Materialize

WHITEPAPER

Why AI Systems Fail—And How Real-Time Data Fixes Them

Your company has invested in a sophisticated customer service chatbot, combining the latest large language model with a vector database containing years of indexed support tickets. The system handles complex product questions and troubleshooting scenarios with impressive accuracy. Yet when a long-time premium customer asks about their recent order, the bot fails in the most basic way possible - it treats them like a stranger, unable to access their current order status or account details. In an instant, your significant AI investment transforms from an efficiency driver into a frustration generator.

While advances in generative AI and large language models have captured headlines and imagination, the success of AI applications hinges on their ability to ground responses in current reality.

Why AI Systems Break

The challenge runs deeper than simple data freshness. Modern AI systems are increasingly complex orchestrations of multiple components - vector databases storing historical knowledge, language models processing queries, and operational databases tracking current state. Each component might work perfectly in isolation, yet the system as a whole fails because it can't maintain a consistent view of rapidly changing business data.

Consider what happens when multiple AI agents need to coordinate actions based on rapidly changing data. A pricing algorithm adjusts product costs based on inventory levels, while simultaneously, a recommendation system suggests products to customers, and a supply chain AI manages restock orders. Without fresh, consistent data, these systems can work at cross-purposes – the pricing engine might drop prices on items that are running low, while the recommendation system pushes those same products to more customers, creating a cascade of inventory problems.

Traditional data architectures weren't built for these demands. Data warehouses update on fixed schedules, creating gaps between reality and decision-making. Streaming platforms can process fresh data but introduce eventual consistency issues, forcing teams to implement complex synchronization logic. Caching layers provide quick access to frequently needed data but risk serving stale results if not meticulously invalidated.

Materialize: Rethinking Data for AI

Materialize, a real-time data integration platform, addresses these challenges through a fundamentally different approach to data management. Rather than forcing AI systems to repeatedly query production databases or work with stale cache data, Materialize maintains continuously updated views of your operational data with strong consistency guarantees. Materialize uniquely addresses three critical requirements that AI systems need to deliver reliable, real-time responses:

- Fast Queries for Online Inference: AI systems need to respond in milliseconds, not seconds. Through incremental view maintenance, Materialize delivers instant query results that support real-time inference without repeatedly hitting production systems.

- Fresh Data That Matches User Expectations: Users expect AI responses to reflect their most recent interactions and current system state. Materialize ensures AI systems always work with up-to-date data, maintaining user trust and enabling meaningful interactions.
- Correct Data for Reliable Responses: Consistency isn't just a technical requirement—it's essential for building trust in AI systems. Materialize maintains a globally consistent view of your data, ensuring all AI components work from the same understanding of current state.

When these three capabilities intersect, they create something more powerful: a complete operational data system that can serve as the foundation for reliable AI applications.

The Power of Complementary Technologies

The emergence of vector databases has revolutionized how AI systems access and understand unstructured information. These specialized databases excel at storing and retrieving high-dimensional embeddings derived from documents, web pages, and historical data, enabling sophisticated semantic search and contextual understanding. They form the bedrock of modern retrieval-augmented generation (RAG) systems, providing the deep subject matter expertise that allows AI models to ground their responses in authoritative content.

Yet vector databases alone aren't enough. While they excel at managing the "what" and "why" of your domain knowledge, they lack the crucial element of "now" – the operational context that turns general knowledge into actionable insights. This is where Materialize enters the equation, not as a competitor to vector databases, but as a crucial complementary technology.

The transformation that occurs when combining vector databases with operational data is profound. Consider this example of an e-commerce assistant responding to a simple product inquiry:

Without operational data, the AI assistant works solely from static knowledge:

```text
Prompt:
You are a helpful assistant for an e-commerce site. Based on the following retrieved documents:
1. Inventory Report: 'As of this morning, 10 Deluxe Espresso Machines were in stock in the central warehouse.'
2. Shipping Policy: 'Standard delivery takes 3–5 business days. Expedited delivery may be available in some locations.'

The user is asking: 'How many of the Deluxe Espresso Machines are available for delivery, and what's the estimated delivery time for my address?'
```

```
Response:
As of this morning, there are 10 Deluxe Espresso Machines in
stock in the central warehouse, so they are available for
delivery.

The estimated delivery time depends on the shipping option
you choose:
- Standard delivery: 3–5 business days
- Expedited delivery: May be available in some locations
```

Now watch how the response transforms when the same vector-retrieved knowledge is enriched with actionable operational data:

text

```
Prompt:
You are a helpful assistant for an e-commerce site. Based on
the following retrieved documents and real-time data:
1. Inventory Report: 'As of this morning, 10 Deluxe Espresso
Machines were in stock in the central warehouse.'
2. Shipping Policy: 'Standard delivery takes 3–5 business
days. Expedited delivery may be available in some
locations.'
3. Live Inventory: 'Currently, 7 Deluxe Espresso Machines
are in stock.'
4. Customer Data: 'User is located in Brooklyn, NY, and has
Gold Membership status.'
5. Delivery Rules: 'Free next-day delivery is available for
Gold Members in Brooklyn.'

The user is asking: 'How many of the Deluxe Espresso
Machines are available for delivery, and what's the
estimated delivery time for my address?'

Response:
Currently, 7 Deluxe Espresso Machines are available for
delivery.
```

```
Since you're located in Brooklyn, NY, and have Gold
Membership status, you qualify for free next-day delivery.
If you place your order today, your Deluxe Espresso Machine
will be delivered by tomorrow.
```

The difference is striking. The first response, while accurate to its knowledge base, offers only generic information. The second response, enriched with operational data from Materialize, delivers precisely what the customer needs to know: current availability, personalized delivery options, and actionable next steps. This transformation from general knowledge to specific, actionable insight is what drives conversion rates and customer satisfaction.

Incremental View Maintenance at Scale

At the heart of solving this challenge lies a fundamental shift in how we process and maintain data views. Traditional systems face a brutal choice: either recompute entire result sets when data changes, leading to high latency and resource consumption, or accept eventual consistency and deal with the resulting complexity. Materialize takes a different approach, leveraging sophisticated incremental view maintenance to process only the necessary updates while maintaining strong consistency guarantees.

Consider this view definition, which maintains an active customer profile combining transaction history, current activity, and segmentation logic:

text

```
CREATE MATERIALIZED VIEW customer_360 AS
WITH recent_orders AS (
    SELECT
        customer_id,
        COUNT(*) as order_count,
        SUM(order_amount) as total_spent,
        MAX(order_time) as last_order_time
    FROM orders
    WHERE order_time >= MZ_NOW() - INTERVAL '30 days'
    GROUP BY customer_id
),
customer_segments AS (
    SELECT
        customer_id,
```

```
        CASE
             WHEN total_spent > 10000 THEN 'platinum'
             WHEN total_spent > 5000 THEN 'gold'
             ELSE 'standard'
         END as segment,
         order_count,
         last_order_time
    FROM recent_orders
)
SELECT
    c.customer_id,
    c.email,
    c.signup_date,
    cs.segment,
    cs.order_count,
    cs.last_order_time,
    i.items_in_cart,
    i.cart_value
FROM customers c
LEFT JOIN customer_segments cs ON c.customer_id =
cs.customer_id
LEFT JOIN active_shopping_carts i ON c.customer_id =
i.customer_id;
```

When this view is created, Materialize doesn't just execute the query once. Instead, it builds an internal representation that tracks dependencies and maintains efficient indexes. As new orders arrive, shopping carts update, or customer information changes, Materialize automatically updates only the affected portions of the view. This approach delivers consistent sub-second query response times even as data volumes and complexity grow.

The Technical Foundation: How Materialize Works

Materialize does all of this by reimagining of how databases process changing data. While traditional systems struggle with balancing freshness against performance, Materialize addresses these limitations through two key innovations.

Differential Dataflow: Making Complex SQL Efficient

The first innovation of Materialize comes from differential dataflow - a computation model that rethinks how we process changing data. Instead of doing heavy computation when you need answers, differential dataflow does a small amount of work every time your data changes, precisely tracking how each change affects your results. By understanding exactly what changed, the system updates only what's necessary - whether you're doing complex multi-way joins, window functions, or even recursive queries for hierarchical data.

Think of it like keeping your house organized: rather than letting things pile up and doing a big cleanup when guests arrive, you do a little work each time something changes. When you need answers, they're already there. This "write-time" approach means queries return instantly, making it perfect for AI systems that need fast, consistent access to complex derived data.

And to keep the system cost effective, differential dataflow shares state between operators. Each piece of computation happens exactly once and gets reused wherever needed. For example, if multiple views need the same join result, that work is shared. This means even sophisticated materialized views stay efficient as your data and query complexity grow.

Virtual Time: Consistency Without Compromise

Materialize's virtual time system introduces a structure that solves one of the hardest problems in distributed systems: maintaining consistency across components without forcing them to synchronize. By placing every update on a common timeline with explicit timestamps, different parts of the system can process data at their own pace while still guaranteeing consistent results.

For AI applications that need to combine data from multiple sources - customer records, inventory levels, real-time signals - this means getting fresh, consistent data without the traditional performance overhead of coordination. Every query sees a correct view of your data at a specific point in time, even as the underlying systems update independently.

Beyond Consistency: Joins Across Data Sources

The true power of this approach becomes apparent when we consider real-world AI applications that need to combine data from multiple sources. Take an e-commerce recommendation engine that must merge active user behavior, product catalog data, inventory levels, and pricing rules. Traditional architectures would require complex ETL pipelines or accept significant delays between updates. Materialize instead maintains materialized views that span multiple data sources while preserving incremental updates and strong consistency guarantees.

Here's how this works in practice:

text

```
CREATE MATERIALIZED VIEW product_recommendations AS
WITH user_interests AS (
    SELECT
        user_id,
        product_id,
        COUNT(*) as view_count,
        MAX(view_time) as last_viewed
    FROM user_product_views
    WHERE view_time >= MZ_NOW() - INTERVAL '24 hours'
    GROUP BY user_id, product_id
),
product_scores AS (
    SELECT
        ui.user_id,
        ui.product_id,
        p.category,
        p.brand,
        ui.view_count,
        i.available_quantity,
        pr.current_price,
        (ui.view_count * 0.3 +
         CASE WHEN i.available_quantity > 0 THEN 0.4 ELSE 0
END +
         CASE WHEN pr.current_price < pr.list_price THEN 0.3
ELSE 0 END
        ) as recommendation_score
    FROM user_interests ui
    JOIN products p ON ui.product_id = p.id
    JOIN inventory i ON p.id = i.product_id
    JOIN pricing pr ON p.id = pr.product_id
)
SELECT
    user_id,
    product_id,
    category,
    brand,
    recommendation_score,
```

```
    ROW_NUMBER() OVER (
        PARTITION BY user_id
        ORDER BY recommendation_score DESC
    ) as rank
FROM product_scores;
```

This view combines streaming user behavior data with reference data from multiple databases, maintaining recommendations that reflect both user interests and business constraints like inventory availability and pricing rules. When a user views a product or inventory levels change, only the affected recommendations are recomputed, ensuring efficient resource utilization while maintaining consistency.

Orchestrating the AI Ensemble

The most sophisticated AI implementations often involve multiple specialized agents working in concert. A modern e-commerce platform might employ separate AI systems for inventory optimization, dynamic pricing, fraud detection, customer service, and delivery routing. Without a unified data foundation, these agents can work at cross-purposes, leading to situations where the pricing engine drops prices on items the inventory system knows are running low, or the customer service bot promises availability it can't verify.

Materialize addresses this orchestration challenge by providing a consistent data plane that all agents can trust. Consider this view that maintains a unified operational state:

text

```
CREATE MATERIALIZED VIEW operational_state AS
SELECT
    p.product_id,
    p.name,
    p.category,
    i.quantity_available,
    i.reorder_point,
    pr.current_price,
    pr.min_price,
    pr.max_price,
    o.pending_orders,
    o.shipping_backlog,
    r.regional_demand,
```

```
    ROW_NUMBER() OVER (
        PARTITION BY user_id
        ORDER BY recommendation_score DESC
    ) as rank
FROM product_scores;
```

This view combines streaming user behavior data with reference data from multiple databases, maintaining recommendations that reflect both user interests and business constraints like inventory availability and pricing rules. When a user views a product or inventory levels change, only the affected recommendations are recomputed, ensuring efficient resource utilization while maintaining consistency.

Orchestrating the AI Ensemble

The most sophisticated AI implementations often involve multiple specialized agents working in concert. A modern e-commerce platform might employ separate AI systems for inventory optimization, dynamic pricing, fraud detection, customer service, and delivery routing. Without a unified data foundation, these agents can work at cross-purposes, leading to situations where the pricing engine drops prices on items the inventory system knows are running low, or the customer service bot promises availability it can't verify.

Materialize addresses this orchestration challenge by providing a consistent data plane that all agents can trust. Consider this view that maintains a unified operational state:

text

```
CREATE MATERIALIZED VIEW operational_state AS
SELECT
    p.product_id,
    p.name,
    p.category,
    i.quantity_available,
    i.reorder_point,
    pr.current_price,
    pr.min_price,
    pr.max_price,
    o.pending_orders,
    o.shipping_backlog,
    r.regional_demand,
```

```
    c.segment,
    o.latest_order_status,
    t.open_ticket_count,
    i.items_in_cart
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
LEFT JOIN tickets t ON c.customer_id = t.customer_id
LEFT JOIN shopping_carts i ON c.customer_id = i.customer_id;
```

For a network of inventory management agents:

```text
CREATE MATERIALIZED VIEW inventory_insights AS
SELECT
    product_id,
    warehouse_id,
    current_stock,
    reorder_point,
    CASE
        WHEN current_stock < reorder_point THEN 'reorder'
        WHEN current_stock < safety_stock THEN 'warning'
        ELSE 'normal'
    END as stock_status
FROM inventory_levels;
```

In both cases, Materialize maintains these views incrementally, ensuring that whether it's a language model seeking context for a response or an autonomous agent making inventory decisions, the necessary information is instantly available without redundant computation. This architectural pattern transforms how AI systems interact with operational data, making actionable data intelligence sustainable at scale.

Scaling AI: The Challenge of Agentic Load

As organizations expand their AI initiatives, they quickly encounter a hidden scaling challenge. Each new AI agent—whether handling fraud detection, inventory optimization, or dynamic routing—introduces additional load on production systems. Every query or computation adds latency, consumes resources, and often duplicates work already being performed by other agents. The

cumulative effect can quickly become unsustainable, turning what should be a transformative technology into a operational burden.

Materialize fundamentally reimagines how AI agents interact with operational data. Instead of each agent repeatedly executing expensive computations against production systems, Materialize shifts this computational burden to its specialized actionable data data engine. Through intelligent pre-computation and incremental maintenance, it transforms what would be repeated, resource-intensive queries into near-instantaneous lookups against continuously updated views.

Consider a real-world example: multiple AI agents monitoring inventory levels across a network of warehouses. Without proper architecture, each agent might independently query for current stock levels, recent sales, and incoming shipments —repeatedly triggering expensive joins and aggregations that strain the production database. Here's how Materialize transforms this scenario:

text

```
CREATE MATERIALIZED VIEW inventory AS
WITH current_inventory AS (
  SELECT
    product_id,
    SUM(quantity) AS total_inventory
  FROM inventory
  GROUP BY product_id
),
recent_sales AS (
  SELECT
    product_id,
    SUM(quantity_sold) AS recent_sales
  FROM sales
  WHERE sale_time >= MZ_MZ_NOW() - INTERVAL '1 hour'
  GROUP BY product_id
),
recent_shipments AS (
  SELECT
    product_id,
    SUM(quantity_received) AS recent_shipments
  FROM shipments
  WHERE shipment_time >= MZ_MZ_NOW() - INTERVAL '1 hour'
  GROUP BY product_id
)
SELECT
```

```
  p.product_id,
  p.product_name,
  p.category,
  ci.total_inventory,
  COALESCE(rs.recent_sales, 0) AS recent_sales,
  COALESCE(rsh.recent_shipments, 0) AS recent_shipments,
  CASE
    WHEN ci.total_inventory > 0
    THEN (COALESCE(rs.recent_sales, 0)::float /
ci.total_inventory)
    ELSE 0
  END AS turnover_ratio,
  RANK() OVER (
    PARTITION BY p.category
    ORDER BY
      CASE
        WHEN ci.total_inventory > 0
        THEN (COALESCE(rs.recent_sales, 0)::float /
ci.total_inventory)
        ELSE 0
      END DESC
  ) AS turnover_rank
FROM products p
  LEFT JOIN current_inventory ci ON p.product_id =
ci.product_id
  LEFT JOIN recent_sales rs ON p.product_id = rs.product_id
  LEFT JOIN recent_shipments rsh ON p.product_id =
rsh.product_id
ORDER BY p.category, turnover_rank;
```

This view encapsulates complex business logic—combining current inventory levels, recent sales trends, and incoming shipments to calculate turnover ratios and rankings. Rather than having each AI agent independently compute these metrics, Materialize maintains this view incrementally, updating it automatically as new data arrives. Every agent accessing this view gets instant, consistent access to the latest insights without additional computational overhead.

The impact on system performance and scalability is transformative. Organizations can deploy more AI agents, tackle more complex use cases, and process higher data volumes without the traditional exponential increase in infrastructure costs. When market conditions shift, online orders spike, or supply chain disruptions occur, the system maintains its responsiveness—each materialized view updates automatically, providing agents with current insights at near-constant latency.

This architectural approach fundamentally changes the economics of AI deployment. Instead of each new agent adding computational burden to production systems, organizations can scale their AI initiatives while maintaining lean, efficient operations. The result is an AI infrastructure that delivers not just actionable data, but sustainable, cost-effective performance at scale.

The Path Forward

The journey to effective AI implementation begins with identifying where stale or inconsistent data limits your current capabilities. Perhaps it's a customer service system that can't access current order statuses, a recommendation engine working from outdated inventory data, or trading algorithms operating on delayed market feeds. These pain points are opportunities for transformation, chances to demonstrate how fresh, consistent data can elevate AI from an interesting technology to a reliable driver of business value.

The future of AI isn't just about bigger models or more sophisticated algorithms. It's about grounding those capabilities in reality, ensuring every interaction, recommendation, and decision reflects the current state of your business. With the right data foundation, that future is within reach today.