WHITEPAPER

# Building the Connected Enterprise

## Strategy Mosaic Model Linking

## Best Practice Guidelines

# Table of contents

# Executive Summary

## Closing the Domain Gap with Mosaic Model Linking

Every business runs on its own internal language. Long before AI entered the conversation, organizations were already wrestling with the challenge of turning complex, technical data into something people across the business could use and agree on. It takes a new employee three to six months to grow comfortable with a company's unique terminology, and up to nine months to fully navigate its data ecosystem. The gap between raw data and shared organizational understanding has always been one of the most expensive inefficiencies in enterprise analytics.

Semantic modeling closes that gap by connecting business definitions to data — transforming technical tables and columns into a shared organizational language that everyone can query, trust, and build on. As organizations grow, however, so does the complexity of that language. Different teams develop different models, different definitions, and different ways of describing the same business reality.

This is where Model Linking becomes essential.

> "Semantic modeling often requires balancing flexibility and control. Large, centralized models are hard to maintain and evolve, while isolated models make cross-domain analysis difficult. Model Linking bridges the gap by keeping models decoupled but analytically connected."

## What Mosaic Model Linking Delivers:

Mosaic Model Linking connects independently managed semantic models so they can be queried through one connected semantic layer. Instead of forcing all data and logic into a single monolithic model, Mosaic enables domain-specific models to be developed separately and linked when needed.

Benefits of adopting Model Linking:

- **Up to 90% reduction** in time to deliver new cross-domain analysis
- **Up to 90% fewer** duplicate definitions for shared entities like Employee, Customer, and Date
- **Materially lower** maintenance burden when upstream schemas change

# Introduction

## Enterprise Analytics Is Failing at the Seams

Most organizations operate across natural domain boundaries that segment their organizational structure into specific disciplines. HR owns employee data, Finance owns cost data, and Operations owns productivity data; each of these teams builds semantic models to serve their specific reporting needs.

The challenge emerges when independently managed models need to answer a shared business question. Without Model Linking, crossing those model boundaries requires additional integration, data engineering, or manual reconciliation. Mosaic Model Linking connects independently managed semantic models through shared business entities, creating a reliable path for cross-domain analysis without sacrificing governance, performance, or flexibility.

Each model remains independently owned and managed by the team responsible for it, while shared business entities such as Employee ID, Customer ID, and Date connect those models through a shared semantic layer.

## The Core Design Challenge

- Large, centralized models are hard to build, harder to maintain, and impossible to govern at scale
- Isolated domain models make cross-domain analysis nearly impossible
- Every custom integration between models creates new duplication, new risk, and new maintenance burden

This document shows how Strategy Mosaic Model Linking provides a middle ground by allowing models to remain decoupled while still being analytically connected, and covers:

- ✦ **Why Model Linking exists and what problem it solves**
- ✦ **How security is enforced across linked models**
- ✦ **Known limitations and feature coverage boundaries**

✦ **How connected semantic models support governed analytics and AI**

# I. Why Model Linking

## The Case for a Modular Semantic Architecture

A common challenge in semantic modeling is balancing flexibility with control. On one side there is the need for flexibility for teams to build models that reflect their own data, metrics, and governance requirements. On the other side, there's the need for control, where the business needs consistent, governed, and cross-domain analysis.

The two traditional responses to this tension each create their own problems:

**Response 1:** Building one large model that contains everything. Cross-domain queries work, but the model becomes increasingly difficult to maintain and evolve. This causes change windows to take longer, ownership becomes ambiguous, and domain expertise gets buried inside a structure no single team can fully understand or safely modify.

**Response 2:** Let each team build and own their model independently. Governance is clean, but cross-domain analysis becomes nearly impossible. Every cross-domain question requires a new integration, a new data engineering effort, or a manual reconciliation between two separately governed result sets.

Model Linking provides a middle ground by allowing models to remain decoupled while still being analytically connected.

## What Model Linking enables:

- Cross-domain analytics without rebuilding models
- Modular development and ownership by domain teams
- Reuse of shared business entities and hierarchies
- Faster iteration and reduced duplication of logic

Model Linking also plays a key role beyond cross-domain analytics. Shared entities enable cross-domain analysis while aligning with data mesh principles. For Import models, linking with live models enables hybrid execution by combining the performance of imported data with the freshness of live connections.

# II. How Model Linking Works

## Architecture Fundamentals

At its core, Model Linking is based on shared business entities. When two models contain common attributes (such as Employee ID, Customer ID, or Date) Mosaic uses those attributes to establish relationships between the models.

Once linked, queries can span across models as if they were part of a single semantic layer. Mosaic handles the complexity behind the scenes by coordinating query planning and execution across models.

### *In practice, this means:*

- Queries are resolved across multiple models transparently
- Execution is pushed down to underlying sources when possible
- Import and live data can be combined in a single query
- Users do not need to know where data lives, only how it relates

Instead of forcing all data and logic into a single, monolithic model, Mosaic Model Linking enables a modular approach where domain-specific models can be developed separately and linked when needed.

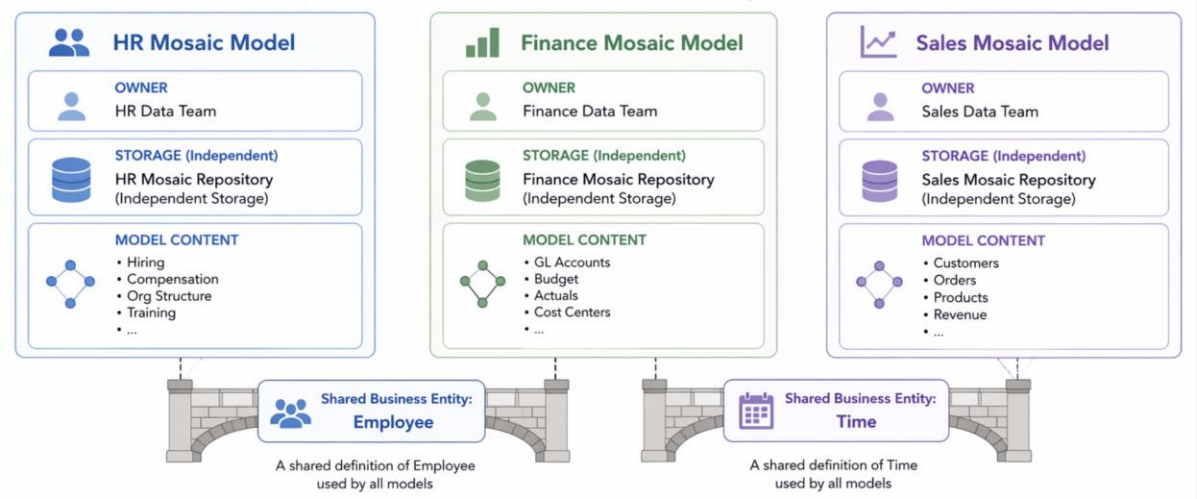


**Figure 1.** Modular Mosaic Models connect through shared business entities

This design reflects how organizations operate: different teams own different data assets, but business questions often span those boundaries. Model Linking bridges that gap without sacrificing governance, performance, or flexibility.

## Creating and Managing Links in Mosaic Studio

Model authors can add existing Mosaic Models and connect attributes that represent the same business entity. Mosaic Studio can suggest potential links, while authors retain control over which relationships are applied.

A typical workflow includes:

1. Add the participating Mosaic Models.
2. Review suggested links between attributes.
3. Confirm that the attributes represent the same business concept and analytical grain.
4. Select the attributes that participate in the link.
5. Define the linked attribute alias.
6. Select the authoritative value source.
7. Apply the link and validate representative queries before publication.

Links can also be created or edited manually through **Manage Links**. A link should be treated as a governed analytical relationship, not merely as a match between similarly named columns.

### Connected Context for Analytics and AI

Connected semantic models can also provide governed business context for AI experiences that need to reason across domains. The same shared entities, business definitions, and security policies used for analytics can support AI access without requiring teams to recreate semantic logic in a separate AI layer.

# II. Continued: How Model Linking Works

## Example: The Cross-Model Query

Consider an organization that assigns sales revenue to individual sales representatives. A user asks for sales revenue by employee department. Revenue by sales representative resides in the Sales model, while the Employee-to-Department mapping resides in the HR model.

With Model Linking, Mosaic resolves this in four steps:

**Step 1:** Identify Employee ID as the shared business entity linking the two models

**Step 2:** Query the HR model for the Employee to Department mapping

**Step 3:** Query the Sales model for revenue grouped by Employee

**Step 4:** Join the two result sets on Employee ID and return a single aggregated result to the user

**Result:** The user sees one table of Revenue by Department. They do not need to know which model supplied which column, and the ownership of each model remains unchanged.

## What does this mean architecturally?

The result is analytically unified, and the ownership remains structurally separate. The HR team continues to own and govern the Employee-to-Department mapping, while the Sales team continues to own and govern revenue. Neither team's model changes because of the linked query, and the user receives a single, coherent answer.

# III. Security and Governance

## How Security Propagates Across Linked Models

One of the most important aspects of Model Linking is that governance is preserved across models. When queries span independently governed models, the applicable security policies remain active throughout the linked analysis.

**Row-Level Security (RLS)**

When row-level security filters are defined in a participating model, those filters continue to apply when that model is used through a linked query. This helps ensure that data access rules remain consistent even when an analysis spans multiple models.

For example, linking an HR model to a Finance model does not give Finance users broader access to employee records. The HR model continues to enforce its employee-level restrictions, while the Finance model continues to govern access to financial data.

**Object-Level Security (OLS)**

Object-level restrictions, such as access controls on specific attributes or metrics, also remain in force across linked models. If a user does not have access to an object in one of the participating models, linking that model to another does not expose the restricted object.

When multiple policies apply to a linked query, the effective result reflects all applicable restrictions. A user must also have access to every underlying attribute that participates in a linked attribute.

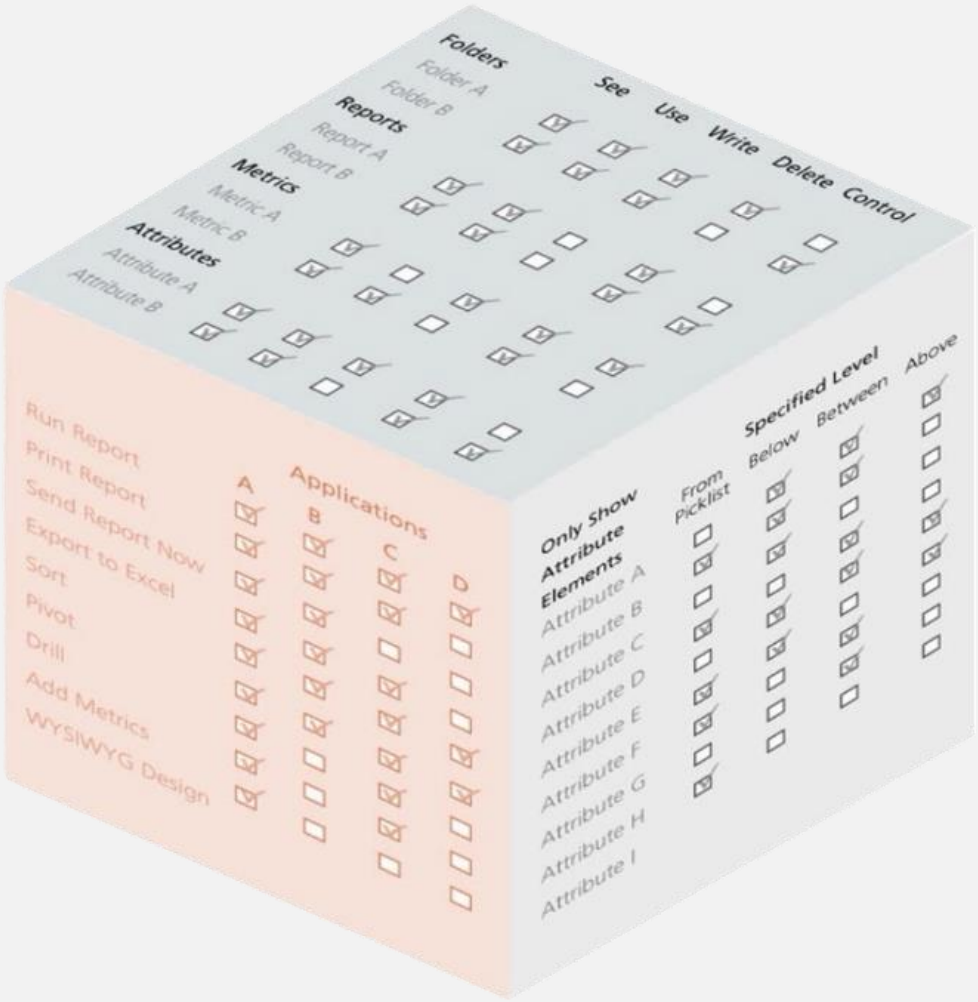


**Figure 2.** Row-level and object-level controls govern data and model access

**Key behaviors:**

- RLS defined in participating models remains enforced during linked analysis.
- OLS restrictions remain active and consistently applied.
- Security logic does not need to be recreated in every model that uses the linked data.
- Access reflects the applicable controls across the models participating in the query.
- Governance remains consistently enforced while model ownership stays decentralized.

# III. Continued: Security and Governance

## Why This Matters for Regulated Industries

In most analytics architectures, adding a new integration point between models also adds a new potential governance gap. When security is defined independently in each model, the boundaries between those models become the places where access controls are most likely to break down. This can result in policies becoming more restrictive than intended and blocking users from data they are permitted to see or becoming less restrictive than intended and exposing data they should not have access to at all.

**Model Linking is designed to work in the opposite direction:** It does not introduce security gaps. Instead, it strengthens governance by making it consistent across the entire semantic layer, an essential requirement for regulated industries.

**Why Consistent Governance Matters in a Distributed Architecture:** Model Linking allows model ownership to remain distributed while security remains consistently enforced across linked analysis. Each domain team continues to define and maintain the controls associated with its own data.

When a query combines objects from multiple domains, the applicable policies remain active throughout the linked analysis. This allows governance and compliance teams to maintain consistent controls without requiring every domain to surrender ownership to a centralized modeling team.

Policies stay close to the data and business context they govern, while Model Linking preserves those controls across the connected semantic layer.

**Governance properties that Model Linking maintains:**

**1. Consistent enforcement:** Security policies continue to apply when data and objects are accessed through linked analysis.

**2. Local policy ownership:** The team responsible for the data continues to define and maintain the policies that govern it.

**3. Governed interoperability:** Models can participate in cross-domain analysis without transferring ownership or creating an ungoverned integration layer.

**4. Reduced duplication:** Other teams do not need to recreate security rules when consuming linked data.

*For regulated industries such as healthcare, financial services, and government, Model Linking is more than an architectural convenience. It addresses governance at the structural level.*

# IV. Implementation Guidelines

## Design Principles

The Mosaic Model Linking Best Practice Guidelines establish three foundational design principles for building a linked semantic architecture that scales. Each principle addresses a specific failure that emerges when linking is implemented without architectural discipline.

1. **Modular Architecture**

Keep models modular and purpose-built rather than monolithic.

Reusable hierarchies should be separated into their own base models. Time, Geography, and Organizational hierarchies can be defined once as a base model and linked to multiple fact models. This avoids duplication and ensures consistent reporting across every model that links to the shared hierarchy.

2. **Domain Ownership**

Different teams can maintain their own models: HR for employee data, Finance for cost data, Operations for productivity data. Shared entities enable cross-domain analysis while aligning with data mesh principles.

3. **Governed Interoperability**

Ownership remains decentralized while interoperability is maintained.

The HR team does not need to grant the Finance team ownership of employee data to support cross-domain queries. HR continues to own and govern the HR model, while Finance continues to own and govern the Finance model. Model Linking through the shared Employee entity connects them analytically without changing who governs what.

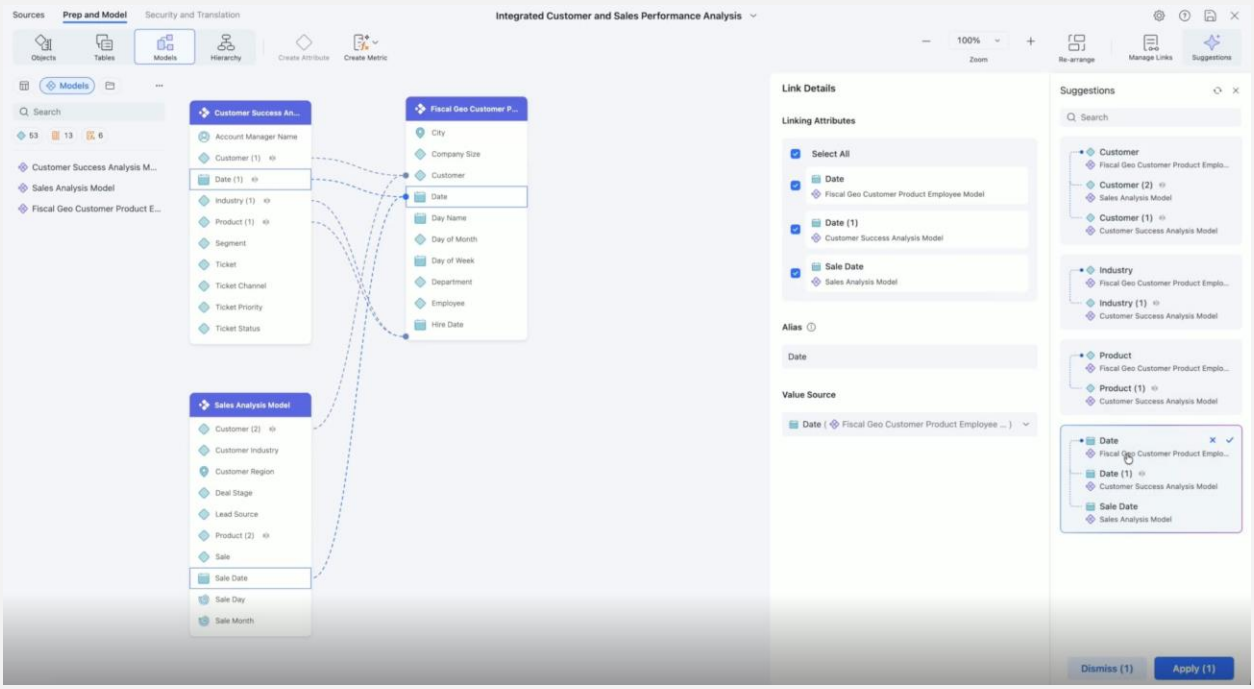


**Figure 3.** Mosaic Studio suggests and manages links across Mosaic Models

# IV. Continued: Implementation Guidelines

## Flexible Data Access and Modernization Strategies

Mosaic supports multiple patterns for connecting models and reusing existing semantic investments. Choosing the right approach depends on data freshness requirements, query performance targets, model ownership, and the current state of the semantic layer.

Model Linking and Mosaic Model-to-Schema mapping are related but distinct capabilities. Model Linking connects independently managed Mosaic Models for cross-model analysis, while Schema mapping allows Mosaic Model attributes to reuse established Mosaic Schema definitions and security.

**Multiple live models:** Enable multi-source federation across live-connect data sources.

**Import plus live models:** Enable hybrid execution, combining the performance of imported data with the freshness of live connections.

**Mosaic Model-to-Schema mapping:** Enable gradual modernization while reusing existing governed semantic assets.

Organizations can map Mosaic Model attributes to established Mosaic Schema attributes. This allows new Mosaic Models to reuse centrally defined dimensions and inherit associated access controls and row-level security filters.

Rather than recreating established definitions and policies, teams can build new Mosaic Models alongside their existing semantic foundation and modernize individual domains over time.

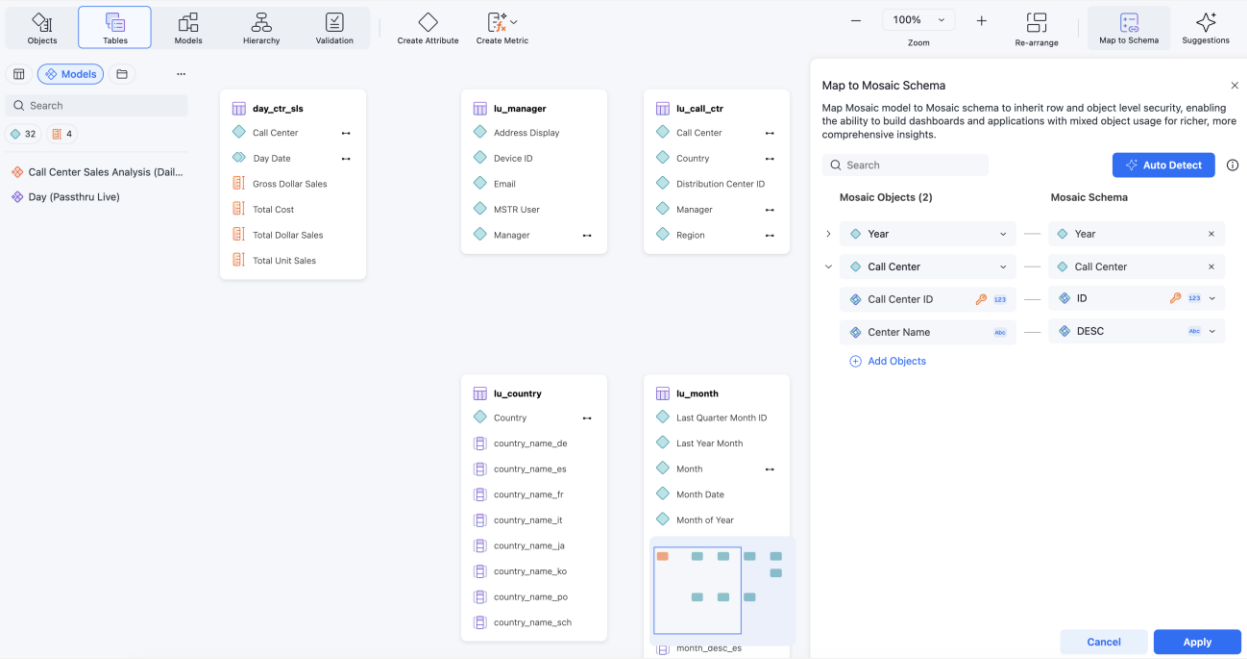


**Figure 4.** Mapping Mosaic Models to Mosaic Schema

# IV. Continued: Implementation Guidelines

## Technical Implementation

### Shared Business Entities as Stable Contracts

The technical reliability of Model Linking depends almost entirely on the quality and consistency of shared business entities. These entities are the join points between models. If they are defined inconsistently, such as through different key formats, levels of

granularity, or business meanings, queries may fail to resolve or, worse, return incorrect results without producing an obvious error.

## Requirements for Reliable Shared Business Entities

**1. Consistent business meaning**
Confirm that the linked attributes represent the same real-world concept across every participating model.

**2. Aligned analytical grain**
Confirm that the entity represents the same level of detail. For example, a corporate customer account is not equivalent to an individual customer contact.

**3. Known cardinality**
Understand whether the relationship is one-to-one, one-to-many, or many-to-many, and test how that relationship affects aggregation.

**4. Compatible keys and formats**
Align data types, key formats, case rules, leading zeros, null handling, and unmatched values.

**5. Authoritative value source**
Identify which participating model supplies the preferred lookup values and display forms for the linked attribute.

**6. Stable ownership and change control**
Assign an owner for the shared entity and coordinate changes across every team that depends on it.

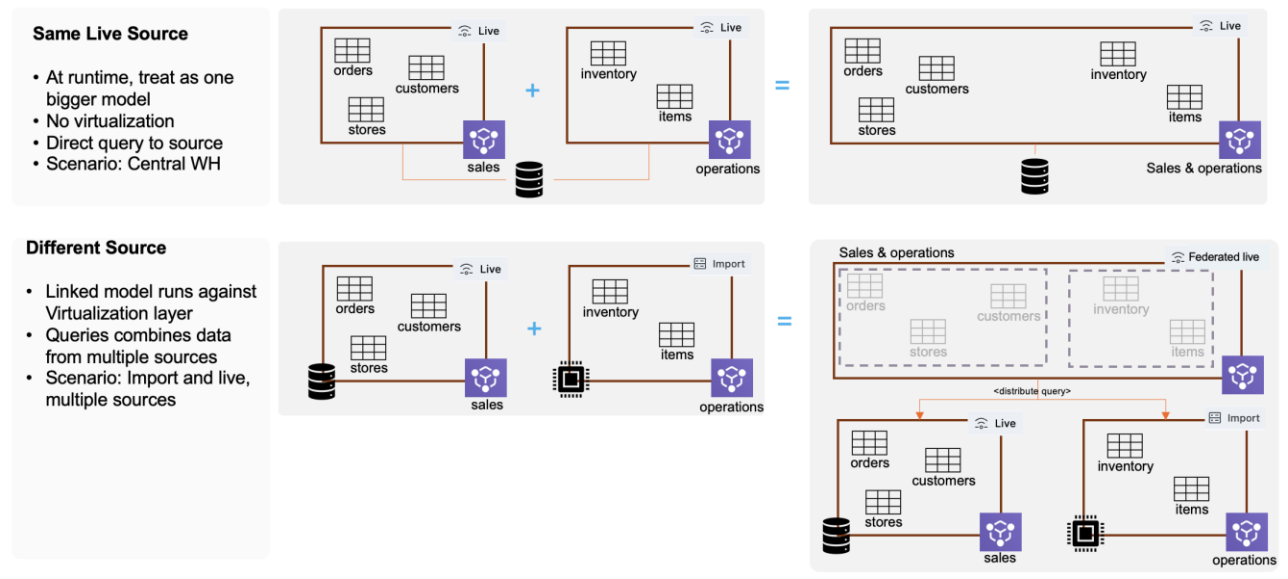


**Figure 5.** Model Linking supports same-source, federated, and hybrid execution patterns

# IV. Continued: Implementation Guidelines

## Execution Strategy Selection

Choosing the right execution strategy, including live, import, and hybrid execution, is one of the most consequential technical decisions in a Model Linking implementation. The wrong choice for a given use case creates performance problems that are difficult to diagnose and expensive to fix after the fact.

### Recommended approach:

- **Multiple models on one live source:** Model Linking can improve modularity and manageability when the models serve different domains or ownership groups. Because the models share a source, more of the query may be resolved directly by that source.
- **In-memory scenarios:** A single model may provide better performance and simpler operations when the data is tightly coupled, refreshed together, and owned by one team.
- **Live federation and hybrid scenarios:** Test performance, source workload, data movement, and memory consumption carefully. When a query cannot be fully pushed down, Mosaic may need to federate intermediate results across participating models.

*Decision factors to consider:*

- **Data freshness needs:** How current does the data need to be for the use case?
- **Query performance requirements:** What response time is acceptable for the intended users?
- **Source system capabilities:** Can the source system handle the query volume and complexity that linked queries may generate?
- **Join cardinality and result size:** How much intermediate data may need to move or be combined when the query crosses models?

## Security and Governance Practices

**Consistent Security Policies**

- Align RLS and OLS definitions across teams to avoid conflicts.
- Establish clear ownership and approval processes for shared entities.
- Document security requirements that must propagate across linked models.

# IV. Continued: Implementation Guidelines

## Change Management

Model Linking creates dependencies between models that did not previously exist. The HR model's department hierarchy was once only the HR team's concern. Once Finance and Operations models link to it through Employee ID, a change to that hierarchy affects all three models' users simultaneously.

**Three Change Management practices:**

1. **Be aware of dependencies between models when making changes**

   Before modifying a base model or a shared entity, identify every linked model that depends on it. The team making the change is responsible for understanding the downstream impact before the change is published.

2. **Validate changes across the full linked model context**

Test changes in the context of the full linked model chain, not only in the model being modified. A change that appears harmless in isolation can affect a linked query that depends on a specific attribute format, relationship, security policy, or execution path.

Use Mosaic Model Validation to review representative results and the underlying query before publication. Validate shared-entity joins, metrics, filters, security roles, and execution patterns across the participating models.

### 3. Establish communication protocols between teams managing linked models

When models are owned by different teams, there is no automatic channel for communicating change impacts. Establish that channel deliberately whether through a shared governance forum, a defined notification process, or a shared model dependency registry.

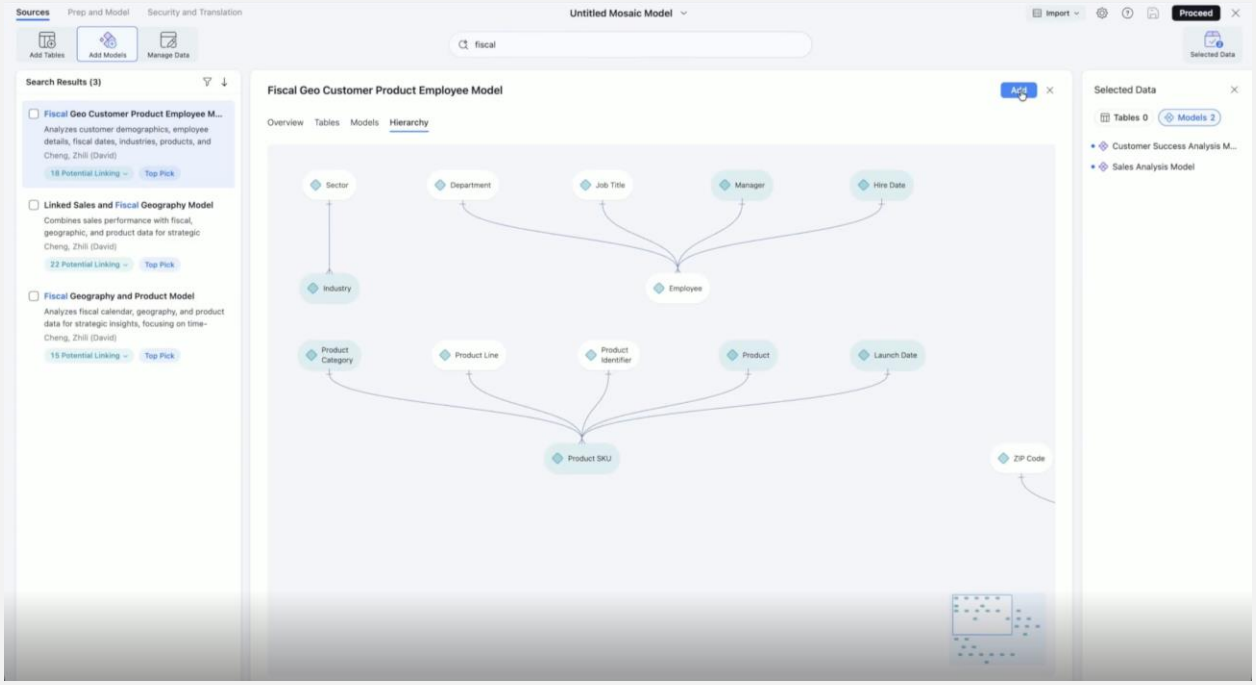


**Figure 6.** Mosaic Studio visualizes model relationships to support dependency review

# V. Common Pitfalls and Troubleshooting

Most Model Linking issues come from a small number of recurring root causes. The patterns below cover what we see most often.

### Pitfall 1: Conflicting or Unexpectedly Restrictive Security Policies

**Symptom:** A user can access an attribute in one model but cannot access the expected result when that model participates in a linked query. A cross-model query may also return an authorization error that does not occur when either model is queried independently.

**Cause:** Security policies remain active across the models and objects participating in the linked query. A restriction in one participating model may reduce the effective result.

**Fix:** Review the applicable policies across all participating models rather than troubleshooting each model in isolation. Test representative user roles after changes to row-level security, object permissions, linked attributes, or model relationships.

When troubleshooting an authorization result, confirm that the user has access to every underlying attribute participating in the linked attribute.

### Pitfall 2: Performance Issues with Multi-Model Queries

**Symptom:** A query that runs quickly against each model in isolation is unexpectedly slow when linked.

**Cause:** Execution cannot be fully pushed down, so Mosaic must federate partial results. This is most likely when linked queries span sources or execution modes and require large or high-cardinality intermediate result sets.

**Fix:** Review the query plan and the size of intermediate result sets. Consider whether frequently reused dimensions should be imported or consolidated, and evaluate appropriate indexing, aggregation, or source-side optimization.

# V. Common Pitfalls and Troubleshooting

## Known Limitations and Feature Coverage

**Unsupported Linking Types**

The following linking configurations are not supported:

- Linking on grouped or compound attribute forms

- Linking on geo-enabled attributes or attributes with compound forms (such as ID and non-ID)
- Importing the same Mosaic model twice to mimic the behavior of attribute roles

**Unsupported SQL Functions in Multi-Source Scenarios**

Some SQL functions and advanced expressions are not yet supported in multi-source or Mosaic-linked scenarios. This can result in:

- Errors or unexpected behavior when using advanced expressions
- The need to simplify or redesign certain calculations

**Feature Coverage Limitation**

- **Auto Metrics** work only for the current model. They are not applied to external or linked models.
- **Freeform SQL (FFSQL) live tables** are not supported for multi-source use cases
- **Access control lists (ACL) on linked attributes** is not fully supported. ACL for linked attributes always inherits the most restrictive governance settings from their source models. Their ACLs cannot be modified independently in Mosaic Studio.
- **Merged attribute forms for linked attributes** do not support governance or display-form modifications in the Mosaic model
- **Multi-select and bulk rename with AI** is not yet available on linked attributes
- **Mosaic Sentinel** does not yet fully support reporting on nested Mosaic-linked models. Monitoring and observation for these scenarios are limited

# VI. Summary

Mosaic Model Linking enables a flexible and scalable semantic architecture by connecting independently managed models through a connected semantic layer. It lets organizations combine domain ownership with cross-domain insight, supports multi-source and hybrid data access, and keeps governance consistently enforced across linked analysis.

In practice, Finance, Supply Chain, Sales, and other domain teams can retain ownership of their models while participating in a single, coherent analytical experience. The applicable security policies remain active when models are queried together, so governance does not loosen as the architecture grows.

By moving away from monolithic models and toward a connected semantic ecosystem, Model Linking provides a practical foundation for modern, enterprise-scale analytics. It is a foundation built for how data works across teams today.

# Prerequisites

*The guidance in this document assumes a specific level of readiness. Before applying Model Linking in a production environment, readers should have:*

1. A working understanding of semantic modeling concepts (attributes, metrics, relationships, and hierarchies).

*For details, please refer to:* https://www2.strategy.com/producthelp/Current/Mosaic/en-us/Content/mosaic_logical_modeling.htm

2. Access to the Mosaic Studio with permission to create and publish Mosaic models

*For details, please refer to:* https://www2.strategy.com/producthelp/Current/Mosaic/en-us/Content/mosaic_studio_prereqs.htm

3. Defined data governance policies, including a convention for data ownership and documented row-level and object-level security requirements.
   - *For details, please refer to the following:*
   - Row-level Security: https://www2.strategy.com/producthelp/Current/Mosaic/en-us/Content/security_filters_in_model.htm
   - Object-level Security: https://www2.strategy.com/producthelp/Current/Mosaic/en-us/Content/mosaic_object_security.htm)

4. Familiarity with your organization's data domains (for example HR, Finance, Sales) and the business entities shared across them.

# Glossary

**Semantic layer:** A business-friendly abstraction over underlying data sources that exposes attributes, metrics, and hierarchies in terms users understand.

**Semantic model:** A single unit of the semantic layer, typically owned by one team and scoped to one subject area (for example, an HR model or a Sales model).

**Model Linking:** The Mosaic capability that connects two or more semantic models through shared business entities so they can be queried as one.

**Shared business entity:** An attribute with consistent meaning and keys across multiple models (for example, Employee ID or Date), used as the join point between linked models.

**Row-Level Security (RLS):** A security mechanism that restricts which rows of data a user can see, based on attributes of the user or the data.

**Object-Level Security (OLS):** A security mechanism that restricts access to specific attributes, metrics, or other model objects.

**Live connection:** A model execution mode in which queries are issued to the source system in real time rather than against a cached copy.

**Import:** A model execution mode in which data is loaded into Mosaic in advance for fast query performance; the counterpart to a live connection.

**Hybrid execution:** A query plan that combines Import and live models in a single result, balancing performance and data freshness.

**Federation:** Executing a single logical query across multiple underlying data sources.