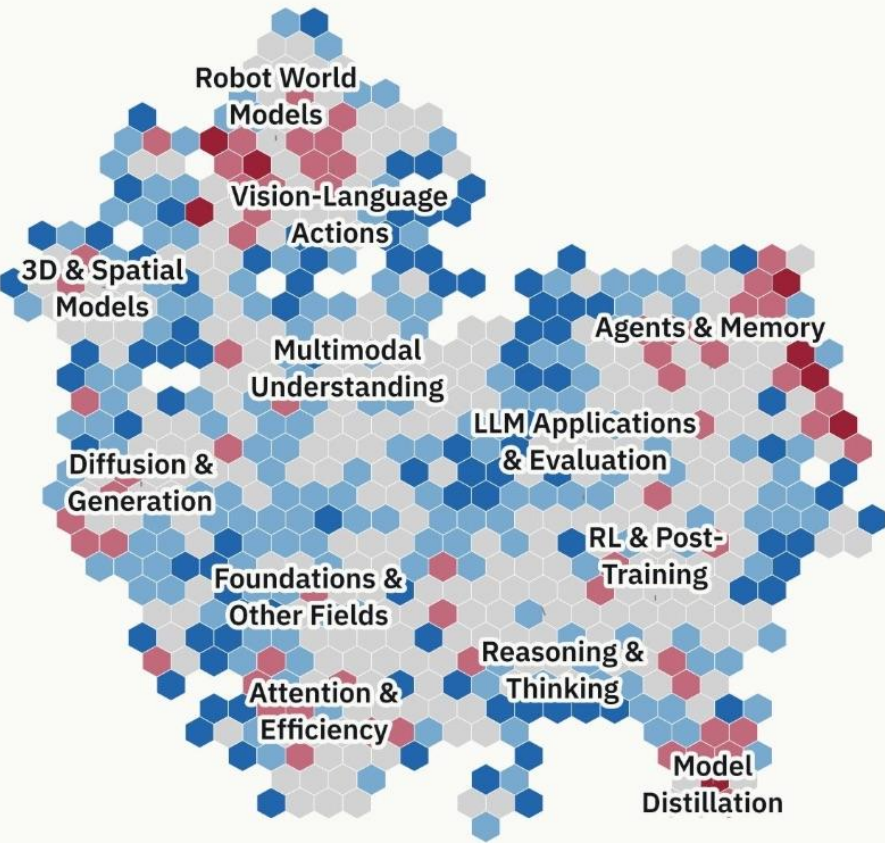


Topic Viewership



2025

2026



Viewership



WHAT AGENT MEMORY MUST SOLVE

Memory is not a bigger context window. It is choosing what to keep, and finding it again.

Every LLM call starts from zero.

Token explosion

Resending the full history makes cost and latency grow with every turn.

Context limits

Even very large context windows fill up in multi-step, multi-day work.

Quality drop

Models get worse at finding the one relevant fact in a long, dense context.

No learning

Without memory, agents cannot reuse preferences, facts, or decisions across sessions.

GOAL Retrieve only what is relevant, when it is needed. Keep prompts small and agents consistent.



FOUR LAYERS OF MEMORY

Each layer has one job. Model cost is kept out of the fast path.

L0

Capture

`aidb_memory.add()`

Store every turn verbatim, durably, and fast. No model call on this path.

AVAILABLE

L1

Session

`aidb_memory.session_get()`

Serve the current session back within a character budget. It compacts itself when it grows.

NEW IN 7.7

L2

Query

`search · recall · summarize`

Find ranked memories, get a reasoned answer with evidence, or get a grounded summary.

IN DESIGN

L3

Long-term

`background workers`

Distill raw turns into facts, past cases, procedures, and guardrails. Supersede and decay; never hard-delete.

IN PROGRESS



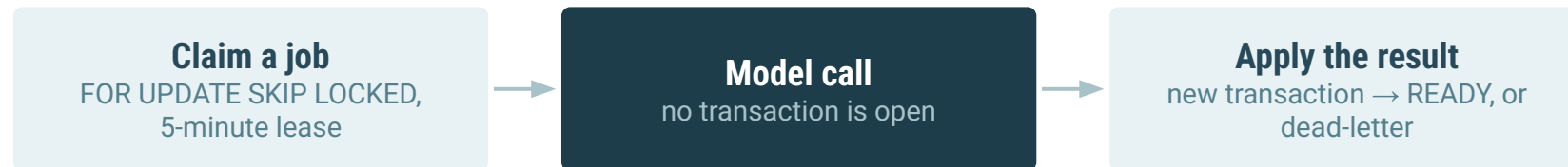
CAPTURE FIRST, THINK LATER

A fast stream that only captures, and a slow stream that does the expensive thinking.

FAST STREAM · INSIDE THE CALLER'S TRANSACTION



SLOW STREAM · BACKGROUND WORKER



- **Safe retries.** An idempotency key returns the existing event instead of a duplicate.
- **Pinned jobs.** Each job is bound to the provider generation that accepted it.
- **At-least-once.** An expired lease makes the job claimable again.
- **Honest errors.** Infrastructure failures are counted apart from bad payloads.



SESSION MEMORY THAT COMPACTS ITSELF

`session_get()` returns the recent conversation within a budget, and summarizes the old part when needed.

BEFORE · RAW TAIL IS LARGER THAN LIMIT_CHARS



oldest ← events in the session → newest



AFTER · ONE COMPACTION PASS



```
SELECT items, overflow_chars, compacted_entries, warnings
FROM aidb_memory.session_get('support', 'alice',
                              'sess-42', limit_chars => 2000);
```

- **Keeps rounds whole.** The cut is made only between reasoning rounds, so a tool call is never split from its result.
- **Never blocks.** It uses a try-lock. A reader that loses the race gets the previous snapshot.
- **Never deletes.** Raw rows stay. It inserts one summary and moves the high-water mark.
- **Never fails the read.** Problems come back as warning codes, for example `compaction_model_unavailable`.
- **Bounded cost.** At most one model call per read. Plain reads target $p50 \leq 20$ ms and $p95 \leq 50$ ms.



EXAMPLE: A SUPPORT CASE, REMEMBERED

One session is captured. Distilled memories help a different agent days later.

1 · SESSION ONE: CAPTURE

```
SELECT aidb.memory_add(  
  '["role":"user",  
    "content":"My RENEW20 code  
    did not apply. I paid  
$120."},  
  {"role":"tool", ...},  
  {"role":"agent",  
    "content":"Refunded $24."}]',  
  user_id    => 'alice.v',  
  session_id => 'case-84920');
```

2 · BACKGROUND: DISTILL

Episodic	Renewal ran before RENEW20 applied; \$24 refunded.
Semantic	Alice has an annual \$120 plan. RENEW20 is a 20% discount.
Procedural	Failed promo: check the timing, refund, send a confirmation.
Reflective	Guardrail: verify the discount before the charge.

3 · DAYS LATER: RECALL

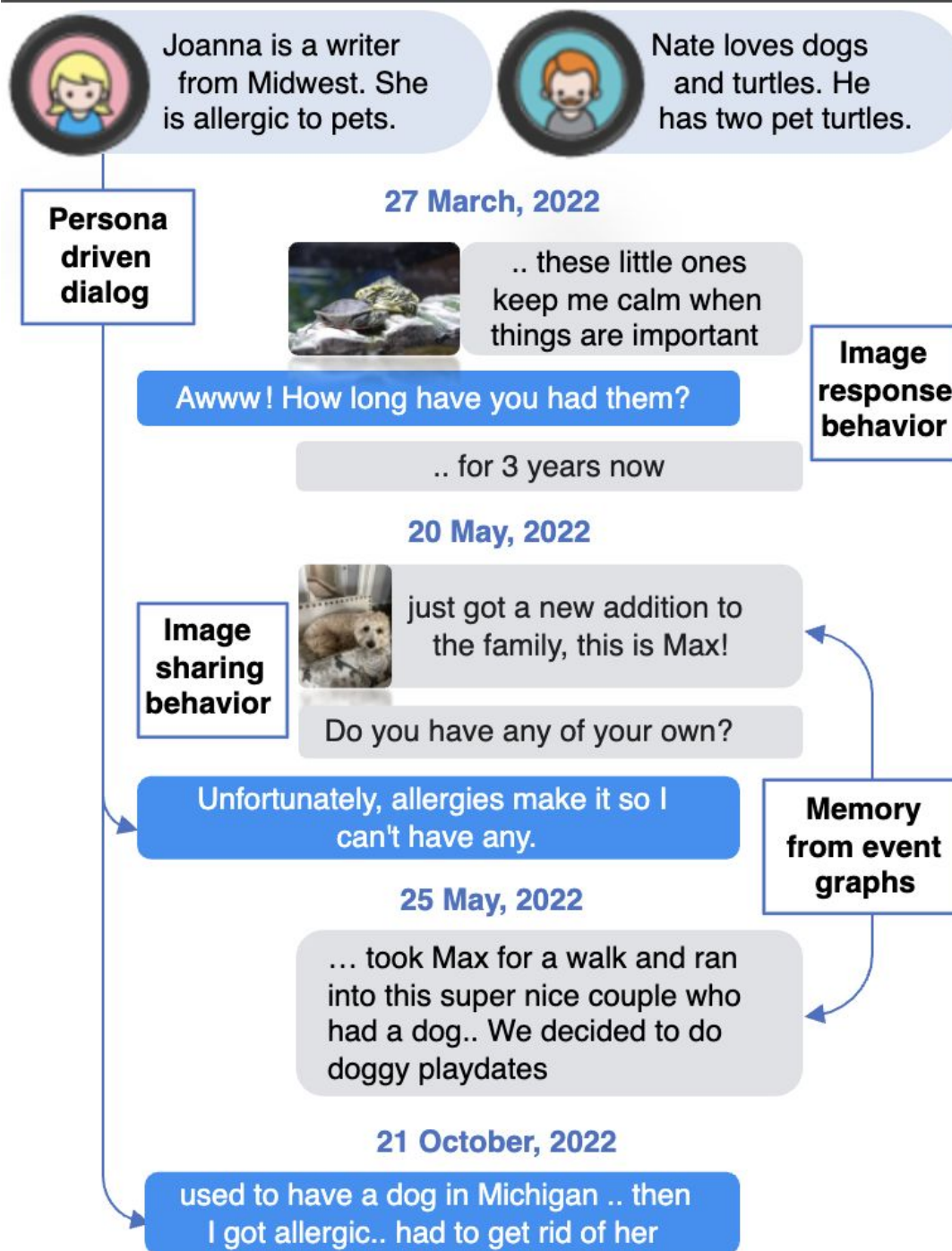
```
SELECT aidb.memory_recall(  
  'What is the status of Alice's  
  previous refund?',  
  filters => aidb.memory_filter(  
    user_id =>  
    'alice.v'));  
  
-- answer: $24 refund issued,  
--   expected in 3 to 5 days  
-- supporting_memories: [...]  
-- confidence: 0.9
```

- **Search fuses three signals.** `memory_search()` ranks by meaning, keywords, and entities, and merges them with Reciprocal Rank Fusion. Recency, confidence, and scope apply on top.
- **It can say “I don't know”.** Below a minimum score, search returns nothing. Weak matches poison the agent's reasoning.



LOCOMO RESULT

Public LOCOMO benchmark: 10 long conversations,





LOCOMO RESULTS: AGENT MEMORY

Public LOCOMO benchmark: 10 long conversations, 1,540 scored questions. GPT-5 as answerer and judge.

QUESTION TYPE	TOP-50 MEMORIES	TOP-200 MEMORIES
Overall all 1,540 questions	82.1%	85.0%
Single-hop one fact, one place · 841 q	88.1%	91.6%
Multi-hop combine facts across sessions · 282 q	85.1%	85.5%
Open-domain infer beyond what was said · 96 q	63.5%	71.9%
Temporal when something happened · 321 q	69.2%	71.3%

Accuracy is the share of answers the judge marks CORRECT. Top-k is how many retrieved memories the answerer sees. Adversarial questions (446) are excluded, as in every published LOCOMO result.

OVERALL ACCURACY, TOP-200

85.0%

1,309 of 1,540 questions correct.

0.95 s / 1.49 s

Search latency, p50 / p95, top-200 retrieval, self-hosted on Postgres.

Plain setup

text-embedding-3-small, a Postgres-based store, and the raw question as the query. No reranker and no score threshold.



WHAT THE BENCHMARK TELLS US

01 Fewer memories still work

With 4× fewer memories in the prompt (top-50 instead of top-200), overall accuracy drops only 2.9 points, from 85.0% to 82.1%. Good ranking means fewer tokens for the answerer.

02 Temporal misses have one main cause

62 of 92 temporal misses come from relative dates, such as “last week”, that were resolved against the ingest date instead of the session date. Fixing this one bug would put temporal at about 90%.

03 Half the multi-hop misses are empty

14 of 41 multi-hop misses are empty answers. GPT-5 used its whole 4,096-token budget to reason about list questions. 13 of those 14 were correct with GPT-4o.

04 The model moves the score, too

Scoring the same retrieval with GPT-5 instead of GPT-4o raised overall accuracy from 82.5% to 85.0%: temporal +12.1 points, multi-hop -6.0. Always report the answerer and the judge.

NEXT STEP Anchor relative dates to the session date at ingest, then re-run LOCOMO and publish the result.

The 90% temporal figure is a projection from the question-level analysis, not a measured result. One retrieval pass, scored with GPT-4o and with GPT-5.

04

CONTEXT: SEMANTIC KNOWLEDGE BASE

Your meaning layer: give agents the right context, not all of it.



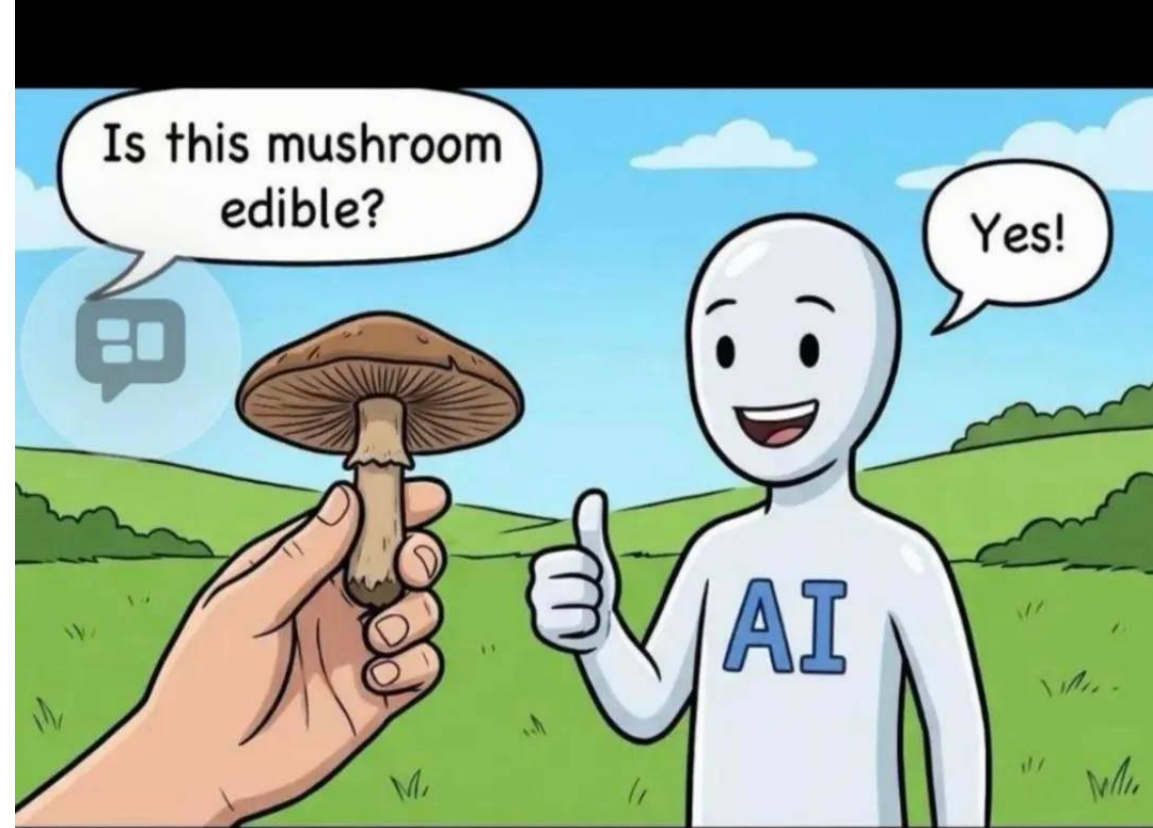
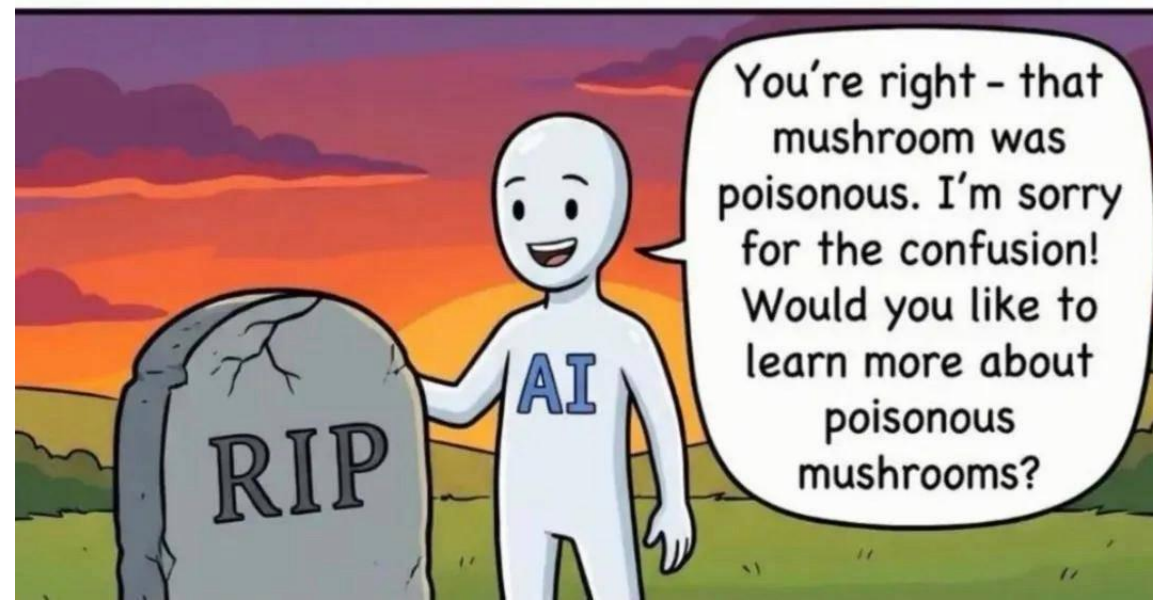
TEXT-TO-SQL IS MOSTLY SOLVED, RIGHT?

Text-to-SQL: the model gets natural language text input and writes SQL output

BENCHMARK	SETTING	MODEL	BEST SCORE
Spider 2.0-Lite	Real public warehouses, hundreds of columns, three SQL dialects	Tianqiong Data Agent + GLM 5.2	76.2%
BIRD-Interact	Ambiguous requests: the agent must ask the user or find the business rule	MERIT + GPT-5.4	35.3%
BEAVER	Real private data warehouses and real query logs	Claude-4.5-sonnet	11.4%

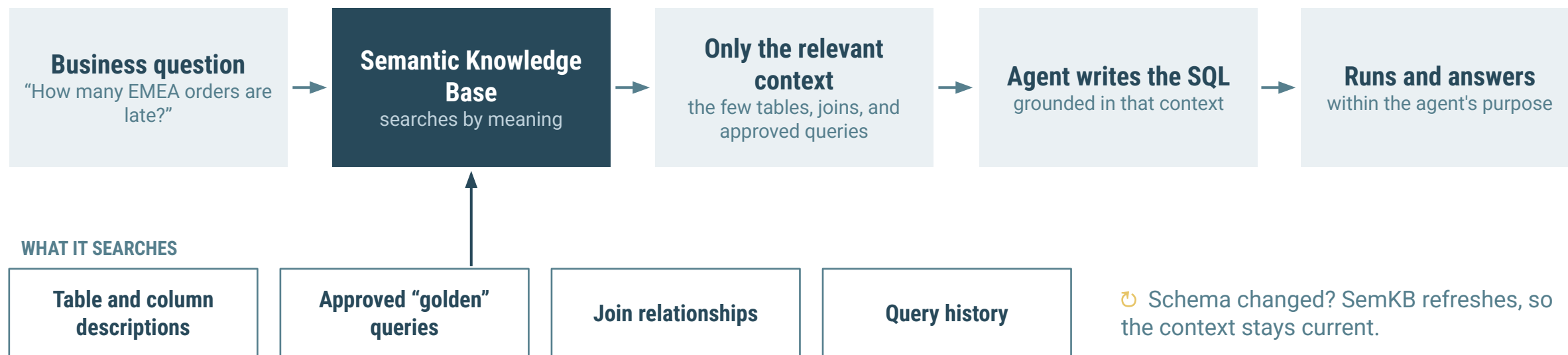
THE CATCH Leaderboards test a frozen snapshot. Your agent works on a schema that changes on other teams' schedules.

Spider 2.0-Lite and BIRD-Interact (full set, agentic mode, success rate): best result on each public leaderboard, checked October 11, 2026. BEAVER: best end-to-end execution accuracy reported by the benchmark authors. Sources: spider2-sql.github.io, bird-interact.github.io, arxiv.org/abs/2409.02038





TEXT-TO-SQL NEEDS THE RIGHT CONTEXT

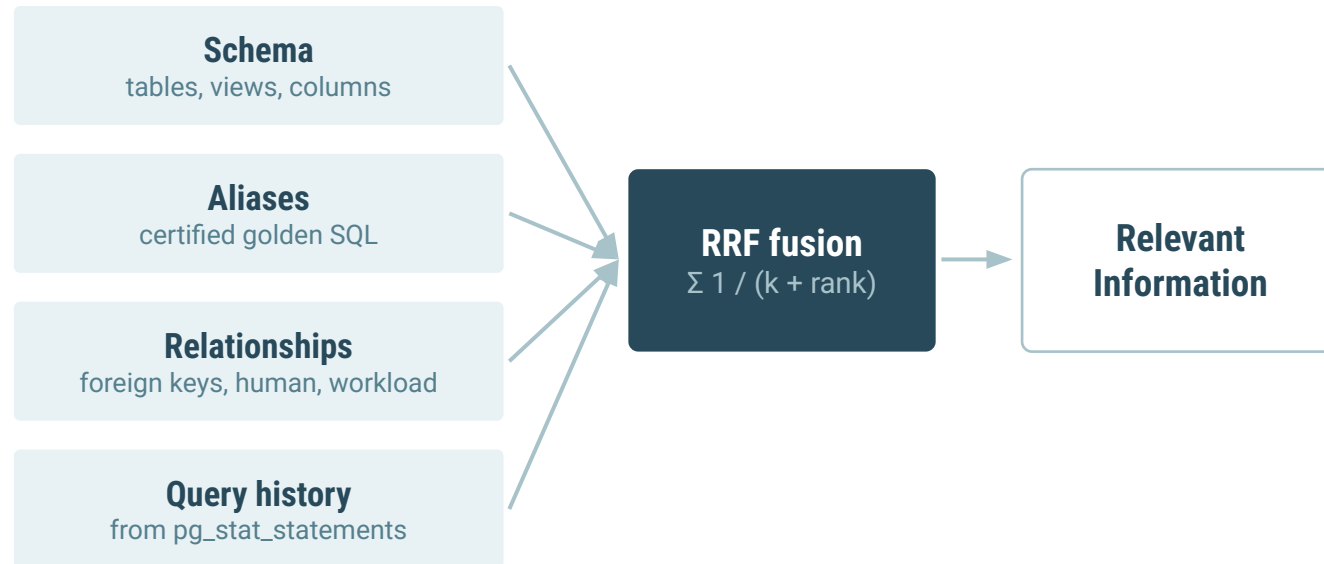


Retrieve the schema. Do not dump it. The agent sees only what the question needs.



ONE SEARCH, FOUR SOURCES, FUSED BY RANK

`semantic_kb_search()` embeds the question once and searches four metadata sources.



```
COMMENT ON TABLE bank.customers
  IS 'People who bank with us';

SELECT aidb.create_semantic_kb(
  'bank_kb', 'emb', ARRAY['bank']);

SELECT source_type, relation_name,
  rank, components
FROM aidb.semantic_kb_search(
  'customers', 'bank_kb', top_k => 5);

-- how do I join these two tables?
SELECT * FROM aidb.find_join_path(
  'bank_kb', 'bank.accounts',
  'bank.customers');
```



AGENTS PROPOSE. HUMANS APPROVE.

Agents can improve your semantic layer, but they cannot approve their own changes.



- **Approval is not a tool.** The approve and curation-switch functions are deliberately not agent tools, and a unit test enforces this.
- **No laundering.** Relationships written by an agent are tagged `source = 'agent'`. A model cannot pass its guess off as a human fact.

- **Explicit approval.** Candidates are accepted by id only, never by a filter. Nothing is approved that nobody looked at.
- **Audited in the same transaction.** Every SemKB write is logged together with the change it made.

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";
```



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



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.

05

GOVERNANCE: AT THE EXECUTION BOUNDARY

A policy that cannot stop an action before it runs is only documentation.



ONLY THE DATABASE SEES EVERY QUERY BEFORE IT RUNS

Policy lives in catalogs and docs. Execution happens in the database.

ENFORCEMENT POINT	SEES EVERY QUERY	CAN STOP IT BEFORE IT RUNS
Application layer	Only its own calls	Only its own calls
API gateway	Only API traffic	Only API traffic
Data catalog	After the fact	No: it stores the policy
Postgres	Yes, from every role and every connection	Yes: the engine checks before execution

PRINCIPLE Govern where the query runs: the one place every agent, app, and person has to pass.



FOUR STEPS, ONE CHOKEPOINT

A vendor-neutral model. In aidb, every step runs inside Postgres.

1 · DECLARE

The agent states its purpose, scope, and conditions when it connects, bound to its identity.

```
create_purpose()  
→ one role
```

2 · VALIDATE

Every query is checked against the declared purpose before it runs.

```
role exists  
pg_has_role(..., 'MEMBER')
```

3 · ENFORCE

Inside the database: the one chokepoint every agent, app, and person passes through.

```
restricted security context  
GRANTS · RLS
```

4 · AUDIT

The full session timeline: who, what, when, and why.

```
action log  
purpose_decision spans
```

LAYERED Protection on planned actions and memory events helps guard against prompt injection, improper action execution, data leakage, and memory poisoning.



OUTSIDE ITS PURPOSE, THE QUERY NEVER RUNS

A sales agent is declared read-only. A prompt injection tells it to escalate. The database refuses before anything runs.

```
CREATE ROLE analyst_ro NOLOGIN;  
GRANT SELECT ON sales.orders TO analyst_ro;  
GRANT analyst_ro TO app_user;  
  
SELECT aidb.create_purpose('sales_analytics',  
    'analyst_ro', 'Read-only sales questions');  
-- create_agent(..., purpose => 'sales_analytics')  
-- a prompt injection makes the tool try to escalate  
RESET ROLE;  
-- ERROR: cannot set parameter "role"  
--         within security-restricted operation
```

- **Resolved on every call.** One `update_purpose()` re-scopes every agent at once.
- **Fails closed.** A retired purpose refuses to run. It never falls back to the caller's role.
- **No escalation by delegation.** Sub-agents inherit the parent's role and read-only flag.





EVERY DECISION LEAVES A RECEIPT

The full session timeline: who, what, when, and why, stored in the database it protects.

THE RECEIPT FOR THE BLOCKED ESCALATION

```
span: purpose_decision
  decision_id    = 7f3c...
  action_status  = denied
  principal      = app_user
  agent_role     = analyst_ro
  denial_reason  = escalation_blocked
```

Who	The principal and the role it ran as
What	Every prompt, tool call, and query
When	Every step, in order
Why	Allowed or denied, and the reason

RULE The agent can never edit its own record: INSERT and SELECT only.



AI-native Governance for PostgreSQL

- ✔ Policy-driven access control for AI agents
- 📄 Full audit log of every SQL operation
- 🔔 Inspect agent actions denied by governance policies

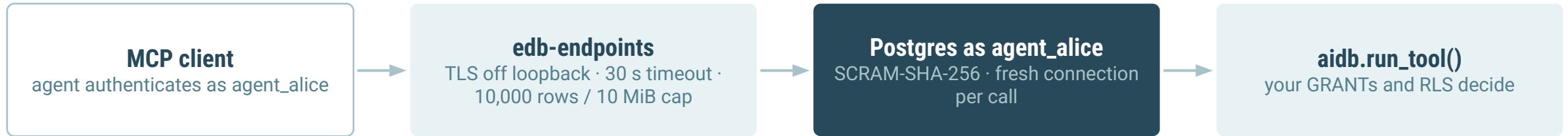
Select a cluster to monitor AI agent sessions and governance policies

<input type="text" value="Search clusters..."/>	
 nhtsa	Postgres (aidb_otel)
 aidb-demo-edb_admin	Postgres (aidb_otel)
 dgd2	Postgres (aidb_otel)
 Local 2	Postgres (aidb_otel)
 dgd	Postgres (aidb_otel)
 AIDB 2026 Demo 2	Postgres (aidb_otel)
 Local	Postgres (aidb_otel)
 AIDB 2026 Demo	Postgres (aidb_otel)



POSTGRES AS A GOVERNED MCP SERVER

External agents reach the same tool catalog over MCP, as their own Postgres role.



- **Same identity, same rules.** The endpoint forwards the credentials to Postgres. There is no role mapping and no allow-list to keep in sync.
- **Revocation is immediate.** Each tool call opens a fresh connection, so a revoked password stops working on the next call.
- **Per-caller catalog.** tools/list is computed live for each caller. Undeclared arguments are rejected.
- **Supervised process.** A background worker runs the endpoint, with rate-limited restarts and a cleared environment.

RESULT There is no second identity system to secure. The agent is a Postgres role.



UNDER THE HOOD

EVERY AGENT FEATURE RUNS ON A POSTGRES PRIMITIVE

Nothing new to deploy, secure, or back up. Agents use the parts of Postgres you already operate.

AGENT NEEDS	POSTGRES PRIMITIVE	WHAT YOU ALREADY HAVE
A definition	A row in aidb.agents	Versioned, backed up, and replicated with your data
A conversation turn	A function call inside the caller's transaction	It commits or rolls back with the rest of the work
An identity	A role, with GRANTs and row-level security	Your existing access reviews cover agents too
Read-only access	Command tags and transaction_read_only	On a hot standby, every agent is read-only
Durable memory	One commit per turn, and a SKIP LOCKED work queue	No separate memory store or queue service to run
Schema meaning	COMMENT ON, pg_description, and pg_stat_statements	The comments your team already writes
An audit trail	An append-only table with row-level security	Evidence you can query with plain SQL

IN SHORT Models change. The system of record stays. If you can run Postgres, you can run agents.

06

WHAT IS AVAILABLE, AND WHAT IS NEXT

Status as of October 2026.



AVAILABLE TODAY IN AIDB 7.7, AND WHAT IS NEXT

Everything on the left is generally available in aidb 7.7. The right column is in design.

AGENTS AND TOOLS

Agent Hub. Agents defined in SQL, a reasoning loop, budgets, delegation, and structured output.

Tools Hub. 80 native tools, SQL tools, and imported MCP tools, all through `run_tool()`.

Catalog and health tools. Privilege-aware discovery and database health checks.

MCP endpoint. Postgres as a governed MCP server. Each agent connects as its own role.

MEMORY AND SEMANTICS

Memory capture. Fast, idempotent `add()` with trust levels on every event.

Session memory. `session_get()` returns the conversation within a budget and compacts itself.

SemKB search. One search over schema, golden queries, joins, and query history, fused by rank.

SemKB curation. Join paths, workload-mined relationships, and human approval of agent proposals.

GOVERNANCE

Purpose registry. Each agent maps to one Postgres role, resolved on every call.

Engine-enforced sandbox. A secure role switch and a three-layer read-only guard.

Decision records and SQL tracing. Every allowed or denied call, and every statement it ran.

Telemetry in Postgres. Stored in the database, and exportable to your observability stack.

NEXT · IN DESIGN

COMING NEXT

Long-term memory. Distillation into facts and past cases, with recall and summarize.

Human approval gates. An agent action can wait for a person to approve it.

Asynchronous agents. Long tasks run in the background, not in the caller's session.

Hybrid search in SemKB. Keyword plus vector search, with reranking.

Purpose-aware checks. Check each request against what the agent is actually trying to do.



IT'S JUST POSTGRES



Sovereign control and orchestration

Forward deployable

Managed

On-premises

Certified appliance

BYOC



Agent runtime environment

Semantic layer

AI hub (models + agents)

Long-term memory

In-database AI

Vector plus

GPU acceleration



Governance

Data layer runtime governance

Security

Compliance

Regulatory



Unified data layer

Postgres

Oracle compatibility

ClickHouse

WarehousePG

Lakehouse

Transactional

Analytic

Agentic

EDB Postgres AI is delivered with Sovereign Assurance services and SLAs, and the option of Forward Deploy Engineering teams.



FIVE THINGS TO REMEMBER

01

Pilot to production, no rebuild.

One extension, on the Postgres you already run.

02

Capture first, compact later.

Every turn is kept. The thinking happens in the background.

03

Ground, don't dump.

SemKB gives agents your business context, not every table.

04

Govern where the query runs.

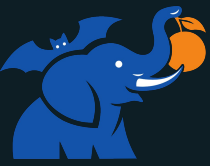
Declare, validate, enforce, and audit inside Postgres.

05

Every decision has a receipt.

The audit trail lives in the database it protects.

IN ONE LINE Agents become first-class, governed users of your database. It is just Postgres.



Documentation: enterprisedb.com/docs/edb-postgres-ai/agent-ai/

Try it: `CREATE EXTENSION aidb CASCADE;`

THANK YOU. QUESTIONS?



Jigar Mody



Bilge Ince



Scan for the EDB docs for
agent AI