



Architecting Business Operational Context for AI systems and agents

In the agentic era, the cycle between data and action is collapsing from days to seconds.

What cloud was to applications,
Business Operational Context
will be to agentic systems.

The data foundation for AI execution,
validation, and continuous improvement



What is Business Operational Context

- The business-ready context AI systems need to understand situations and act correctly, every time, in real time.



Precise

Delivers only the exact context required, minimizing token cost, privacy exposure, and hallucinations



Real time

Composed and synced in real time from source systems, for low-latency MCP and A2A access to current business state



Governed

Quality, privacy, and isolation applied dynamically based on task-specific guardrails



Transactional

Supports both read access and write-back across operational workflows

Why the **business entity** is the unit of context

Autonomous systems need a unified, task-relevant unit of context.
That unit is typically a business object such as a customer, order, loan, or employee.

Scope

The unit of context required for the task at hand



Example entity: Customer

Isolation

A contained unit of context that can be governed independently

Actionability

The unit that can be accessed and acted on as one

The business entity is the unit of operational action.
Context must be organized around it to enable access and action.

Why **current architectures** can't deliver Business Operational Context



Data lakes

Designed for analytics, not real-time, transactional context delivery

APIs / MCP

Built for predefined access, not dynamic context assembly

Vector DBs

Optimized for knowledge retrieval, not for complete, real-time context

Business Operational Context must be:

- 1 **Precise:** Exact, dynamically assembled for the task
- 2 **Real-time:** Context composed and delivered from live sources
- 3 **Governed:** Runtime control, privacy, and isolation
- 4 **Transactional:** Read and write-back across workflows

From enterprise data to **entity-scoped** operational context

k2view

Cases

Contracts

Customers

Invoices

Payments

20M

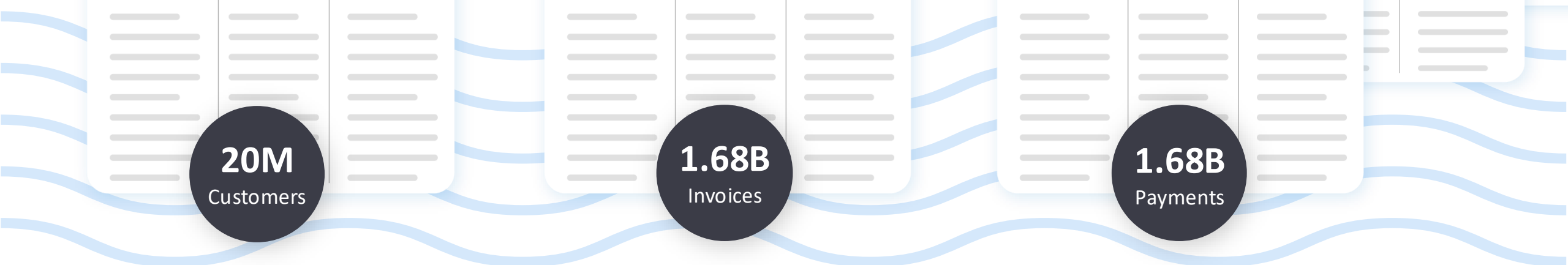
Customers

1.68B

Invoices

1.68B

Payments



From enterprise data to **entity-scoped** operational context

Customers

Invoices

Payments

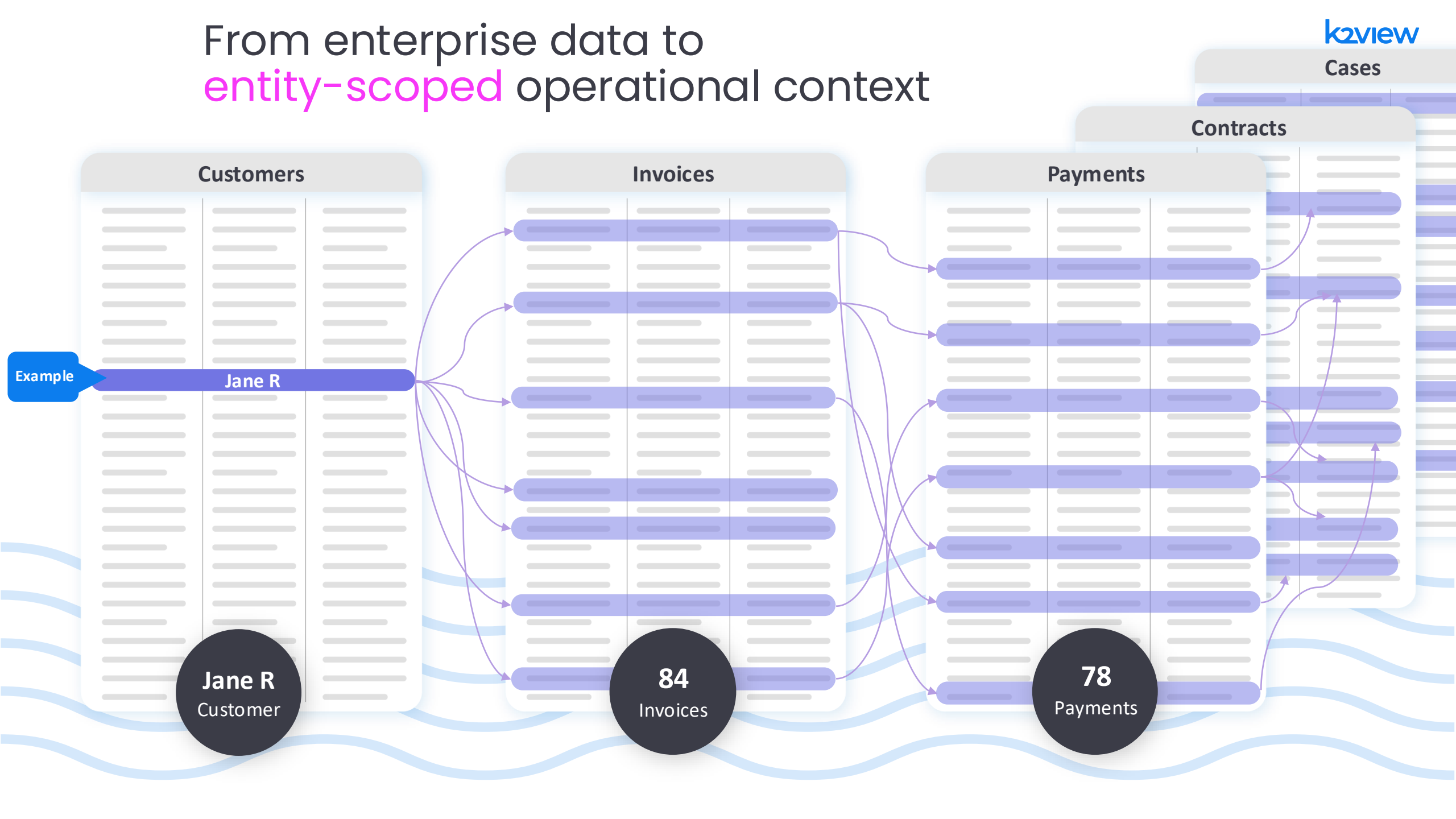
Example

Jane R

Jane R
Customer

84
Invoices

78
Payments



One entity's operational context,
stored in its own **Micro-Database™**

1 Micro-Database per entity instance

Precise.
Isolated.
Synced.

Jane R
Customer



Customer details: Jane R | SSN: 688-03-7099 | Email: jrobbins@abc.com | Phone: 770-352-4452

Invoices

84

Invoices

Payments

78

Payments

Contracts

5

Contracts

Cases

22

Cases

84

Statements

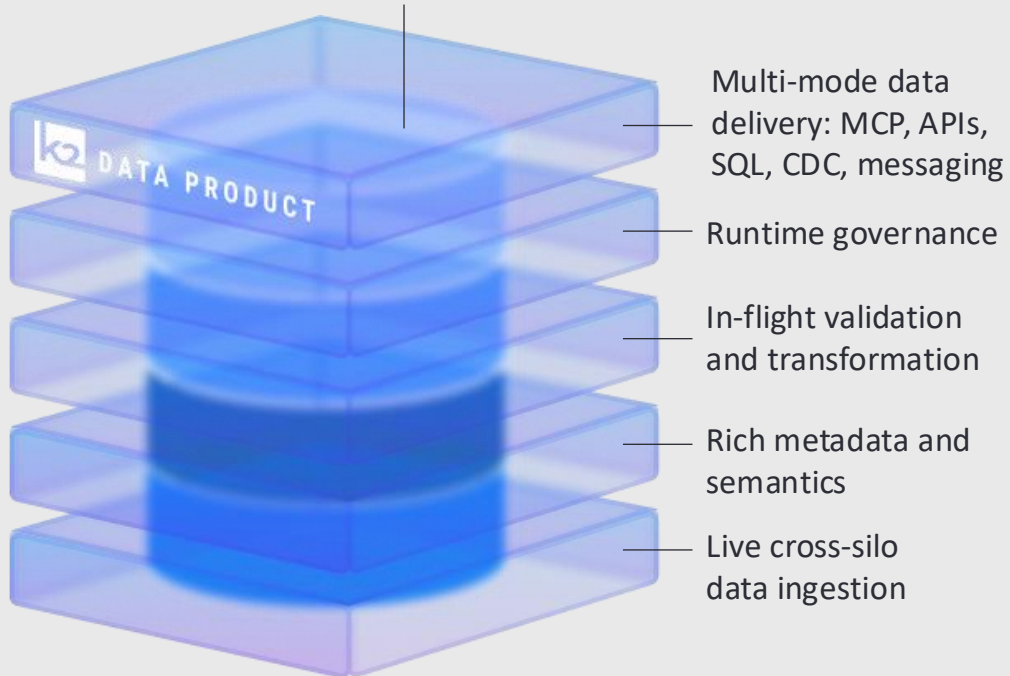
78

Payments

K2view entity-centric data products compose and deliver entity-scoped context

Data product instance

Live operational context
for a business entity instance stored
in its own Micro-Database™

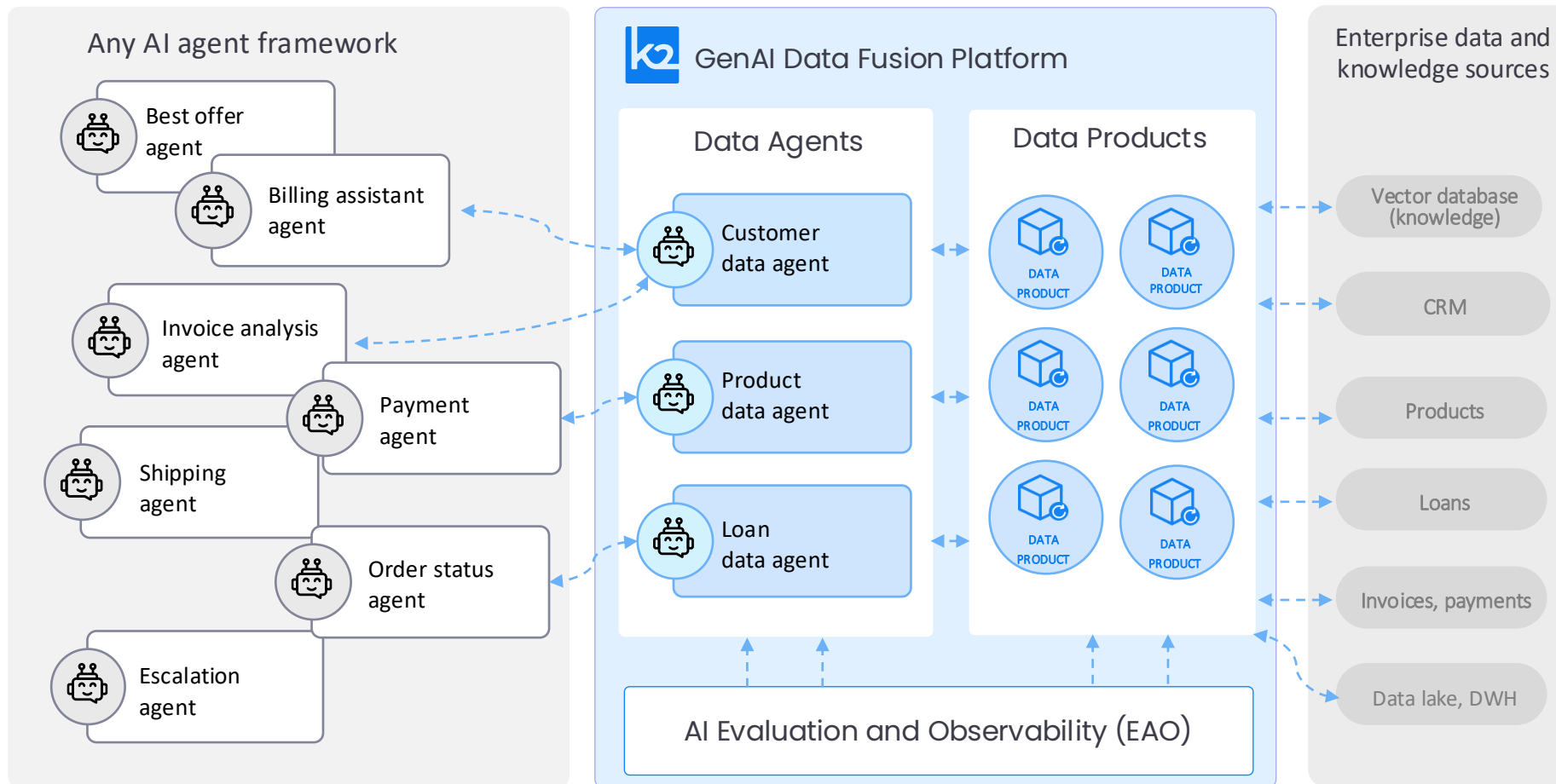


Architectural characteristics

1. Real-time, end-to-end
2. Precise context assembly and delivery for minimal cost and accurate reasoning
3. Entity-level isolation, governance, and inflight validation for quality, security and privacy
4. Entity-level caching and synchronization for low latency, cost-controlled scale

How K2view enables agentic operations at scale

Decoupling AI agents from data handling for reuse, control, and scale



Data agents

Interpret intent, determine needed context, orchestrate data product interactions

- Decouple context handling from AI agents
- Standardize data product interactions across AI agents
- Minimize token usage

Data products

Compose and deliver entity-centric operational context for agentic execution

- Unified across systems
- Governed at runtime
- Transactional by design

AI Eval & Observability

Test and evaluate data agents to improve trust, grounding, and outcomes

- Test agents before production
- Assess context grounding
- Learn from runtime outcomes

How **Cellcom** transformed call center operations with GenAI and enterprise data

Customer operational context, assembled in real time, for the CSR to service the customer



Recent activity and risk

- 3 support calls last week (related to fiber issues)
- Escalation and churn risk signal in last interaction



Customer value and relationship

- 10-year customer
- Mobile, TV, and Internet services
- VIP in the last 6 months



Billing changes

- Last monthly charge +\$18
- Due to international calls and new service



Financial Health

- All payments on time



Improved core customer service KPIs



- Shorter call handling time (10-30%)
- Higher first contact resolution (10-20%)
- Faster rep ramp-up (20%-40%)
- More consistent service quality

Business operational context (not LLMs) determines **operational AI success**

- You're no longer "supporting AI", you're defining the data architecture it depends on
- Traditional data architectures weren't built for real-time, governed Business Operational Context
- The foundation for Business Operational Context must be built now, before it becomes the bottleneck



K2view delivers Business Operational Context at production scale, with the required latency, and at controlled cost

k2view

Thank you!

The background of the slide features a dark blue gradient. In the lower half, there are several dynamic, glowing light streaks in shades of blue and purple, curving upwards from left to right, creating a sense of motion and energy.

k2view

Traditional Architectures



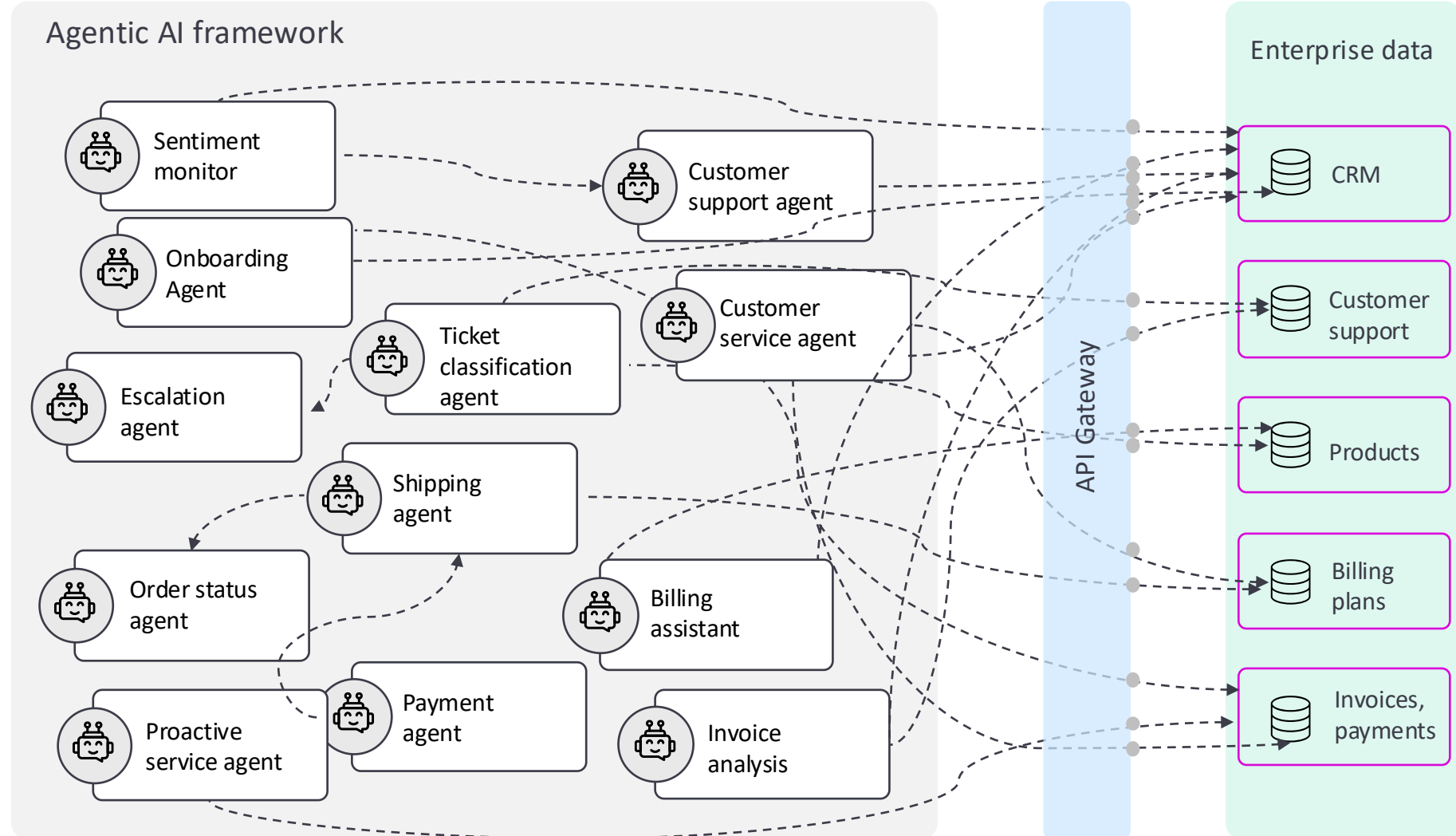
Approach 1: APIs

✦ **Architectural gap:** Built for known and finite questions

! Result:

API sprawl and agent spaghetti

- API sprawl that's hard to govern and evolve
- Fragile architecture that's hard to scale to new use cases
- Unpredictable load on operational systems
- Inconsistent enforcement of privacy and security policies



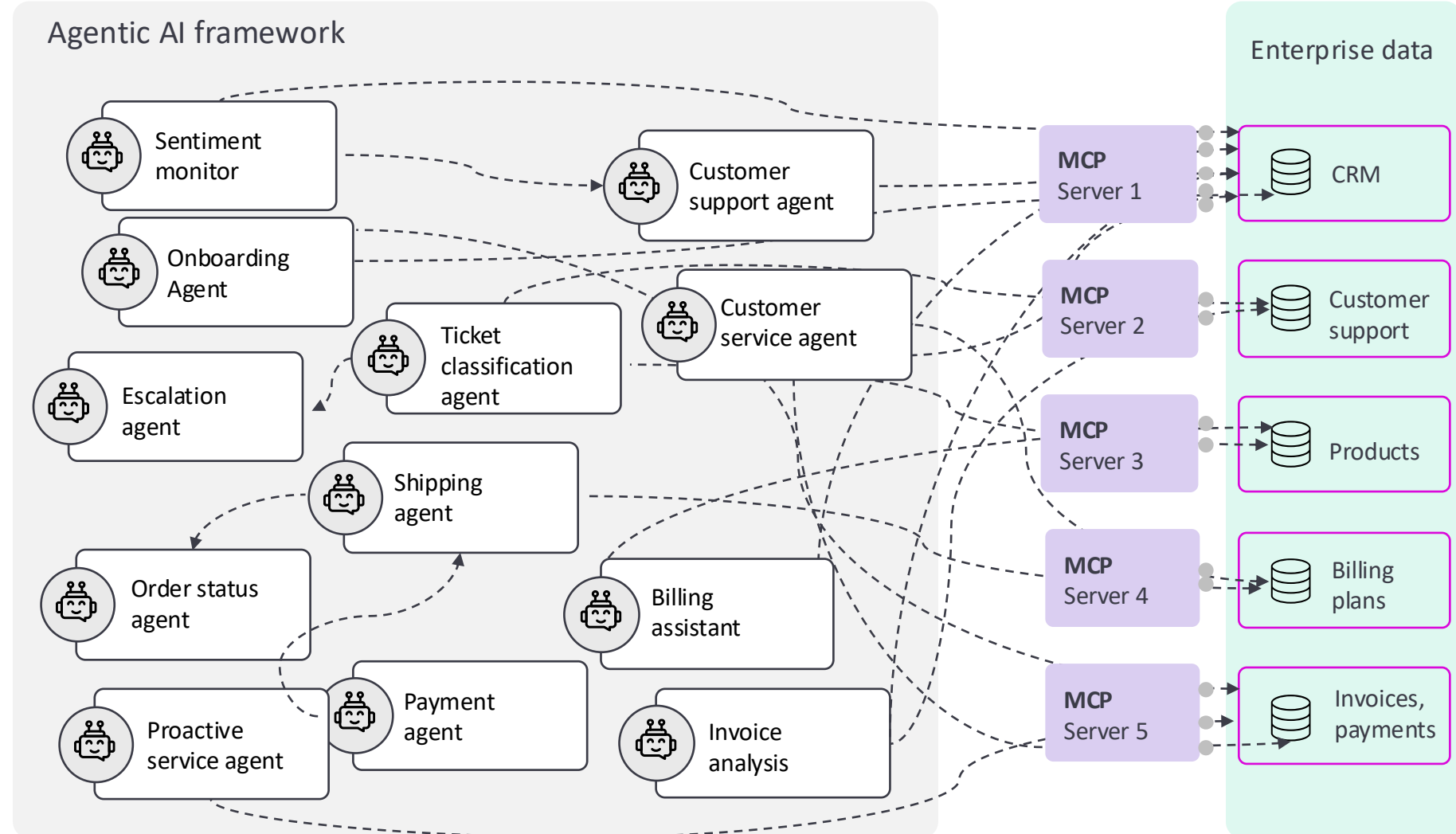
Approach 1b: MCP

✦ **Architectural gap:** MCP exposes systems, not business context

! Result:

MCP lacks the business context

- MCP returns system-specific data that the LLM must integrate
- Redundant data assembly duplicates work and drives token costs up
- Latency grows with the number of systems queried
- Expanded security risk



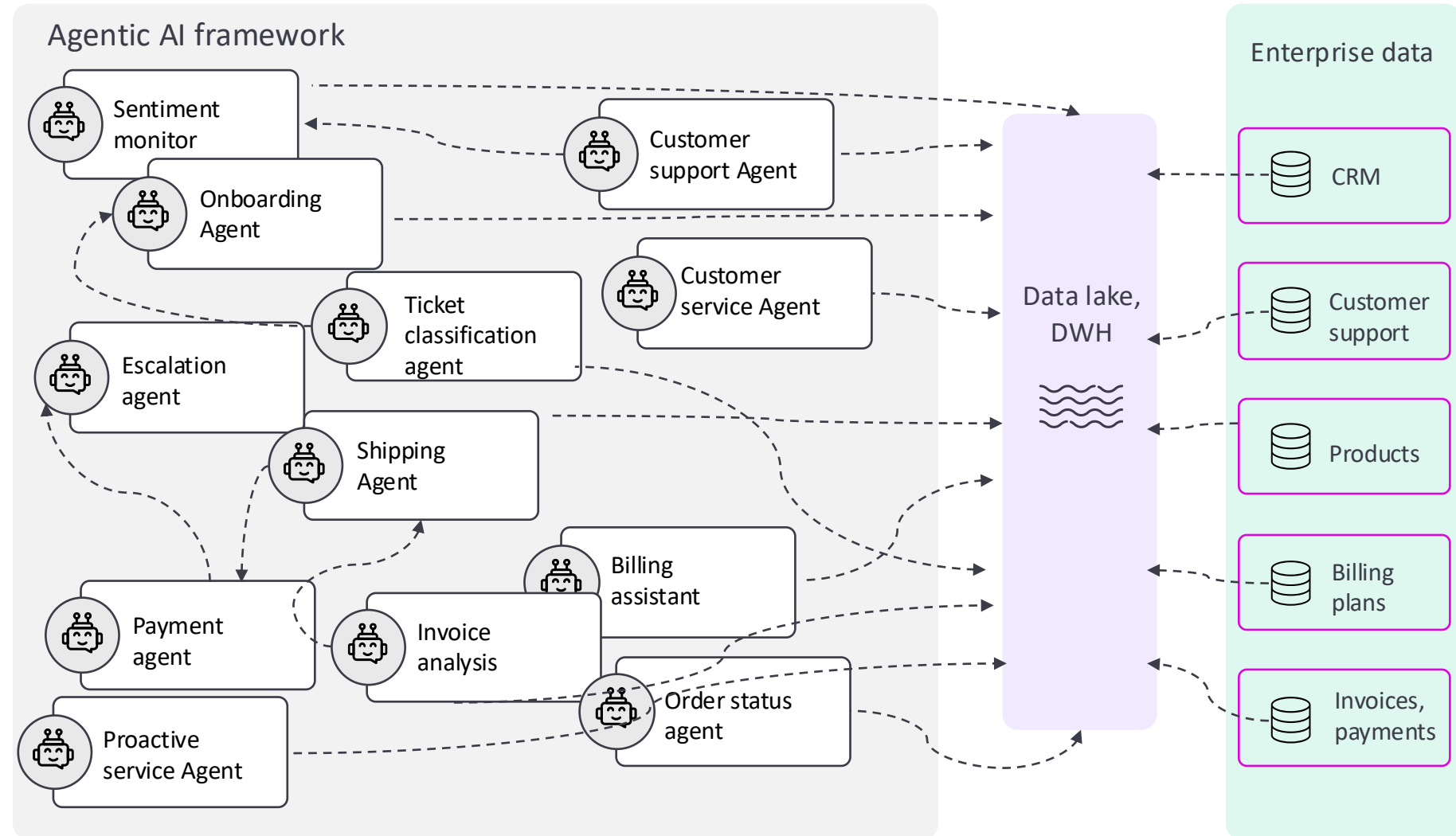
Approach 3: Data lake/warehouse

✦ **Architectural gap:** no governed runtime layer for ai questions

! Result:

Expensive, slow, and fragile AI

- High query and compute costs per question
- Latency unsuitable for interactive or agentic use and can't be proactive.
- Stale or incomplete operational context
- Privacy and security risks



k2view

Thank you!