



# Mainframe Data and AI

**Unlocking VSEn and z/OS Data for Real Time Operational Use Cases  
and Artificial Intelligence (AI) Leverage and Acceleration**

Boris Bulanov, K2view  
*[boris.bulanov@k2view.com](mailto:boris.bulanov@k2view.com)*

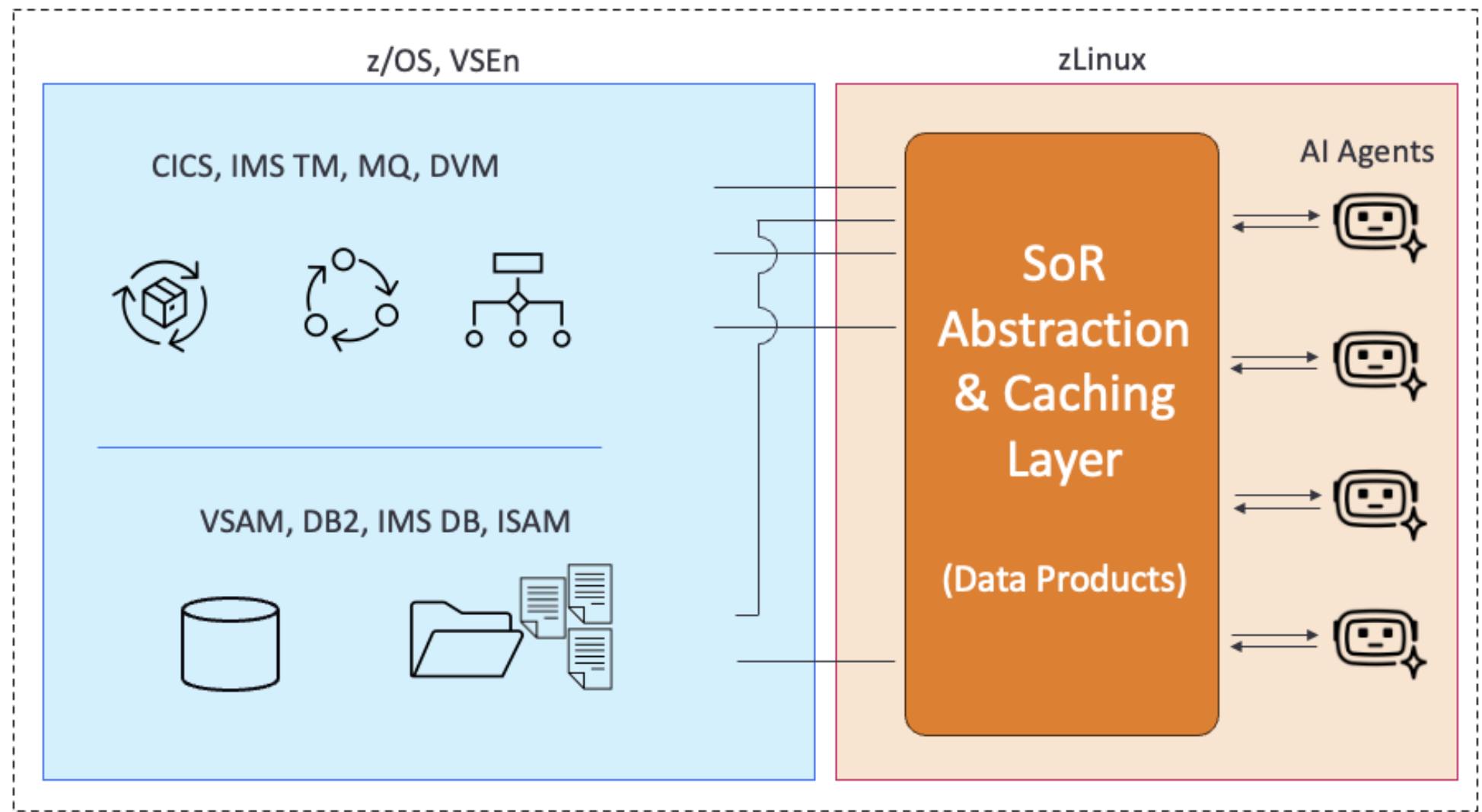


# Key Takeaways

- Mainframe-hosted systems of record (SoRs) are among the enterprise's most valuable AI assets
- This value must be unlocked by making SoRs consumable by AI agents
- The **operational data product** pattern is the approach; K2view is a proven implementation of this approach



# Concept





# Topics

- Context for AI in an Enterprise
- Organizing data for AI consumption
- Operational Data Product Platform case-study – K2view
- Use-cases on Mainframe



# Context for AI in an Enterprise



# AI Is on an Exponential Trajectory for Enterprises

## Recent headlines:

- *Solving new Math*
- *Compiling Linux Kernel*
- *Transforming ERP*



## Building a C compiler with a team of parallel Claudes

To stress test it, I tasked 16 agents with writing a Rust-based C compiler, from scratch, capable of compiling the Linux kernel. Over nearly 2,000 Claude Code sessions and \$20,000 in API costs, the agent team produced a 100,000-line compiler that can build Linux 6.9 on x86, ARM, and RISC-V.

**An OpenAI model solved a famous math problem that stumped humans for 80 years**

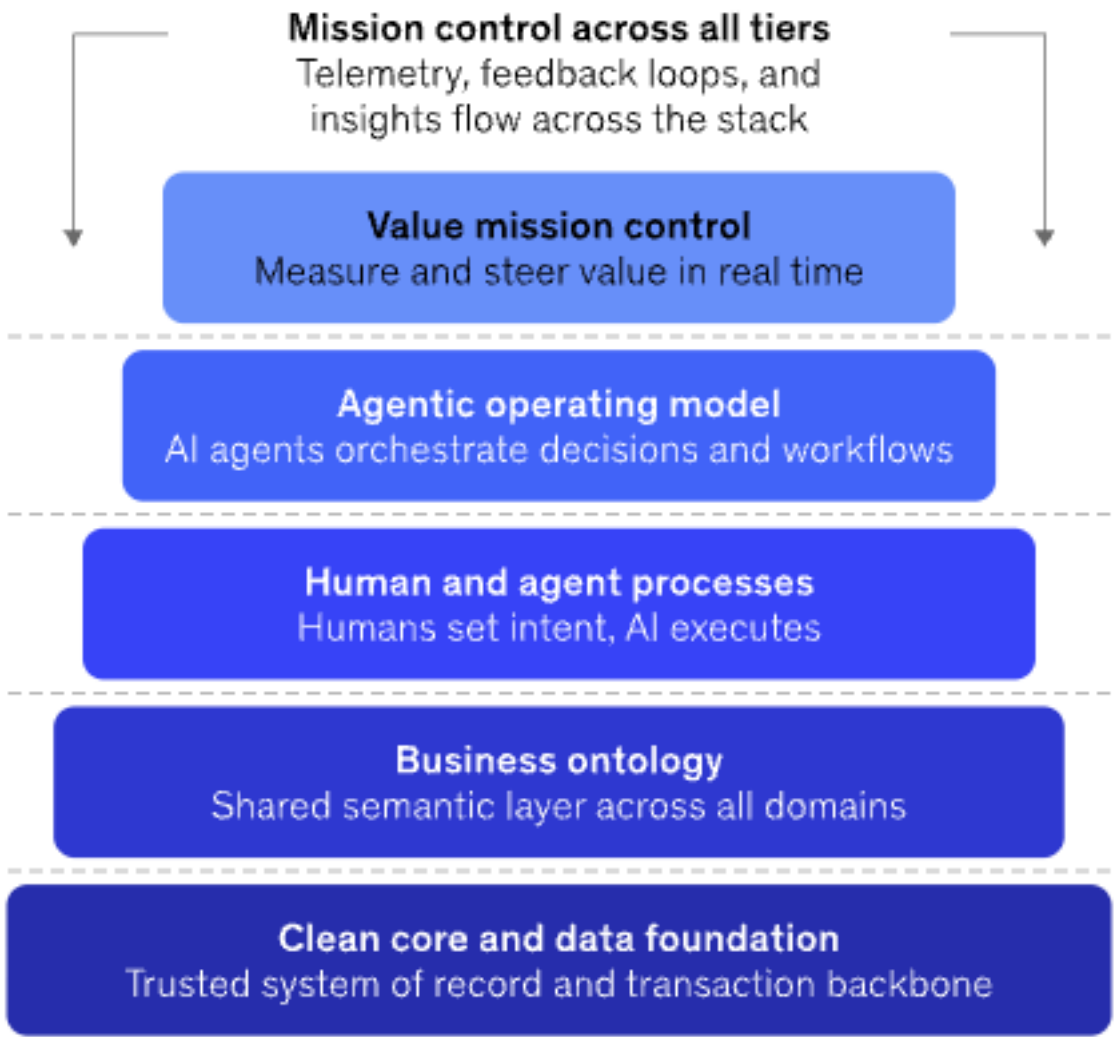
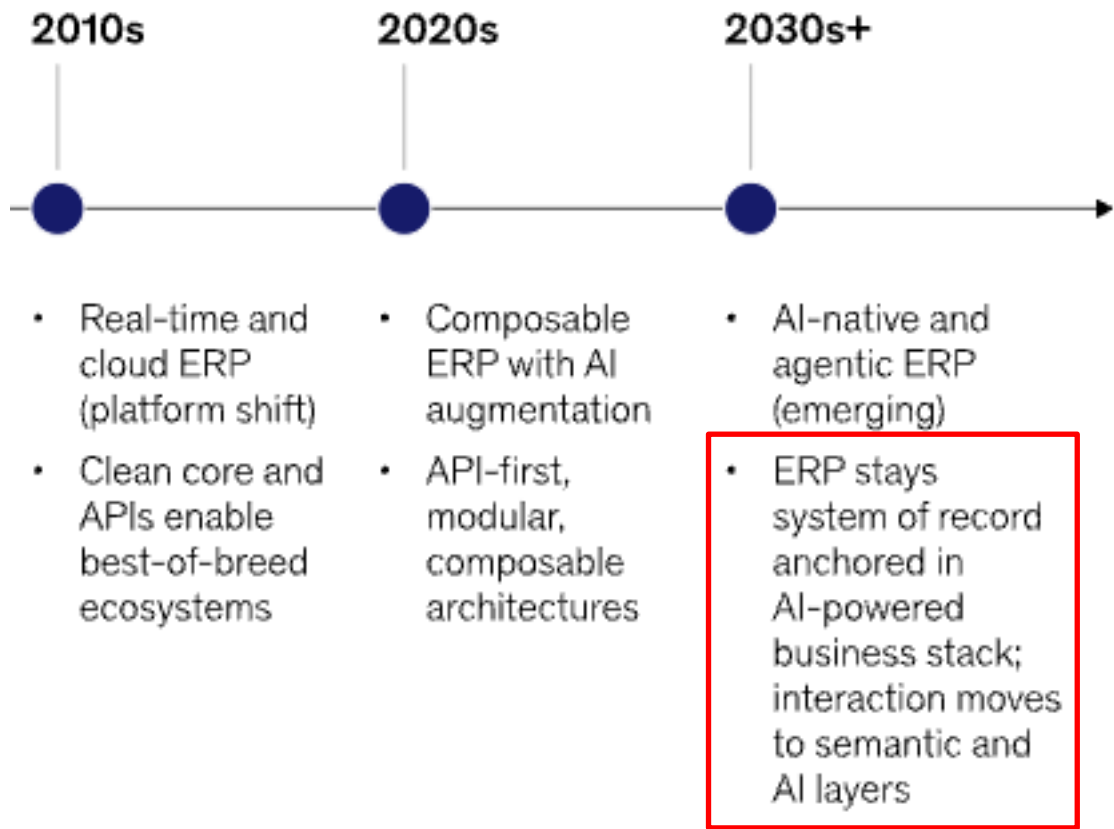




# The end of ERP as we know it ?

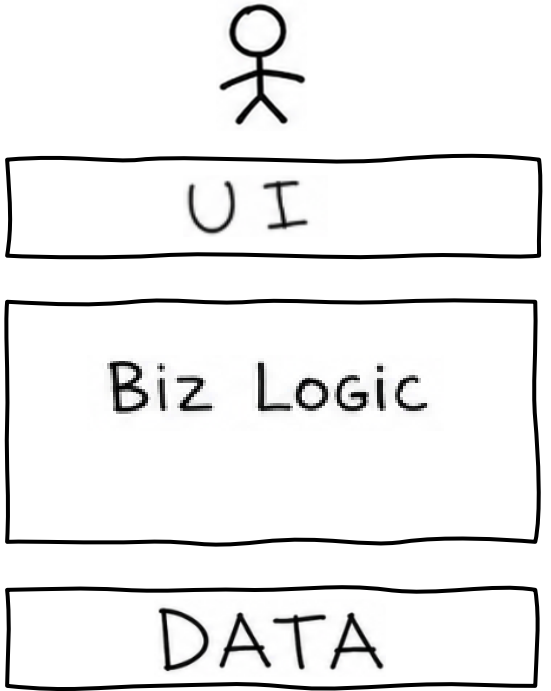
By McKinsey

*“ERP is evolving from transaction-centric systems of record to AI-native, agent-driven decision platforms”*

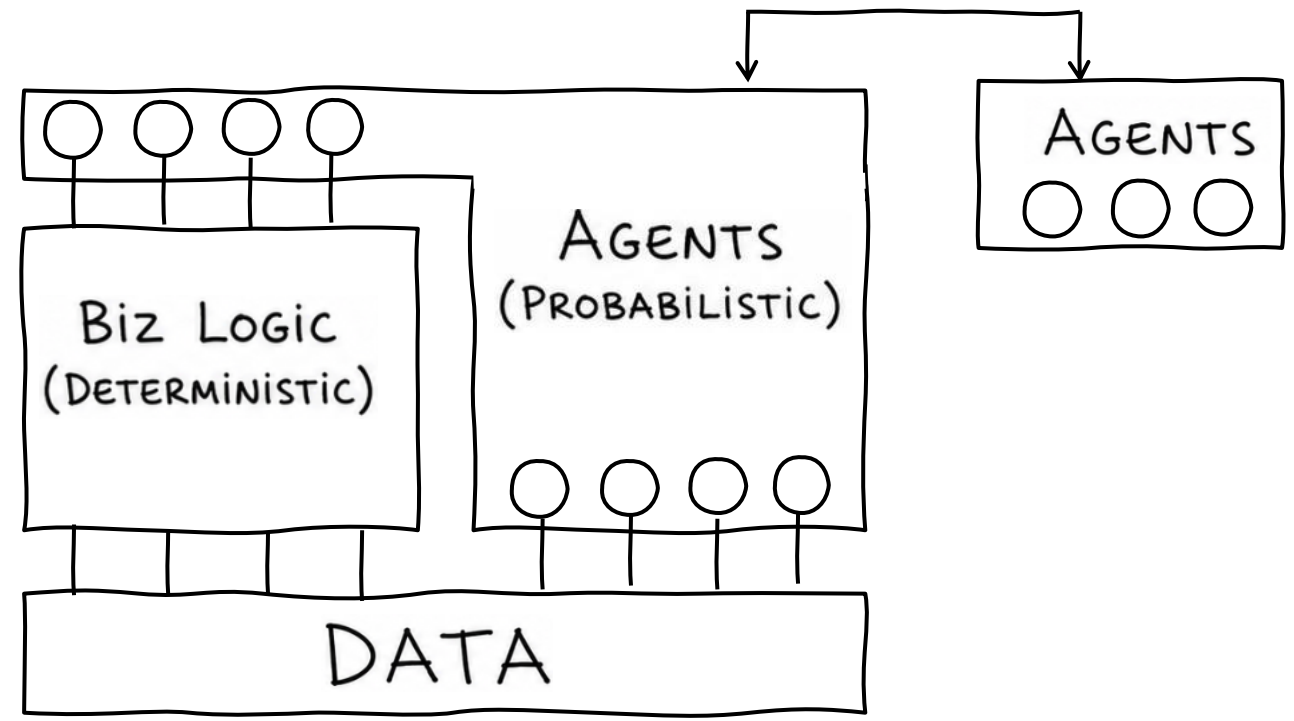




# TODAY

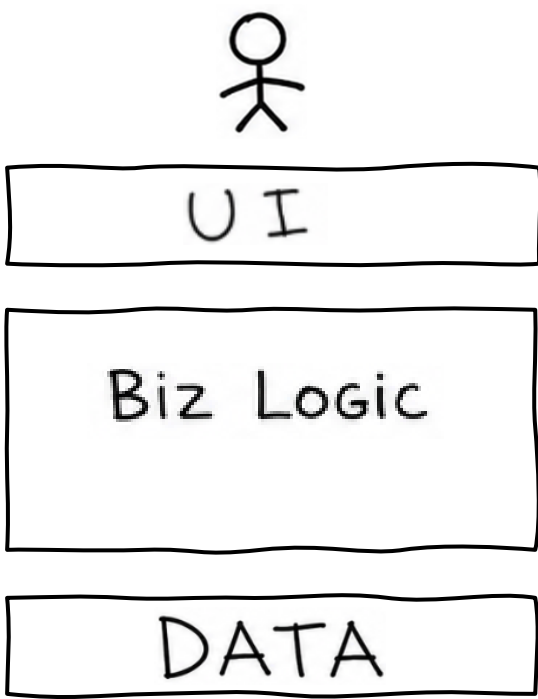


# FUTURE STATE

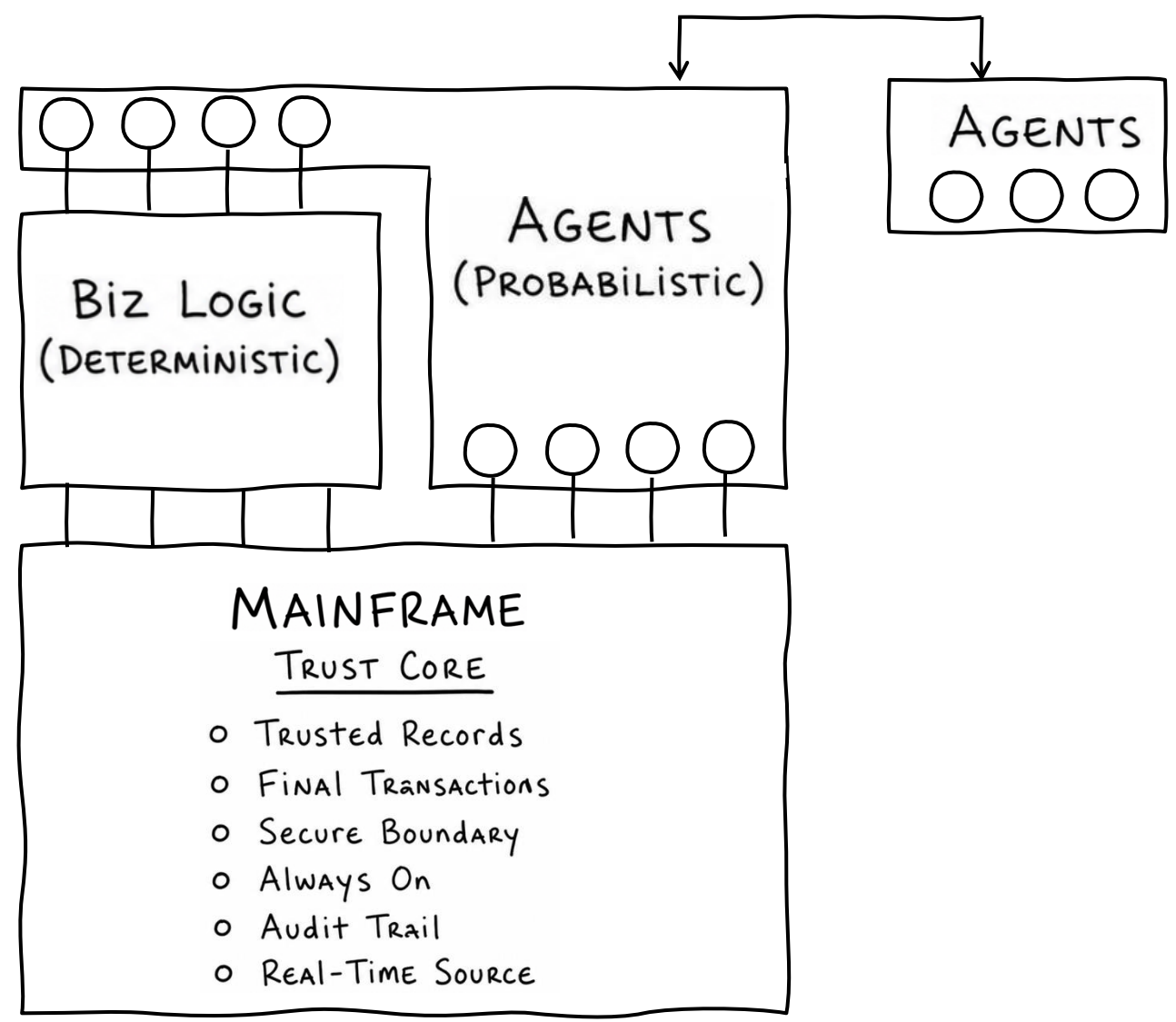




# TODAY



# FUTURE STATE WITH MF



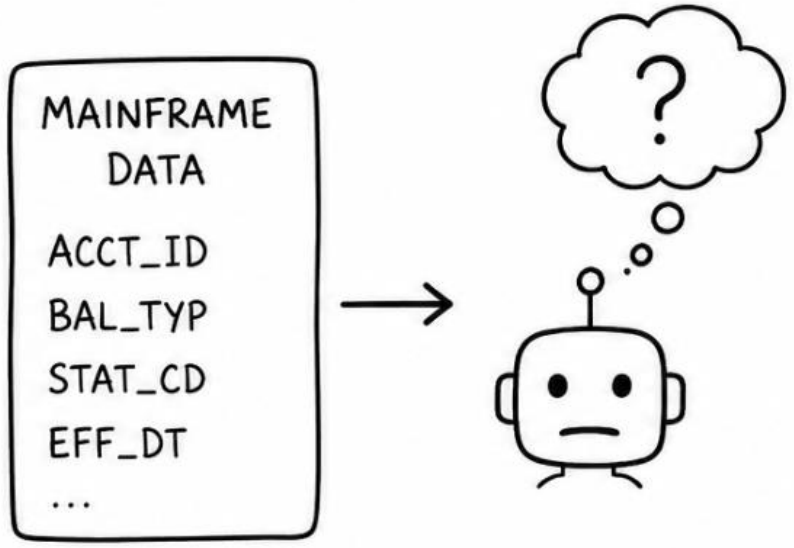


# Organizing Data for AI



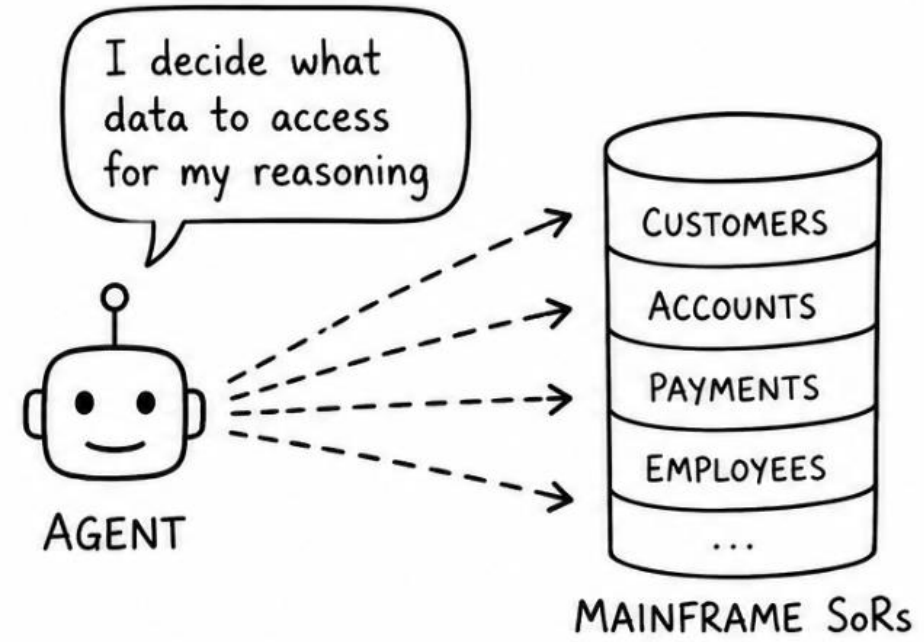
# Can't expose MF SoRs Directly to AI Agents

## ① NO BUSINESS CONTEXT



Agents see technical data, not business meaning.

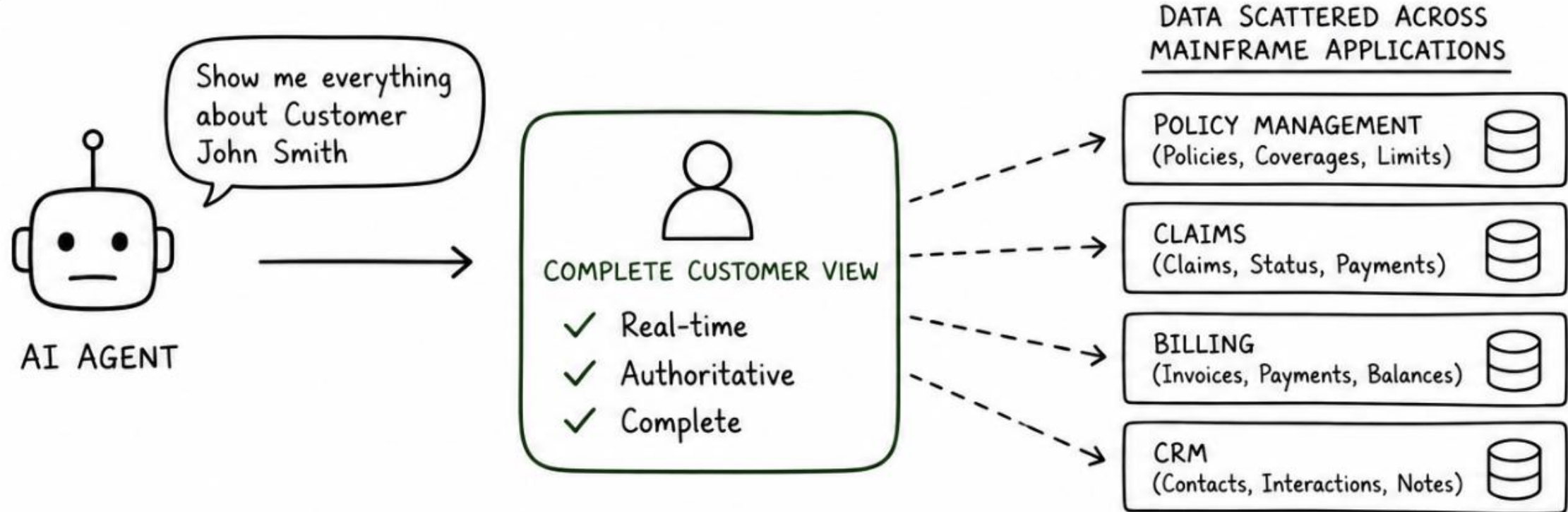
## ② NO CONTROL BOUNDARY




No boundaries, no guardrails. Agent decides what to access.



# AI Agent Data Requirements



 AI Agents need real-time, authoritative, and complete data to deliver accurate answers and safe actions.

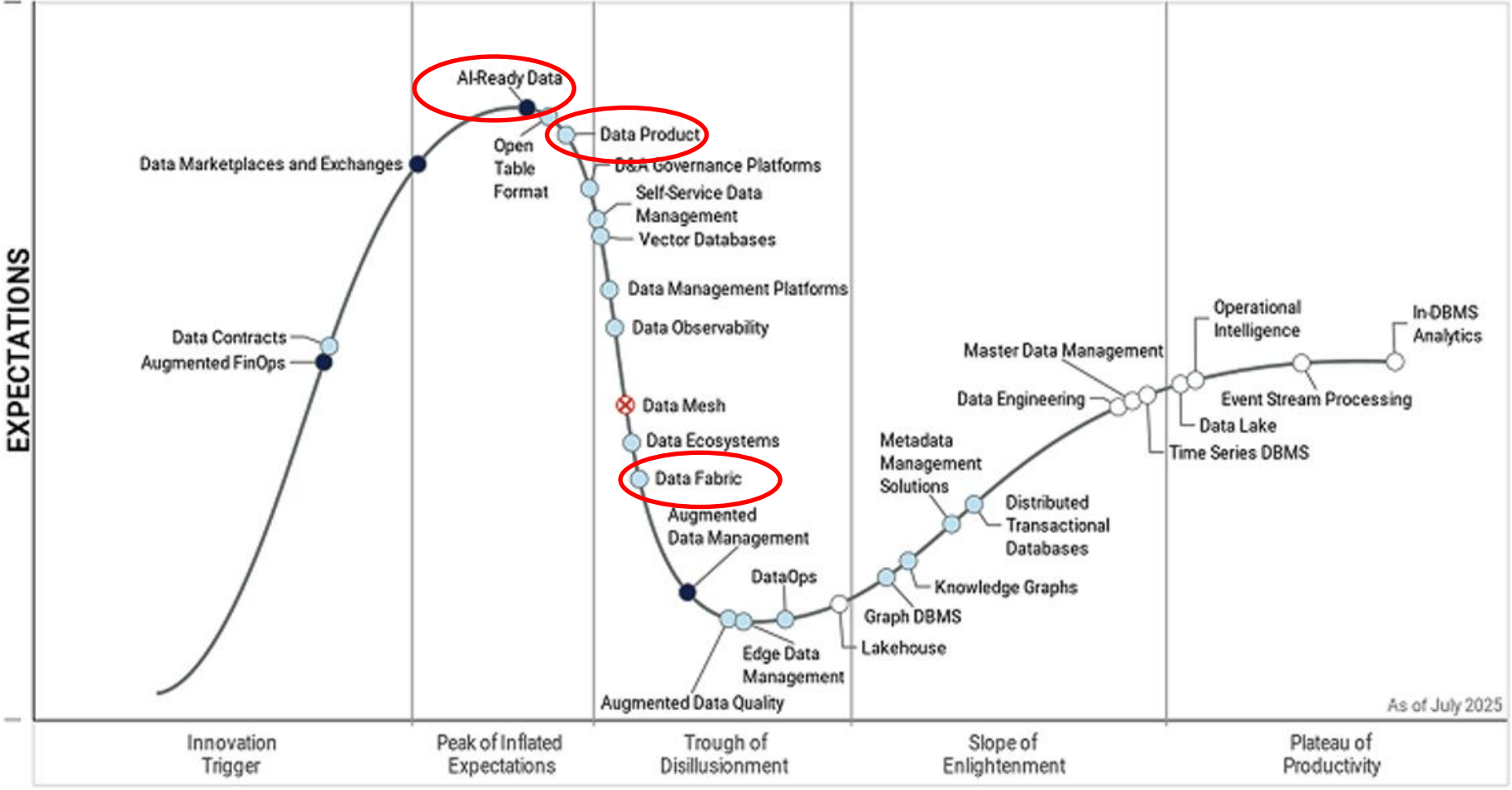


# Data Management Approaches

## Related Concepts:

- Digital Twin
- Business Entity
- AI-Ready Data
- Data Fabric
- Data Product

Hype Cycle for Data Management, 2025



Plateau will be reached: ○ <2 yrs. ● 2-5 yrs. ● 5-10 yrs. ▲ >10 yrs. ⊗ Obsolete before plateau

As of July 2025



# What is a Data Product?

- **Reusable** data asset
- Maintained as a **“software product”**
- Built-in security/privacy
- Leading to a **composable** data architecture

Data Products Stack

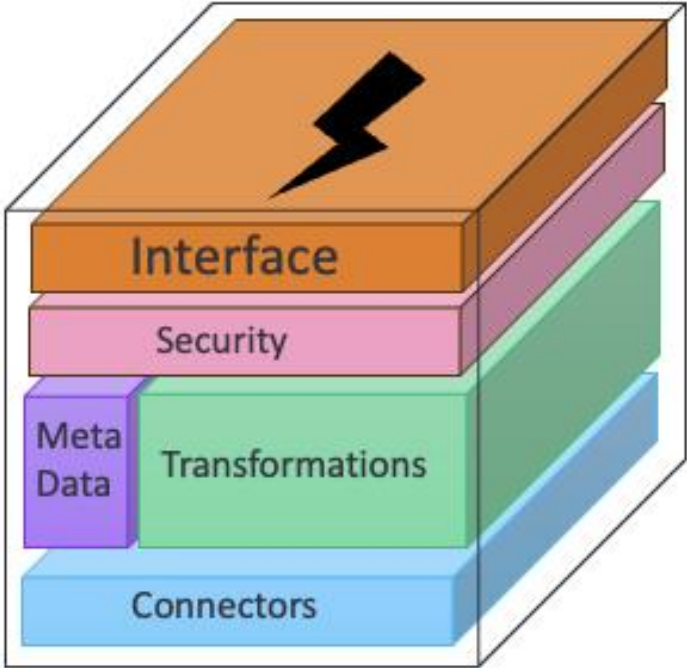
Users, Consumers, Applications

Data Products (Platform)

Data Sources



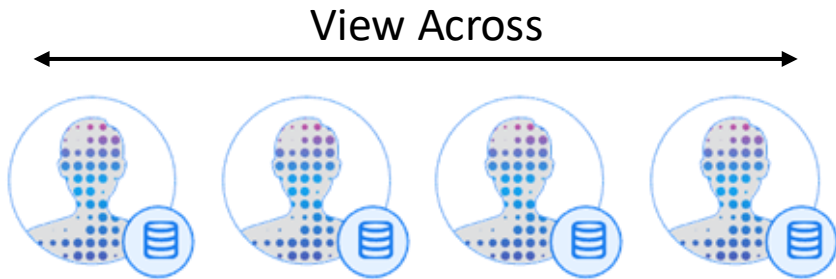
Data Product





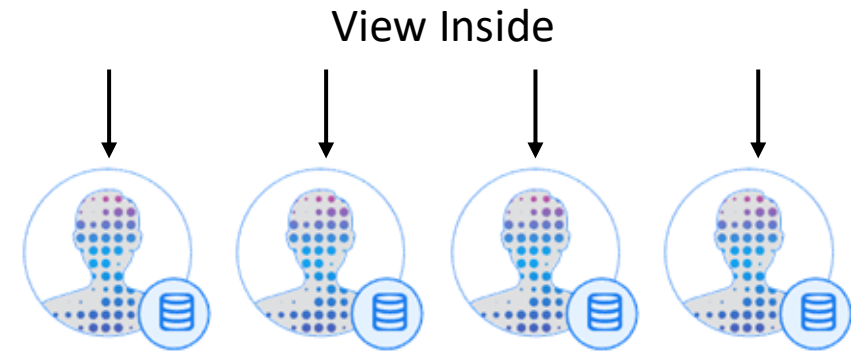
# Types of Data Products

## Analytical Data Products



- Mostly used for internal “Decision Support”
- Aggregations, batch, read-only
- Examples:
  - “What was yesterday’s revenue by region?”
  - “Which customers are likely to churn?”

## Operational Data Products



- Mostly used by upstream business applications
- Lookups, real-time, read-mostly, ACID
- Examples:
  - “What is the average balance for customer123?”
  - “What branch does she visit the most ”



# Data Product Platform case-study – K2view



# K2view: Agentic data products at scale

## Media & Telco



## Financial Services



## Retail



## Healthcare



## Consulting and Staffing



## Manufacturing & Logistics, Oil & Gas



## ISVs (OEMs), technology





# Technology At Scale

Gartner Magic Quadrant “Visionary” company

**Tier 1 Bank**

**350+**

Applications under TDM

**Tier 1 Telco**

**15PB/Month**

Data product  
transactions (3,500/s)

**Tier 1  
Insurance**

**500,000+**

Data elements  
in 1 data product

**Tier 1 Telco**

**300M+**

Micro-DBs deployed

**Tier 1 Telco**

**<300ms**

Response time  
committed



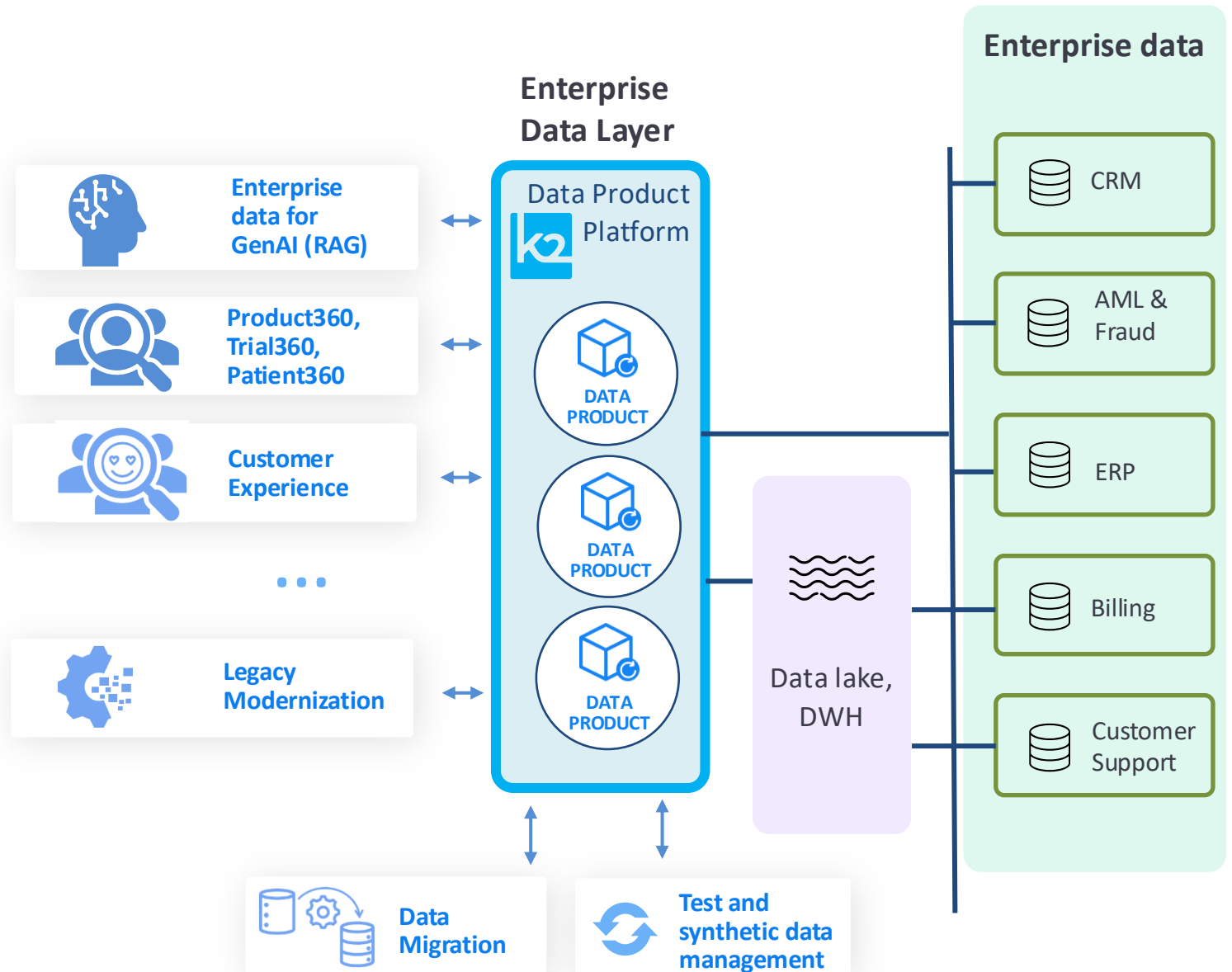
# K2view Enterprise Data Product Platform

## Non-functional Capabilities:

- Unlimited scale
- Real-time performance
- Fresh data

## Functional Capabilities:

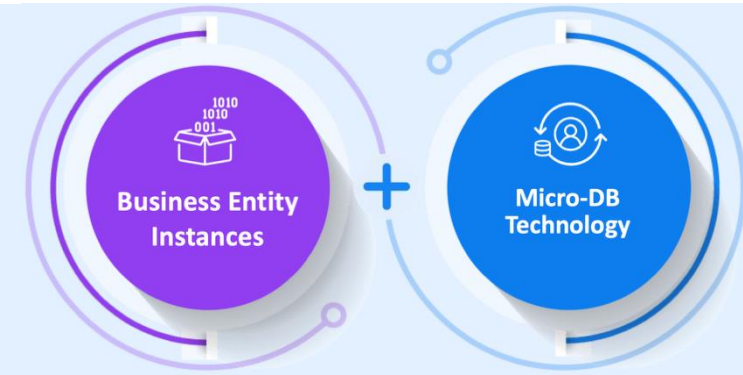
- Flexible Common Data Model
- Easily consumable
- Data Security and Privacy
- Portfolio of connectors
- Real-time ETL
- Memory/Caching/Virtualization
- Storage of cached data
- Flexible data synchronization
- Test Data Management





# Our Secret Sauce: The Micro-DB™

One Micro-DB for every business entity instance



Instead of querying massive DBs with complex joins, and then unifying, processing, and masking the data in batch...

**query a single Micro-DB that's always fresh, unified, and compliant, *in msec***

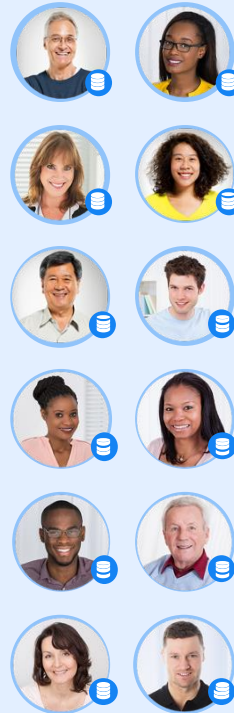
- ✓ Quicker time to value
- ✓ Greater trust in customer data
- ✓ Common language between business & IT
- ✓ Real-time performance
- ✓ Data governance enforced on the fly

## CREATE | PROCESS Micro-Databases



- Ingest
- Unify
- Enrich
- Transform
- Mask
- Secure
- Compress

## SYNC | DELIVER | EXPOSE Micro-Databases



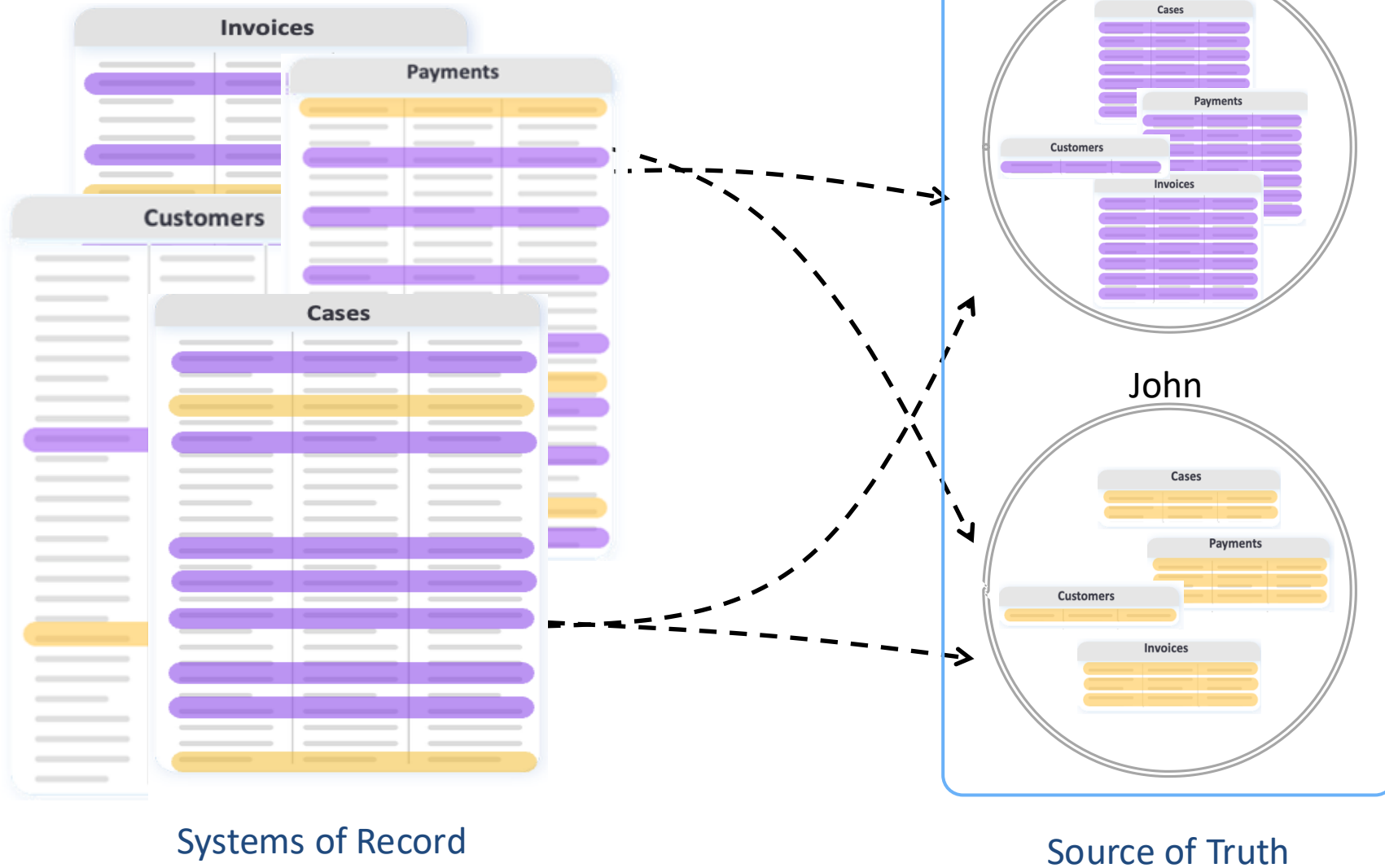
- Segmentation
- DWH/ Lake
- CRM
- Cloud Apps
- AI/ML
- IVR
- Mobile Apps

- Databases
- Legacy systems
- SaaS Apps
- Applications
- Web Services
- Files





# Accessing Data with Business Entity Instances



- IT/Business alignment
- Real-time data access
- Hyper scale
- Data security



# Data Product Design Studio

**k2studio**

File Fabric Edit Selection View Go Run Help

Schema (customer) x

Implementation > Logical Units > customer > Schema (customer)

65% Search

**CRM (Salesforce)**

- CRM\_Customer (population.flow 1)
  - customer\_id
  - customer\_type
  - first\_name
  - last\_name
- CRM\_Interactions (population.flow 2)
  - interaction\_id
  - customer\_id
  - interaction\_type
  - interaction\_date
- CRM\_Addresses (population.flow 2)
  - address\_id
  - street\_address
  - city
  - state

**DDA (Oracle)**

- DDA\_Accounts (population.flow 2)
  - account\_id
  - customer\_id
  - account\_number
  - branch\_id
- DDA\_ACH\_Transfers (population.flow 3)
  - transfer\_id
  - sender\_account\_id
  - receiver\_account\_id
  - amount
- DDA\_Fees (population.flow 3)
  - fee\_id
  - account\_id
  - fee\_type
  - fee\_amount
- DDA\_Overdrafts (population.flow 3)
  - overdraft\_id
  - account\_id
  - overdraft\_amount
  - overdraft\_date
- DDA\_Transactions (population.flow 3)
  - transaction\_id
  - account\_id
  - transaction\_type
  - amount
- DDA\_Virtual\_Accounts (population.flow 3)
  - virtual\_account\_id
  - real\_account\_id
  - virtual\_account\_name
  - balance

**Loans (Cosmos DB)**

- Loan\_Accounts (population.flow 2)
  - loan\_account\_id
  - customer\_id
  - loan\_type
  - requested\_amount
- Loan\_Escrow\_Transactions (population.flow 3)
  - escrow\_transaction\_id
  - loan\_account\_id
  - transaction\_date
  - transaction\_amount
- Loan\_Covenant\_Reviews (population.flow 3)
  - review\_id
  - loan\_account\_id
  - covenant\_type
  - review\_status
- Loan\_Interest\_Rates (population.flow 3)
  - interest\_rate\_id
  - loan\_account\_id
  - interest\_rate
  - effective\_date
- Loan\_Servicer\_Changes (population.flow 3)
  - servicer\_change\_id
  - loan\_account\_id
  - previous\_servicer\_name
  - new\_servicer\_name
- Loan\_Interest\_Transactions (population.flow 3)
  - interest\_transaction\_id
  - loan\_account\_id
  - transaction\_date
  - interest\_amount
- Loan\_Payments (population.flow 3)
  - payment\_id
  - loan\_account\_id
  - payment\_date
  - amount\_paid

**CCMS (SQL Server)**

- CCMS\_Credit\_Cards (population.flow 2)
  - card\_id
  - customer\_id
  - card\_type
  - card\_number
- CCMS\_Credit\_Card\_Payments (population.flow 3)
  - payment\_id
  - card\_id
  - statement\_id
  - payment\_date
- CCMS\_Credit\_Card\_Statements (population.flow 3)
  - statement\_id
  - card\_id
  - statement\_date
  - total\_due
- CCMS\_Credit\_Card\_Transactions (population.flow 4)
  - transaction\_id
  - card\_id
  - statement\_id
  - transaction\_date

**Tax (Couchbase)**

- Tax\_Customers (population.flow 2)
  - customer\_id
  - tax\_identifier
  - residency\_status
  - tax\_filing\_status
- Tax\_Deductions (population.flow 3)
  - deduction\_id
  - customer\_id
  - tax\_year
  - deduction\_type
- Tax\_Customer\_Properties (population.flow 3)
  - property\_id
  - customer\_id
  - purchase\_price
  - purchase\_date
- Tax\_Liabilities (population.flow 3)
  - liability\_id
  - customer\_id
  - tax\_year
  - tax\_type
- Tax\_Payments (population.flow 3)
  - status
- Tax\_Transactions (population.flow 3)
  - transaction\_id
  - customer\_id
  - transaction\_type

**Payroll (Postgres)**

- Payroll\_Contracts
- Payroll\_Processing

afusion-demo-elitebank\* Project Version: Deployment Status: All Deployed Java: Ready



# Managing all data product lifecycle phases

2

## Data Integration

- Support all integration methods (Messaging, Streaming, CDC etc.)
- Apply smart synch policies (time interval, decision function etc.)

3

## Data Product Engineering

- AI copilot-assisted data pipeline creation and documentation
- Data transformation and orchestration
- Data governance: data quality and data privacy policies and enforcement

1

## Data Product Design

- Data catalog-driven design
- AI-based classification
- Auto-discovery and modeling of logical semantic layer
- Virtualization and/or persistence

4

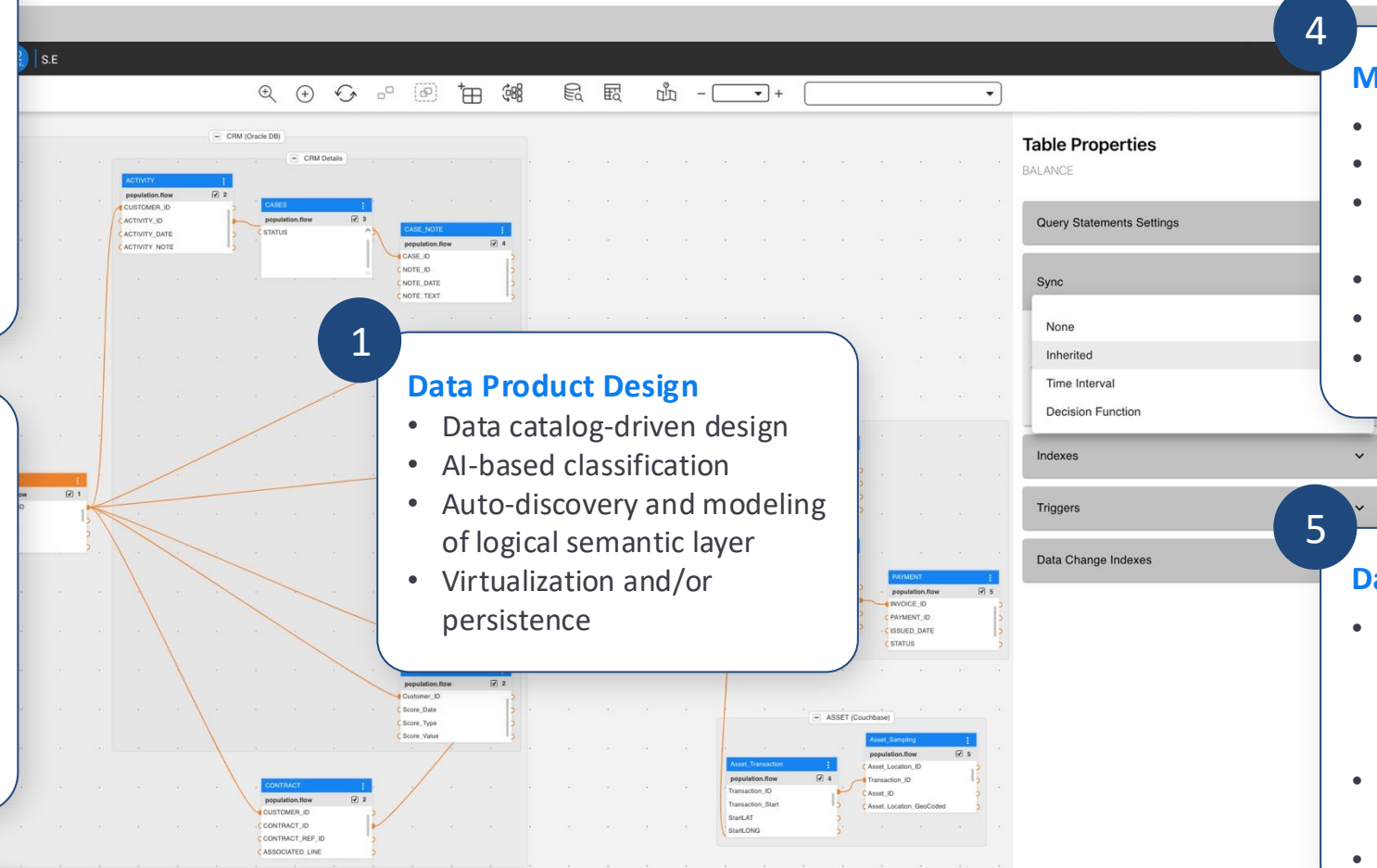
## Monitoring

- Observability
- Performance
- Reporting and dashboarding
- Alerting
- Quality assurance
- Role-based access controls

5

## Data Product Delivery

- Low-code framework to create, debug, and deploy data services in minutes
- Expose and Deliver data in pull or push mode
- Deliver data in mSec





# Lightweight ETL Engine

## Design Studio

- Visual drag and drop
- Real-time
- Bi-directional
- Any type of data source: databases, files, events, applications, etc.
- Built-in data-level debugger
- Library of hundreds of pre-built components

The screenshot shows the k2studio interface for designing an ETL flow. The main workspace is divided into four vertical panels: **Input**, **Source**, **Stage 1**, and **LU Table**.   
1. **Input**: Contains a component named 'PopulationArgs' with fields like CUSTOMER\_ID, SSN, FIRST\_NAME, LAST\_NAME, SATISFACTION, and PROFILE\_PIC. It also has a 'DEL SyncDeleteMode' component.   
2. **Source**: Contains a 'Db Query' component. A connection line links the 'iid' field of the 'PopulationArgs' component to the 'params' field of the 'Db Query' component.   
3. **Stage 1**: An empty stage for processing.   
4. **LU Table**: Contains a 'CUSTOMER' table component. A connection line links the 'result' field of the 'Db Query' component to the 'affected' field of the 'CUSTOMER' table component.   
On the right side, there is a configuration panel for the 'CUSTOMER' table. It shows 'Query : SourceDbQuery', 'All Fields' selected, and a list of 'Inputs' including 'interface : string', 'CRM\_DB\_SQLITE', 'sql : string', 'params : any', and 'parent\_rows : [object]'. The 'sql' field contains the query: `select * from main.CUSTOMER`.   
At the bottom right, there is a small inset window showing a zoomed-in view of the connections between the 'Db Query' and 'CUSTOMER' table components.



# Operational Data Products on Mainframe

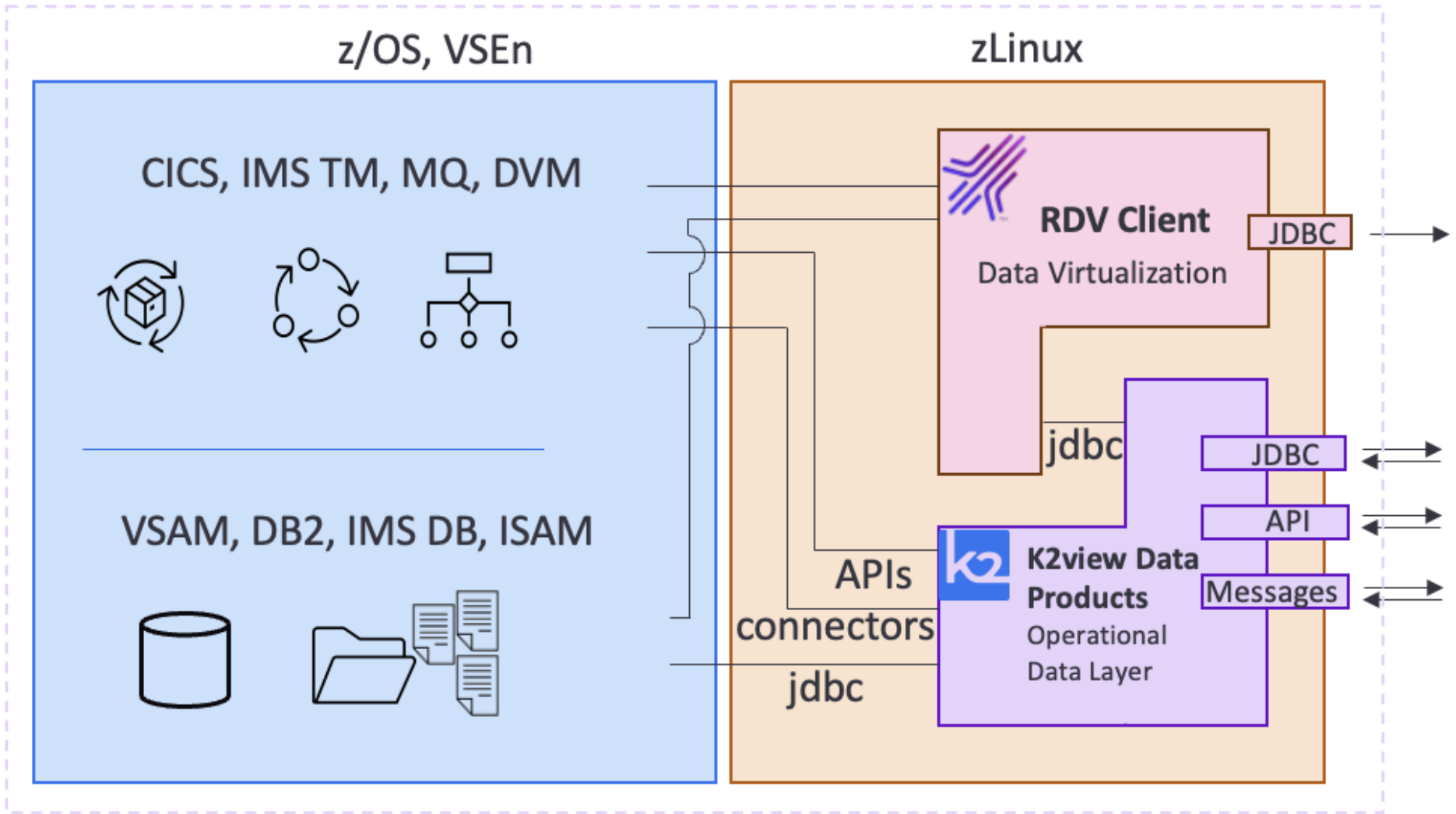


# K2view Patterns on Mainframe

- **Data Products Platform**
  - Business-entity-centric view of data across multiple systems
  - Caching layer for real-time performance and hyper-scale
- **Data Products for Agentic AI**
  - Organizing enterprise data for optimal Agentic access
  - Unified semantics, CRUD for actions, data vaults for security
- **Test Data Management**
  - Data Masking and Synthetic Data Generation
  - Self-service, DevOps, security, multiple environments
- **Data Migration**
  - Upgrades – between versions and technologies
  - Modernization – Mainframe to distributed systems



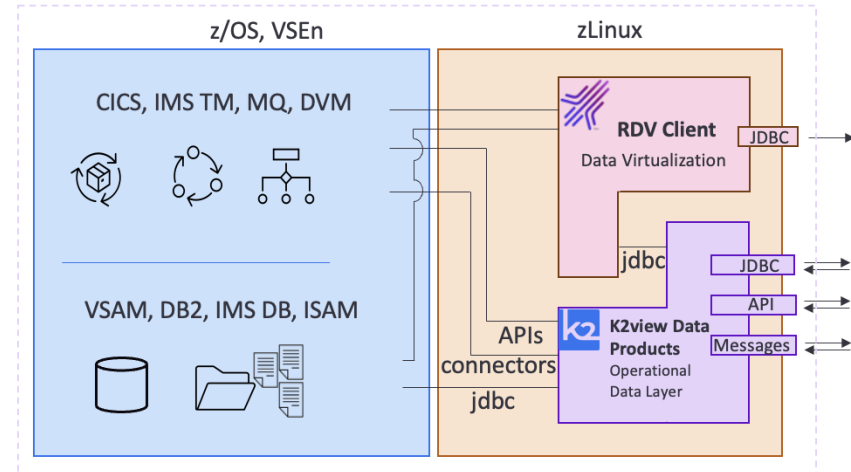
# K2view Data Products on Mainframe





# Transforming data access on Mainframe

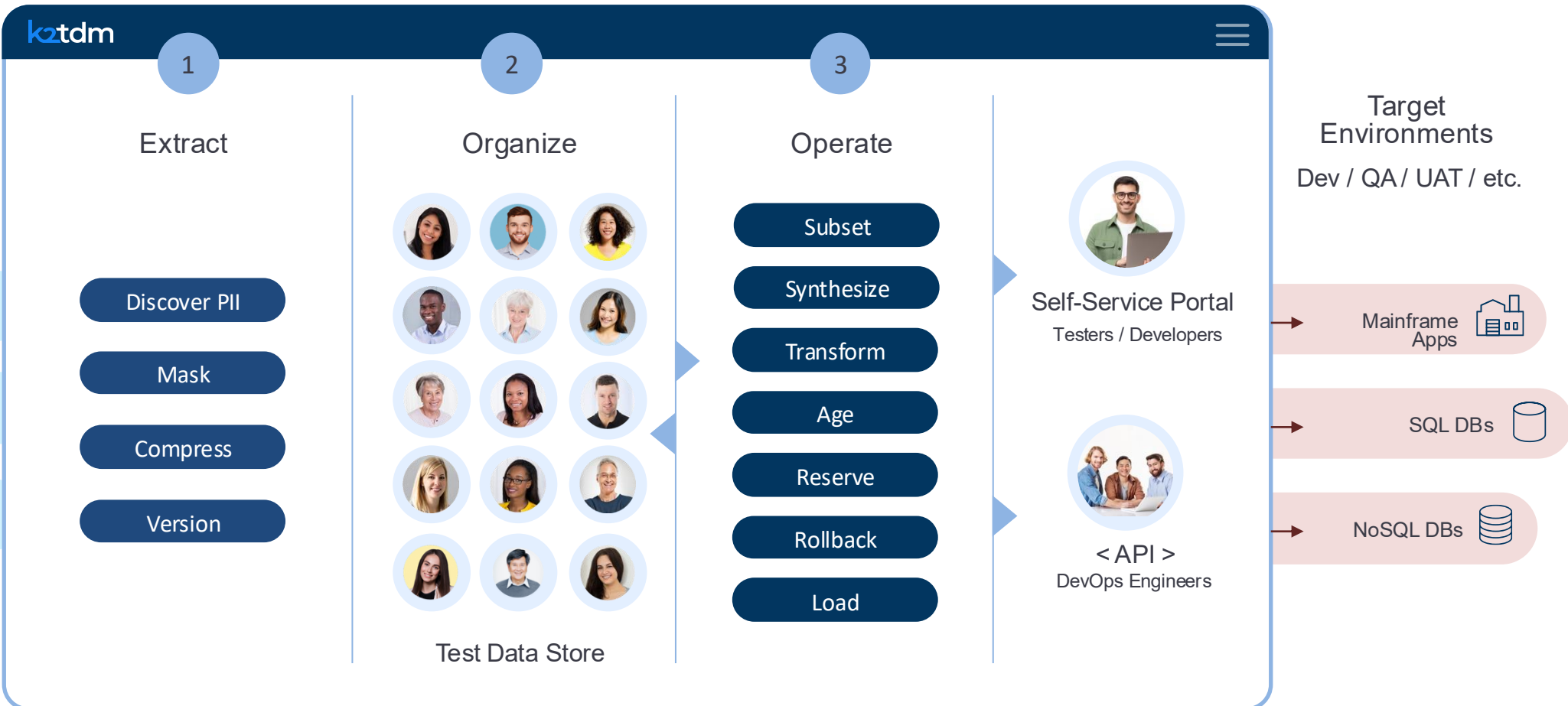
- **Execution inside the mainframe perimeter**
  - Security, reliability
- **Business-focused entity-centric design**
  - Understood by Agents and Business
- **Data organized for reuse**
  - Same entities utilized across many use-cases and technologies
  - Governed and secure



- **Modern, open, extensible architecture**
  - JDBC, APIs, MCP, Events
- **Caching and synchronization**
  - Offloading workloads from Systems of Record
  - Reducing MIPS spend

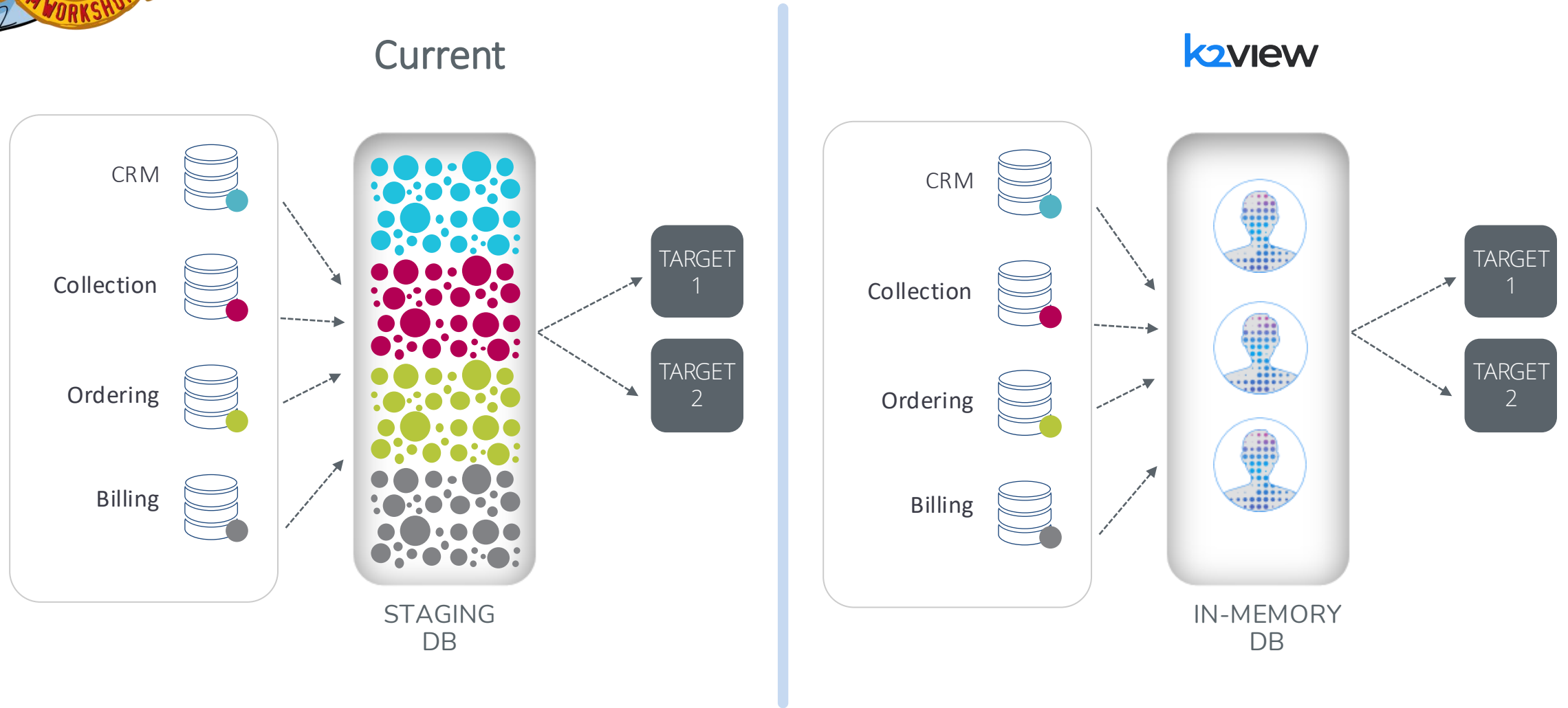


# Test Data Management for Mainframe





# Migration Approach Current -vs- K2view





# Key Takeaways

- Mainframe-hosted systems of record (SoRs) are among the enterprise's most valuable AI assets
- This value must be unlocked by making SoRs consumable by AI agents
- The **operational data product** pattern is the approach; K2view is a proven implementation of this approach