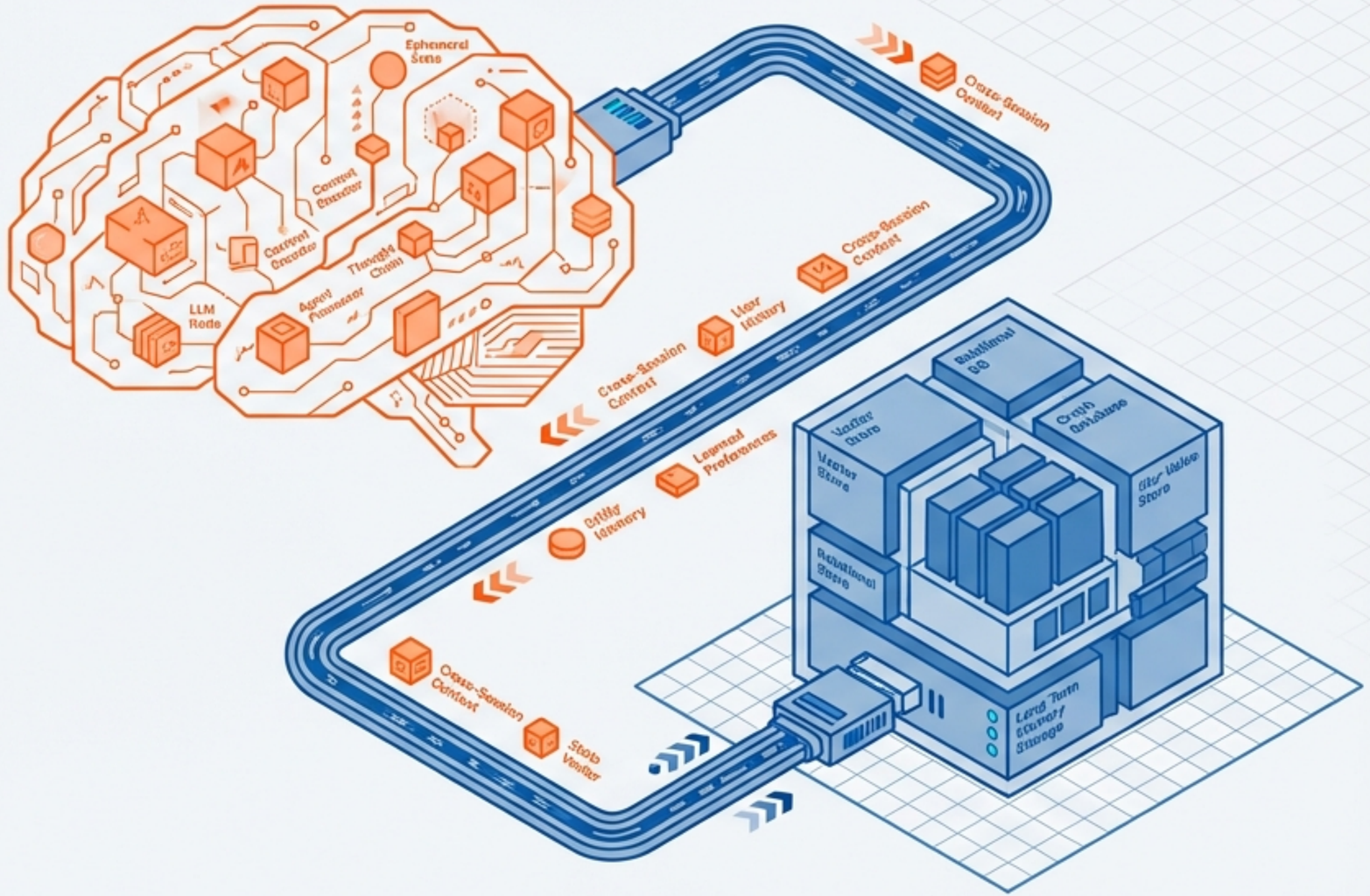
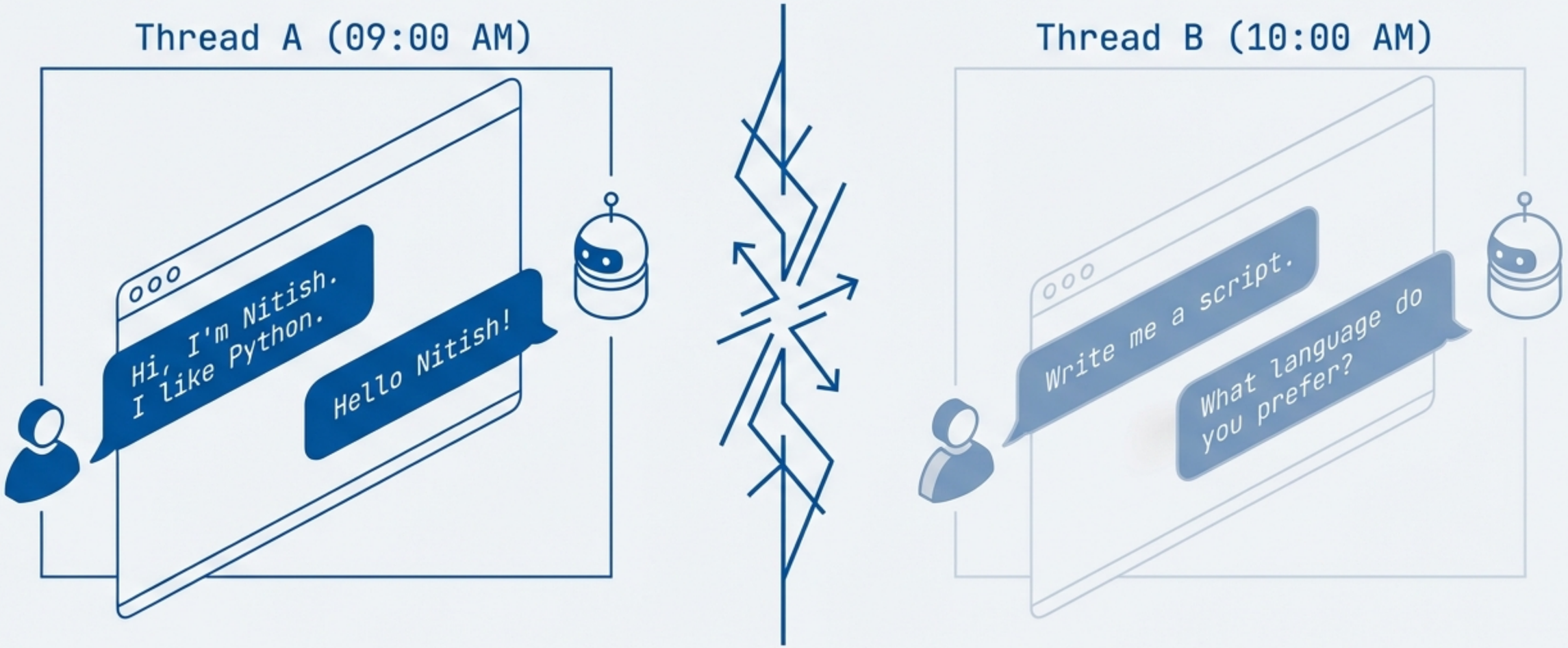
Beyond the Thread: Architecting Long Term Memory

Implementing persistent, cross-session context in LangGraph agents.



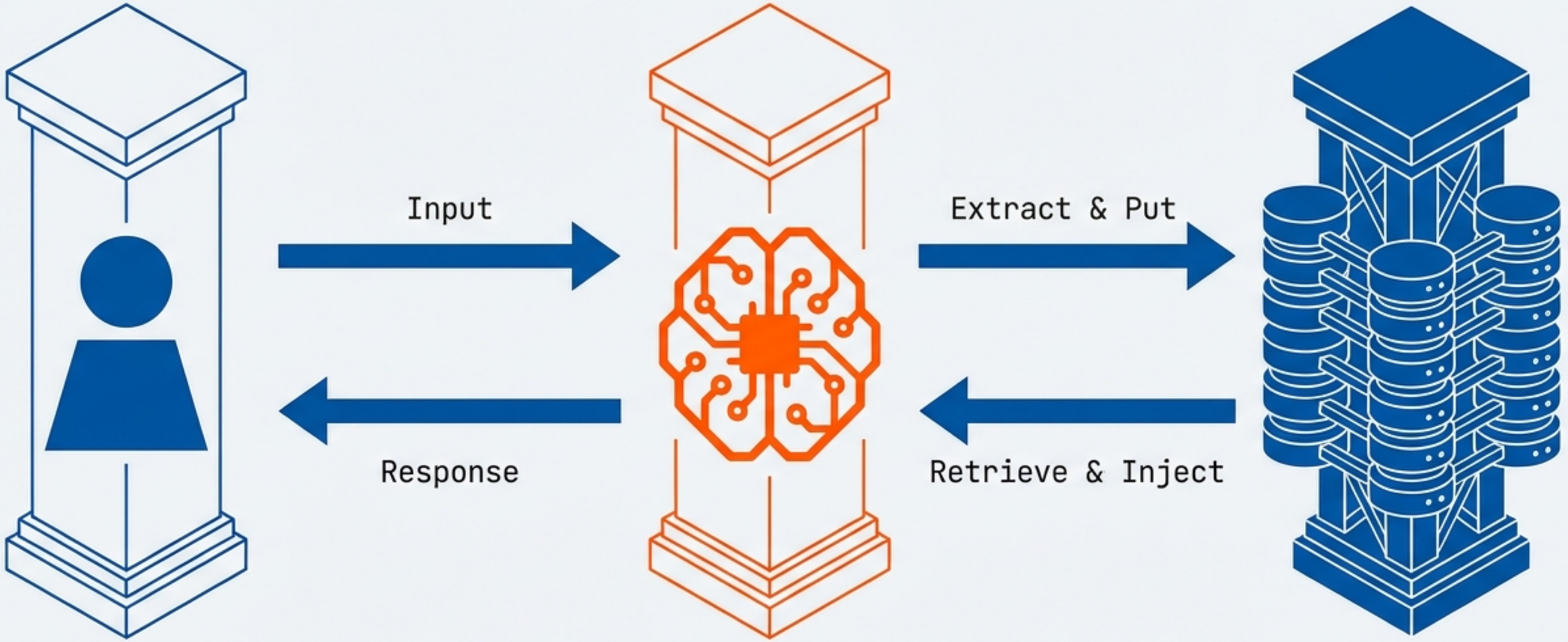
Based on the technical implementation by CampusX

The Problem of Ephemeral Context

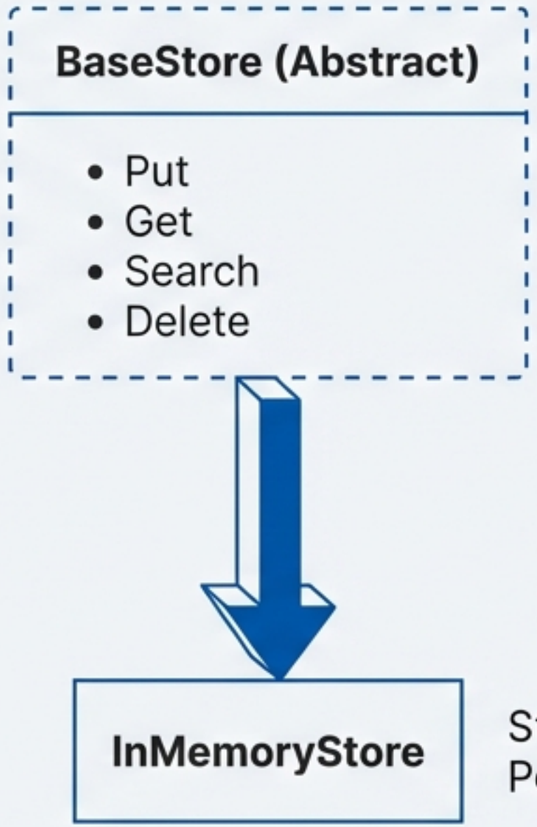


LLM context is thread-scoped. Closing the window resets the relationship.

The Third Pillar of Architecture



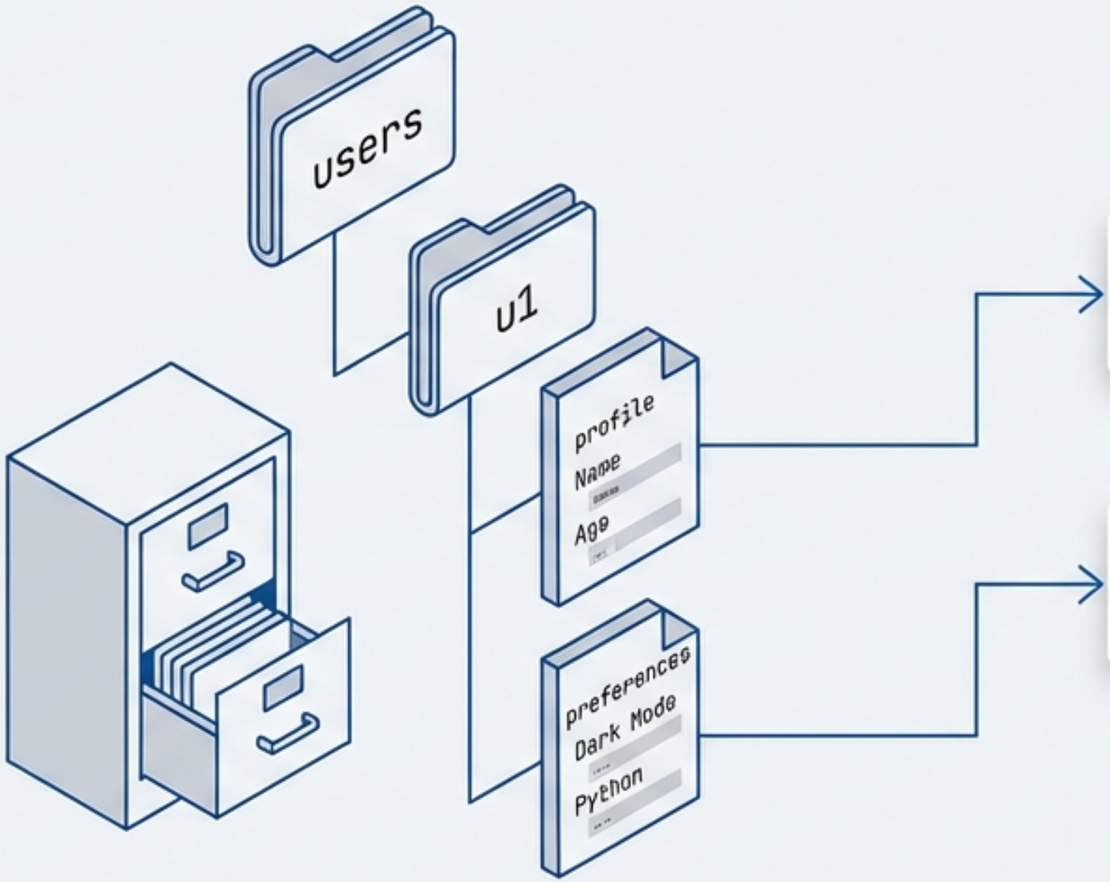
The BaseStore Abstraction



Stores data in RAM.
Perfect for prototyping.

```
from langgraph.store.memory import InMemoryStore

store = InMemoryStore()
```

Organizing Chaos with Namespaces




```
namespace = ('users', 'u1', 'profile')
```

```
namespace = ('users', 'u1', 'preferences')
```

Namespaces are tuples that act as directory paths for memory isolation.

The CRUD Operations

PUT
(Write)

```
store.put(namespace, key='1',
                value ={'data': 'Likes Pizza'})
```

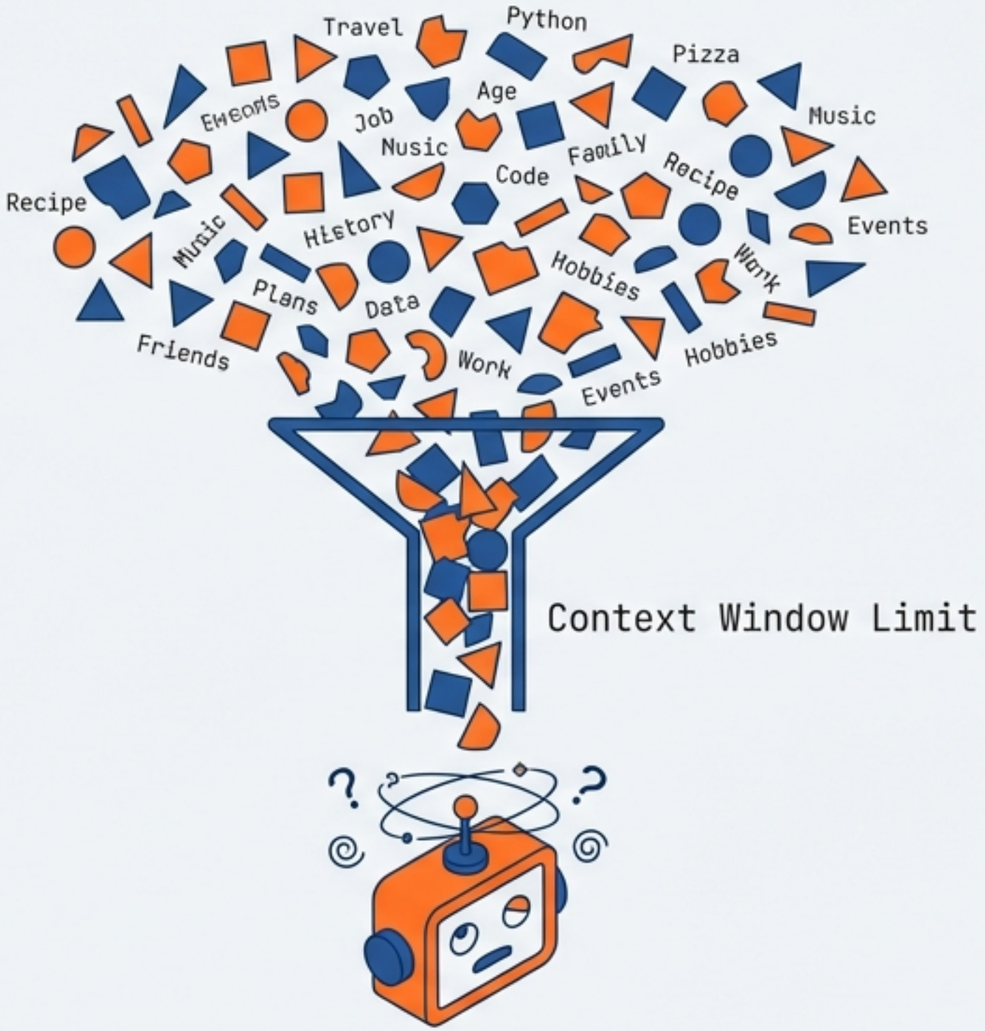
GET
(Read Exact)

```
store.get(namespace, key='1')
```

SEARCH
(Browse)

```
store.search(namespace)
```

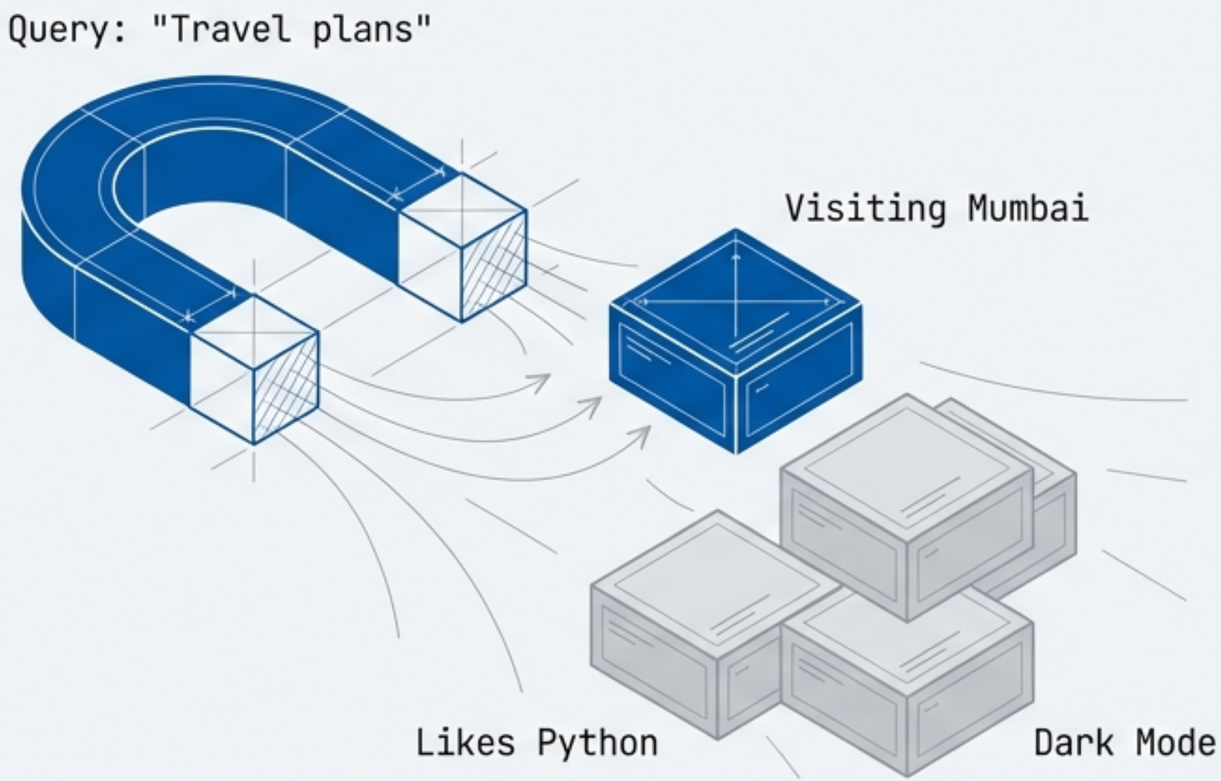
The Retrieval Bottleneck



Scenario:
Query: "What is my travel plan?"

Insight:
Using store.search() dumps everything.
We need a filter for meaning.

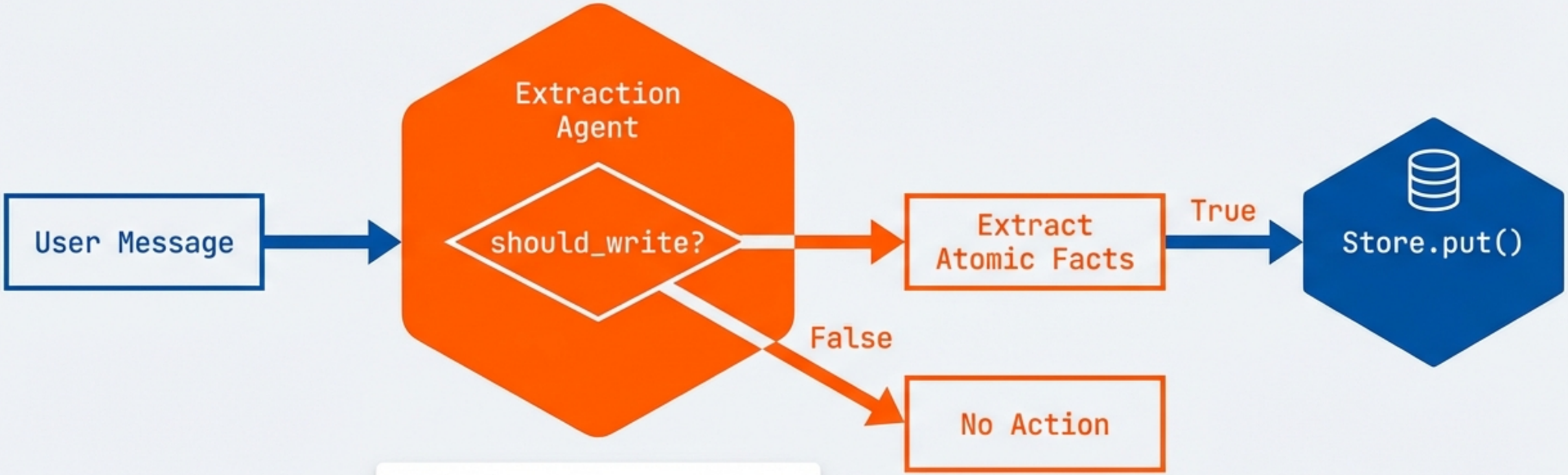
Enabling Semantic Search




```
store.search(
  namespace,
  query='What is user currently learning?',
  limit=1
)
```

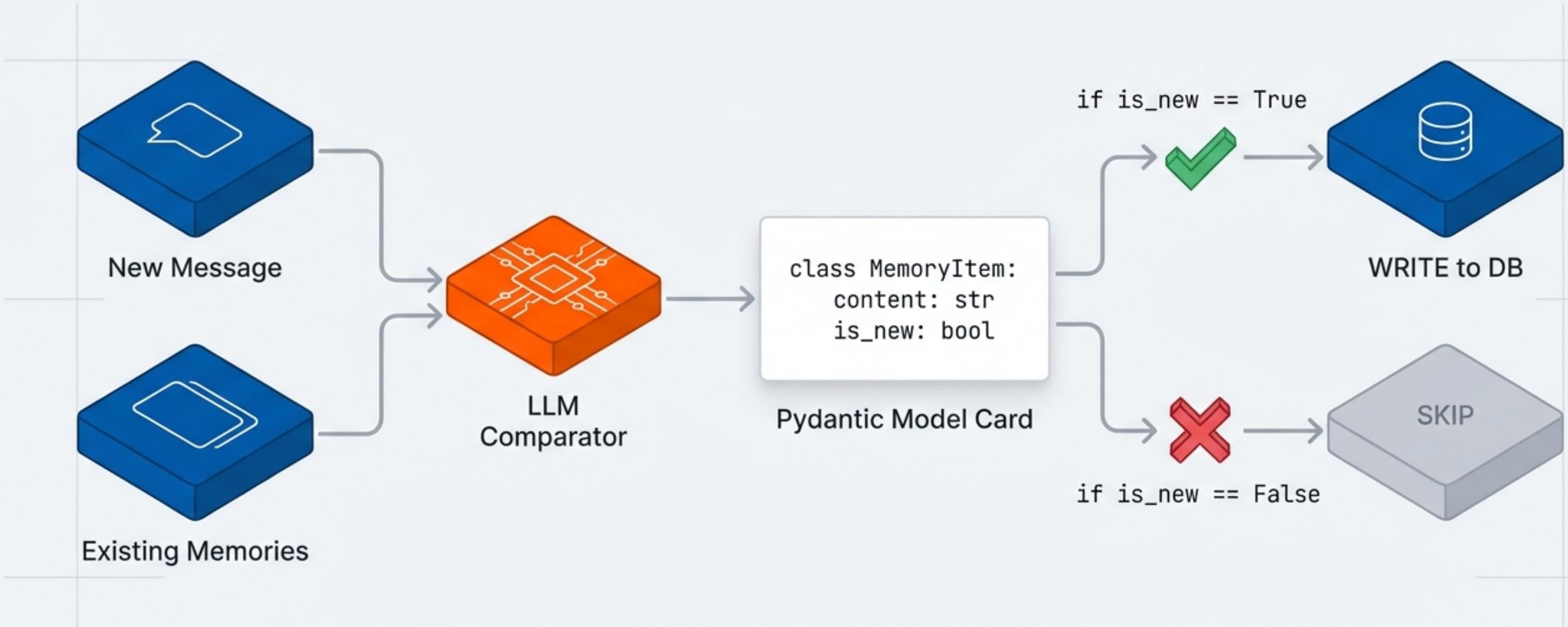
Vector embeddings allow retrieval by meaning, not just keywords.

The Extraction Workflow



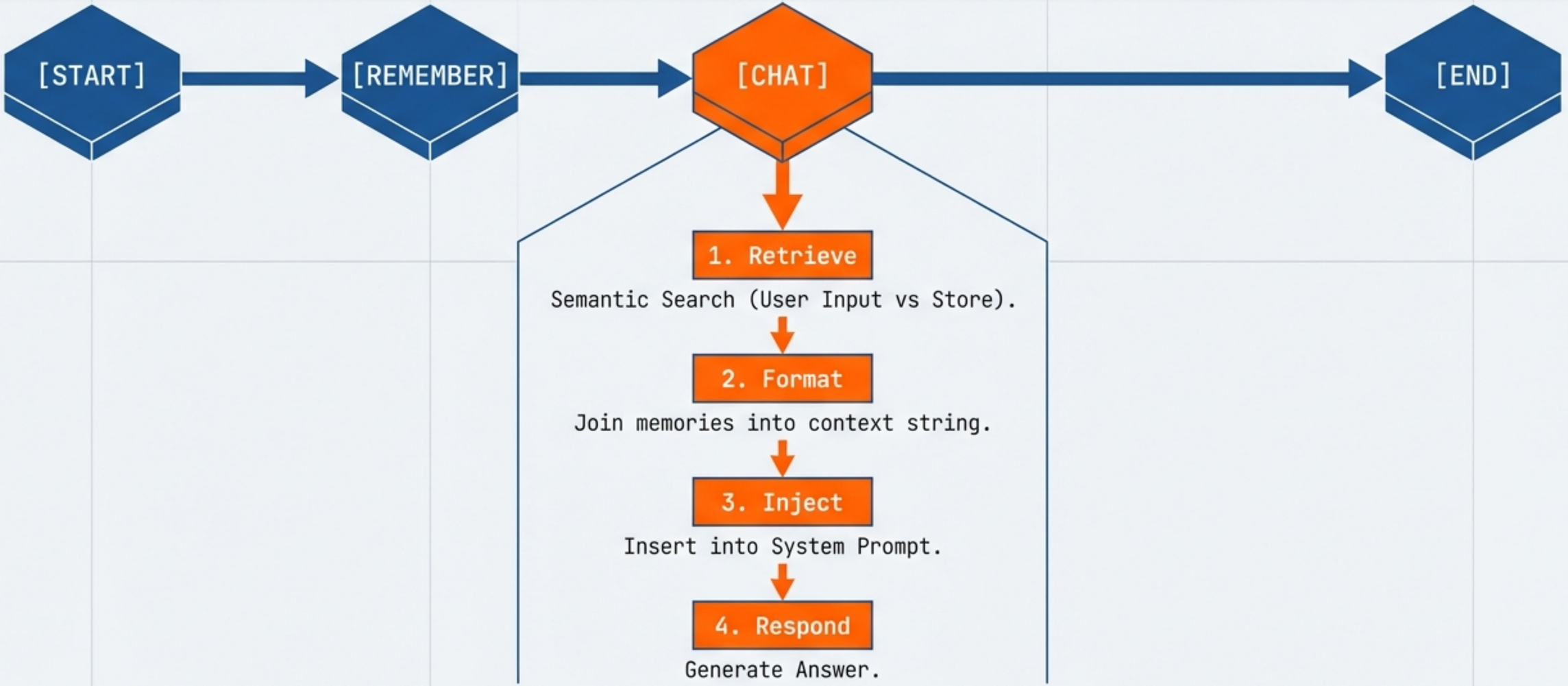

```
Input: "I teach AI"
-> should_write: True
-> Fact: 'User teaches AI'.
```

Solving De-Duplication



Prevents storing redundant facts like 'My name is Nitish' multiple times.

The Injection & Chat Workflow



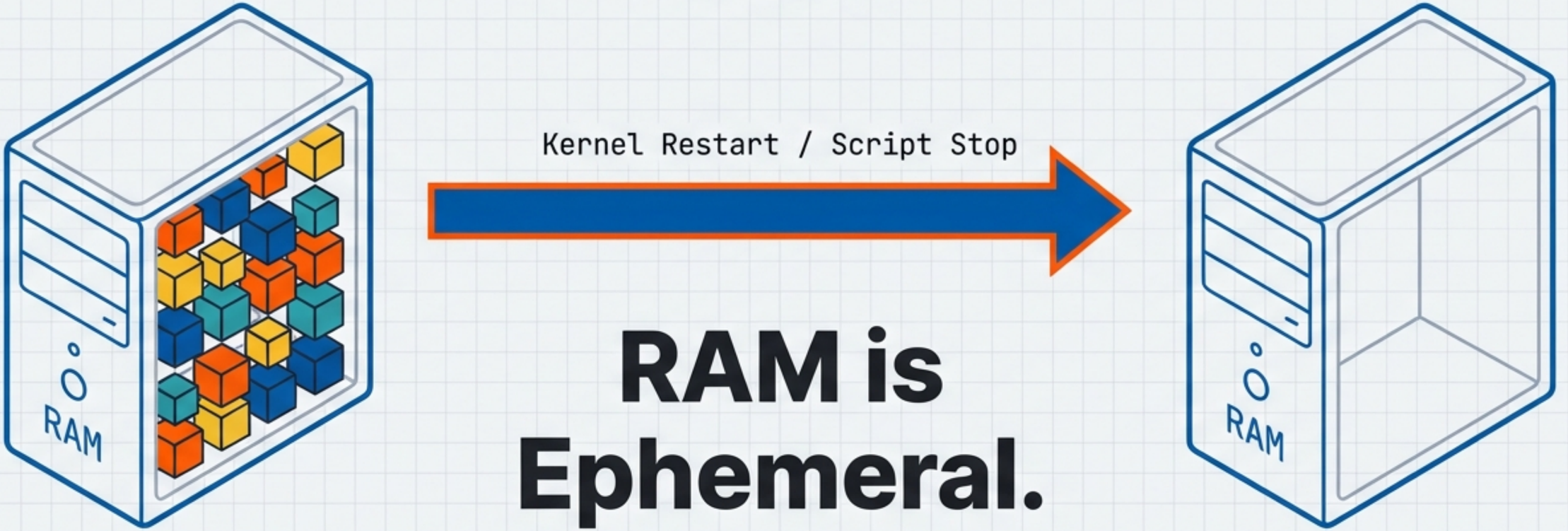
Engineering the System Prompt

```
You are a helpful assistant with memory
capabilities.

Address user by name.

Reference known projects/preferences (e.g.,
'Since you use TypeScript...').

Adjust tone to be friendly and natural.

Suggest 3 relevant follow-up questions
based on profile.
```

Forces the LLM to use the injected context.

The Volatility of InMemoryStore

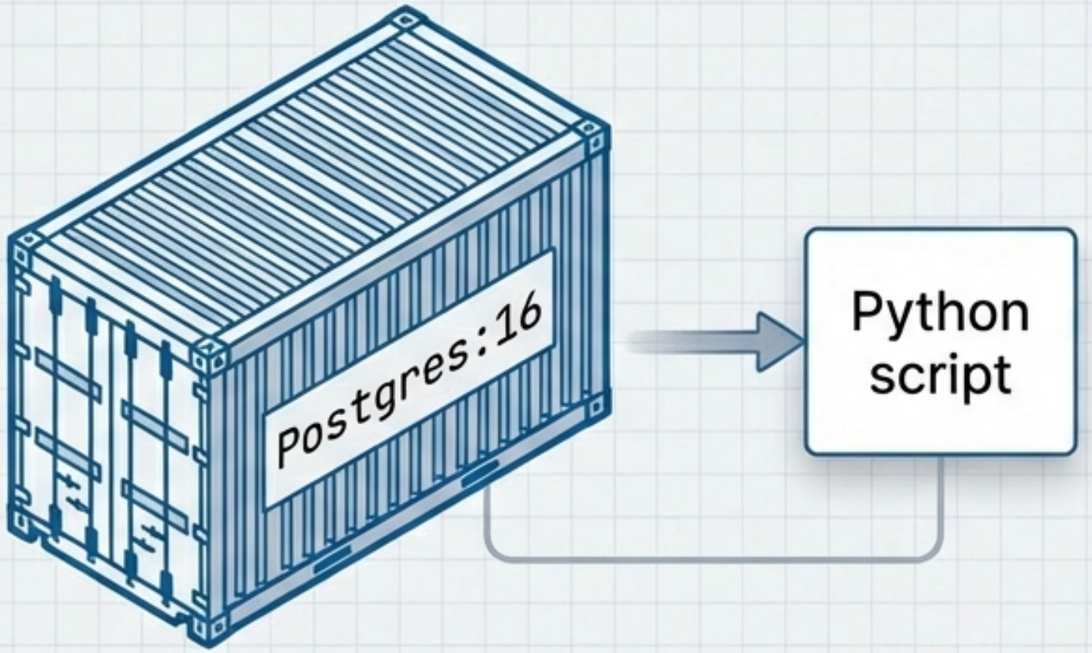
Reboot



RAM is Ephemeral.

In-Memory prototyping is fast, but data vanishes on restart. Production requires persistence.

Going Pro: The Postgres Store

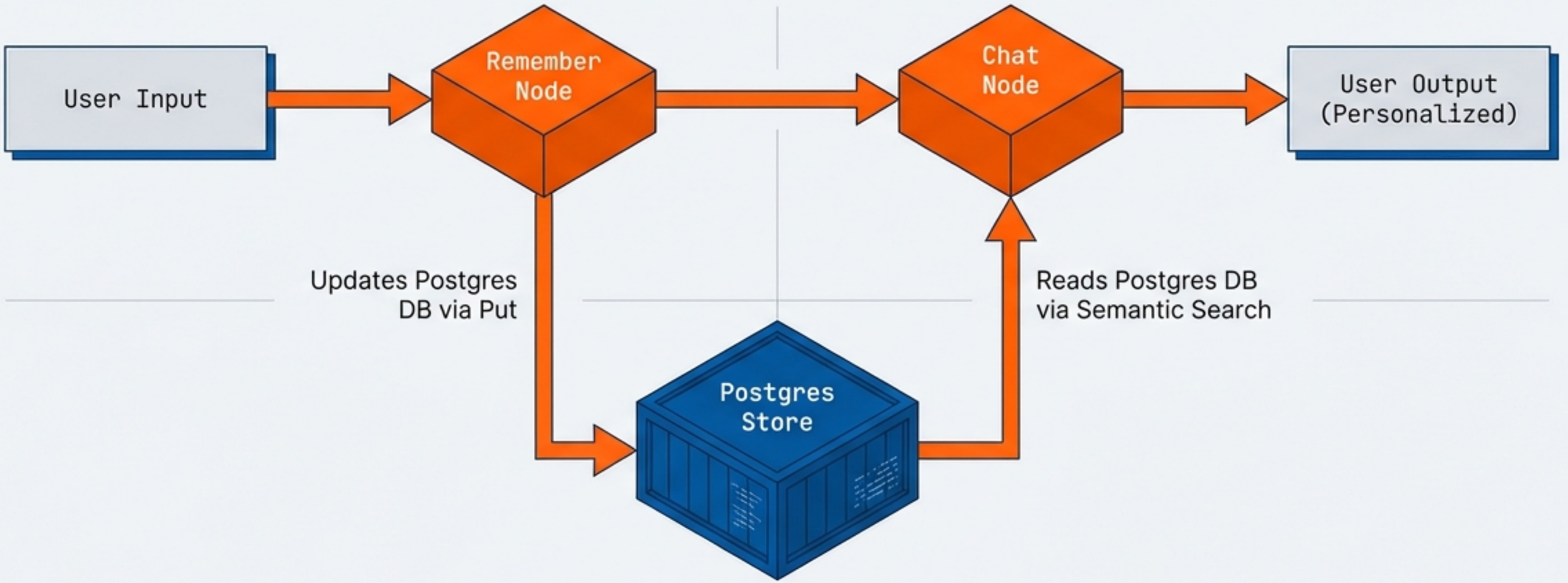


CODE SNIPPET

```
- store = InMemoryStore()
+ with
      PostgresStore.from_conn_string
      (DB_URL) as store:
```

The Graph logic remains unchanged. Only the backend is swapped.

The Complete Persistent Agent



CHAT EXAMPLE

User: Explain GenAI.

Bot: Sure Nitish, since you like Python, here is a code example...