

How to build ontology from your data lineage

Build a governed ontology from the lineage and semantic mappings already in your data estate, so every business concept is grounded in a real asset.

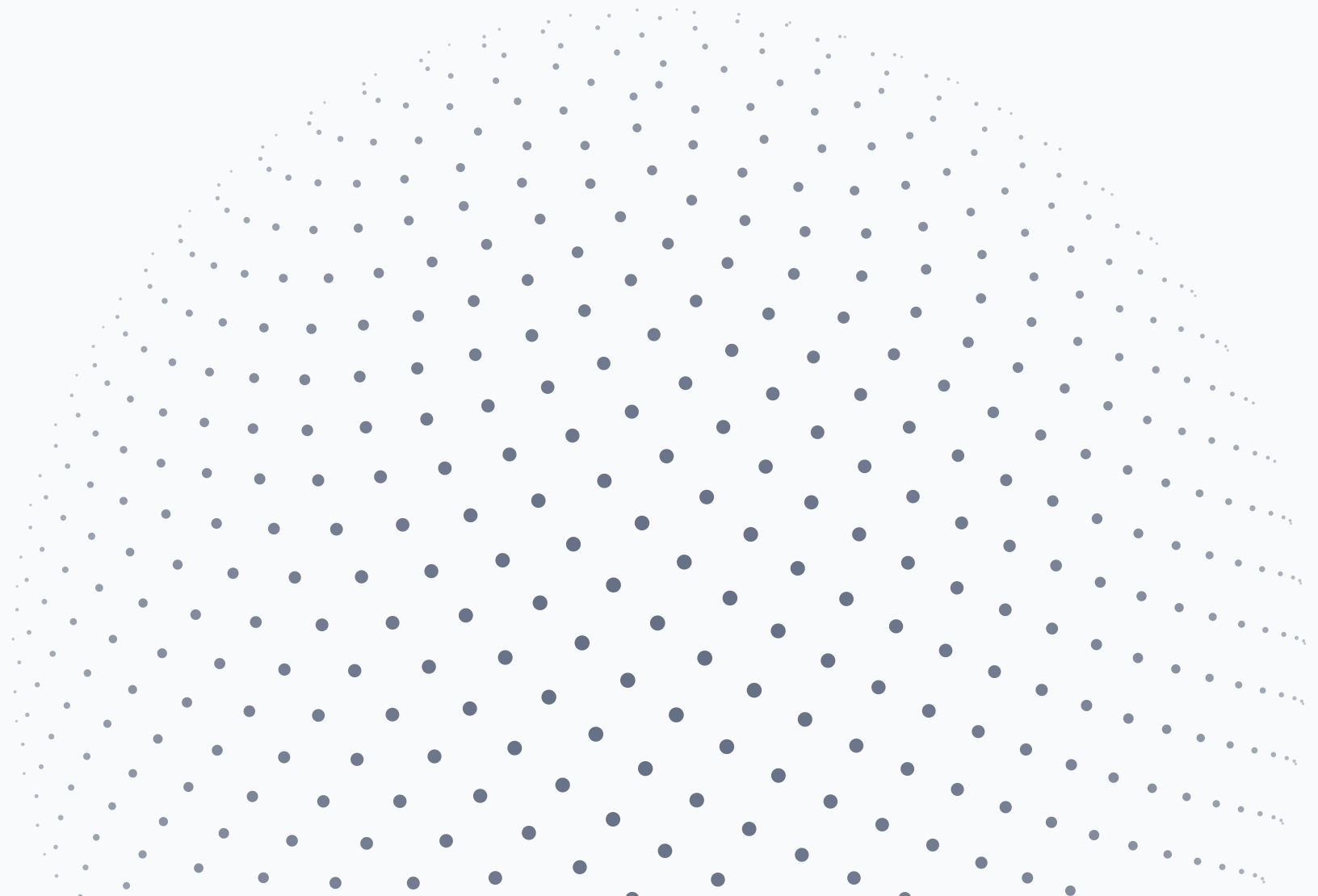


Table of Contents

A document is a claim about the system	03
The estate already describes itself, in code	03
Bottom-up means the concept starts attached	04
Binding keeps the term and the metric connected	05
What the agent receives	05
Keeping it honest after day one	07
Where this lives	07
Where to start	09

The ontology project delivered in nine days, and the speed was the problem.

A governance team pointed an LLM at a folder of policy manuals, data dictionaries, and a compliance handbook, and got back a term list longer than anything the team had ever documented by hand. Entities, relationships, hierarchies, definitions written in confident prose.

Then an architect asked where the term "eligible member" came from. Which table, which column, which transformation. The graph had a definition of "eligible member" and no record of where it lived.

A document is a claim about the system

Top-down ontology construction inherits the accuracy of whatever it reads.

Policy manuals describe the intent of a system at the moment somebody wrote them down. Data dictionaries describe a schema as of the last quarter anyone maintained them. Both drift away from the systems they describe, quietly, because nothing forces them to stay in sync.

An LLM reading those documents produces a confident model of the business as documented. What runs in production is the business as implemented. When those two disagree, the ontology has no way to know, and neither does the agent reading it.

That gap is where AI reliability breaks. The model reasons correctly against a description that stopped being true a year ago.

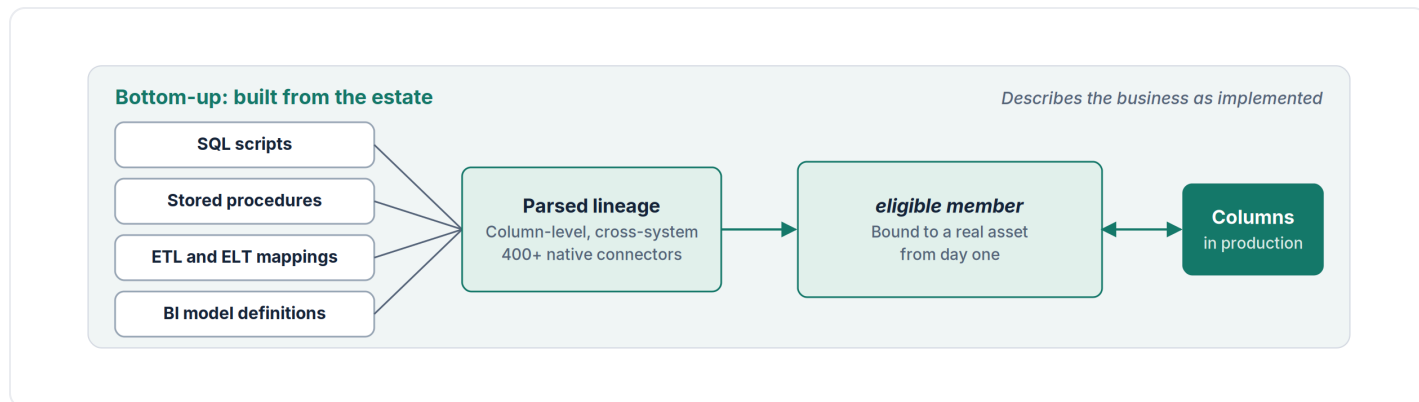


The estate already describes itself, in code

The artifacts that decide what your data means are the ones that execute: SQL scripts, stored procedures, ETL and ELT mappings, BI model definitions, the calculation logic inside a semantic model.

MetaKarta computes lineage by parsing those artifacts directly. Column-level, cross-system, across 400+ native connectors that reach from legacy on-premises platforms through modern cloud. Lineage by design, not by observation.

That parsed lineage is a description of your business written by the systems running it. It's also unreadable to the business, which is what an ontology is for.

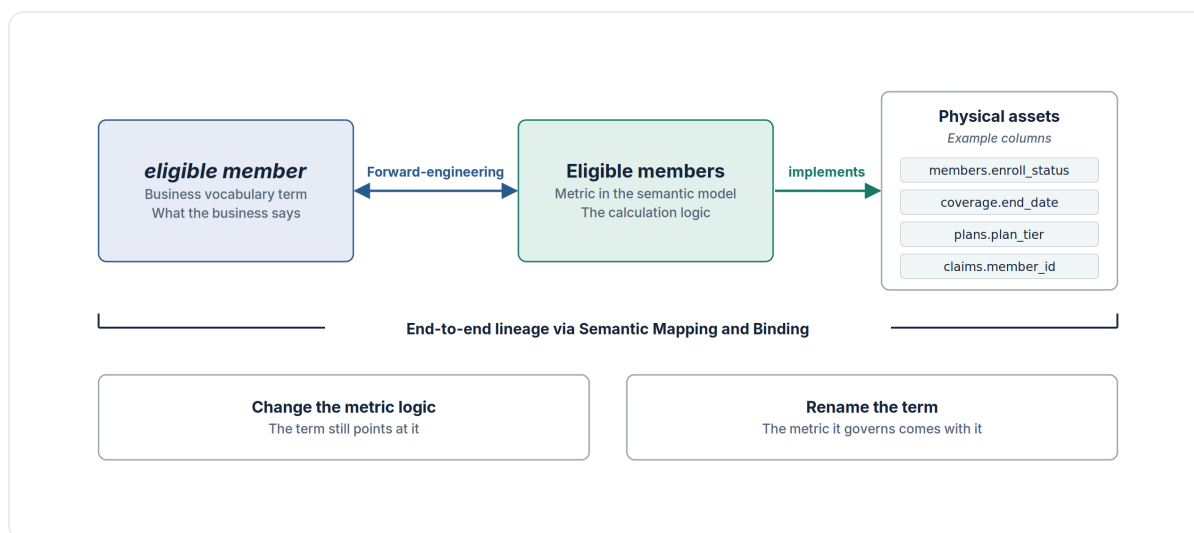


Bottom-up means the concept starts attached

Ontology Modeling builds from what's already in the shared metadata repository: parsed lineage, semantic mappings, the physical assets underneath both.

You define business entities, relationships, and hierarchies as a governed ontology model, and MetaKarta helps you populate it from the estate. It reads lineage and formulas and generates term documentation automatically, so the first draft of a concept arrives with its implementation already attached.

Every concept is bound to a real asset from day one. A top-down concept is a claim waiting to be checked. A bottom-up concept comes with the check already done, because the thing it points at is what generated it.



Binding keeps the term and the metric connected

Semantic Mapping & Binding links each business vocabulary term to the semantic model and the physical assets that implement it, which gives you end-to-end lineage from the word a business user says to the column a query reads.

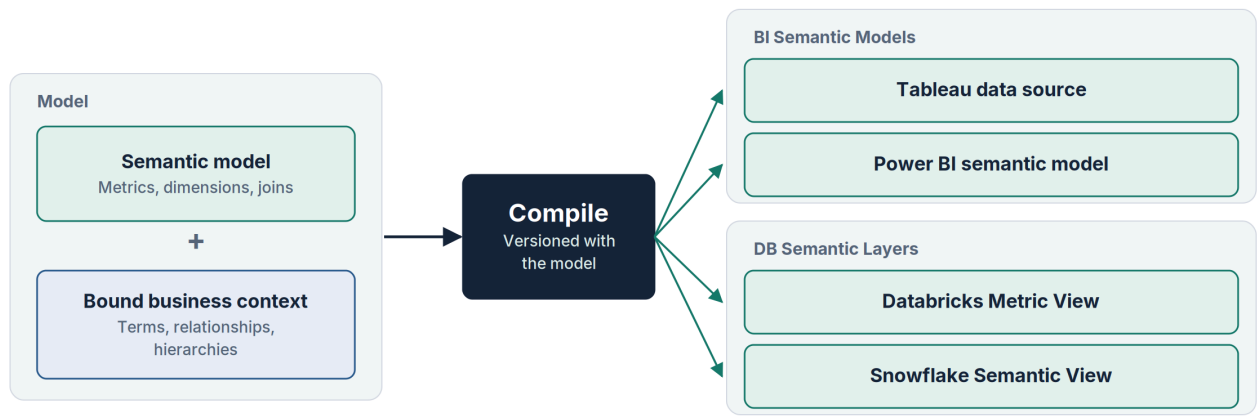
Ontology Forward-Engineering does the same work in the other direction. Define a semantic link from a vocabulary term to the metric that implements it, and the two stay connected as both evolve. Change the metric logic and the term still points at it. Rename the term and the metric it governs comes with it.

The result is a vocabulary that stays integrated with the model, versioned alongside it and compiled alongside it.

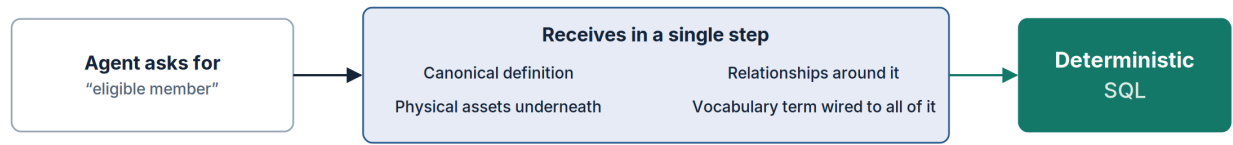
What the agent receives

In MetaKarta, the ontology compiles with the model. Nothing gets assembled at runtime.

When a semantic model compiles into a Snowflake Semantic View, a Databricks Metric View, a Power BI semantic model, or a Tableau data source, the business context bound to it compiles alongside. Meaning travels with the definition, all the way into the warehouse and the BI tool.



Meaning travels with the definition into the warehouse and the BI tool



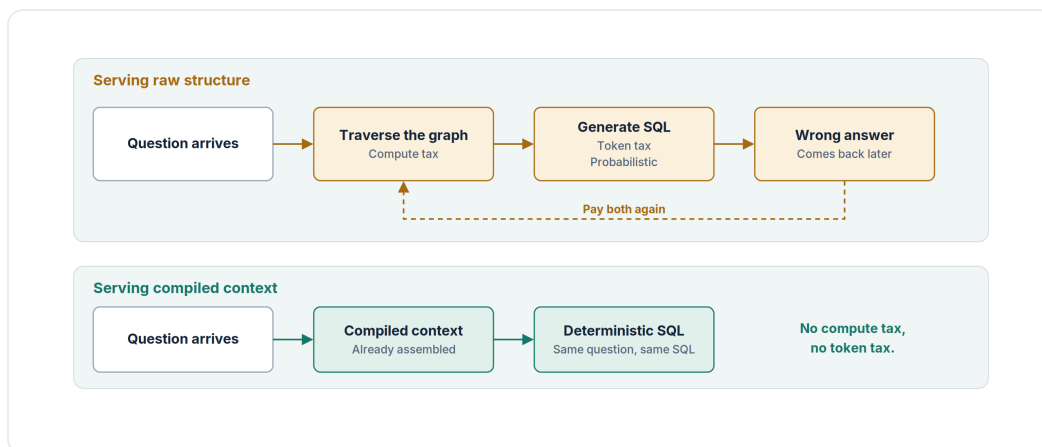
Compiled context, deterministic SQL. No compute tax, no token tax.

A semantic model plus its bound business context compiling into Tableau and Power BI semantic models and Databricks and Snowflake semantic layers, with an agent receiving the definition, relationships, and physical assets in one step.

An agent asking for "eligible member" gets the canonical definition, the relationships around it, the physical assets underneath, and the vocabulary term wired to all of it. No graph traversal to assemble meaning while the question is waiting. No compute spent walking a structure to guess at intent.

Serving an agent raw structure costs compute to traverse it and tokens to generate probabilistic SQL, then costs both again when the answer comes back wrong.

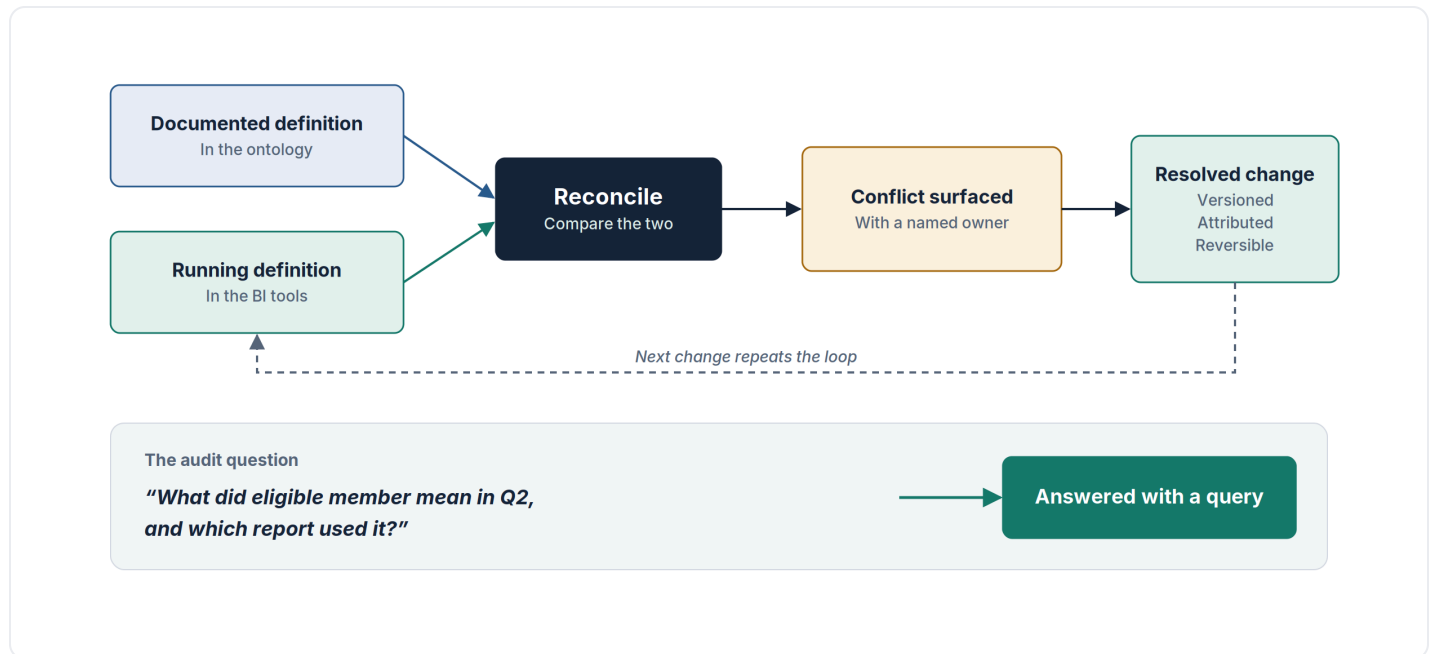
Compiled context produces deterministic SQL. No compute tax, no token tax.



Keeping it honest after day one

An ontology built once and left alone drifts the same way a data dictionary does.

MetaKarta reconciles the ontology against what your BI tools actually implement. Where the documented definition and the running definition disagree, the disagreement surfaces as a conflict somebody owns, while it's still a modeling question.

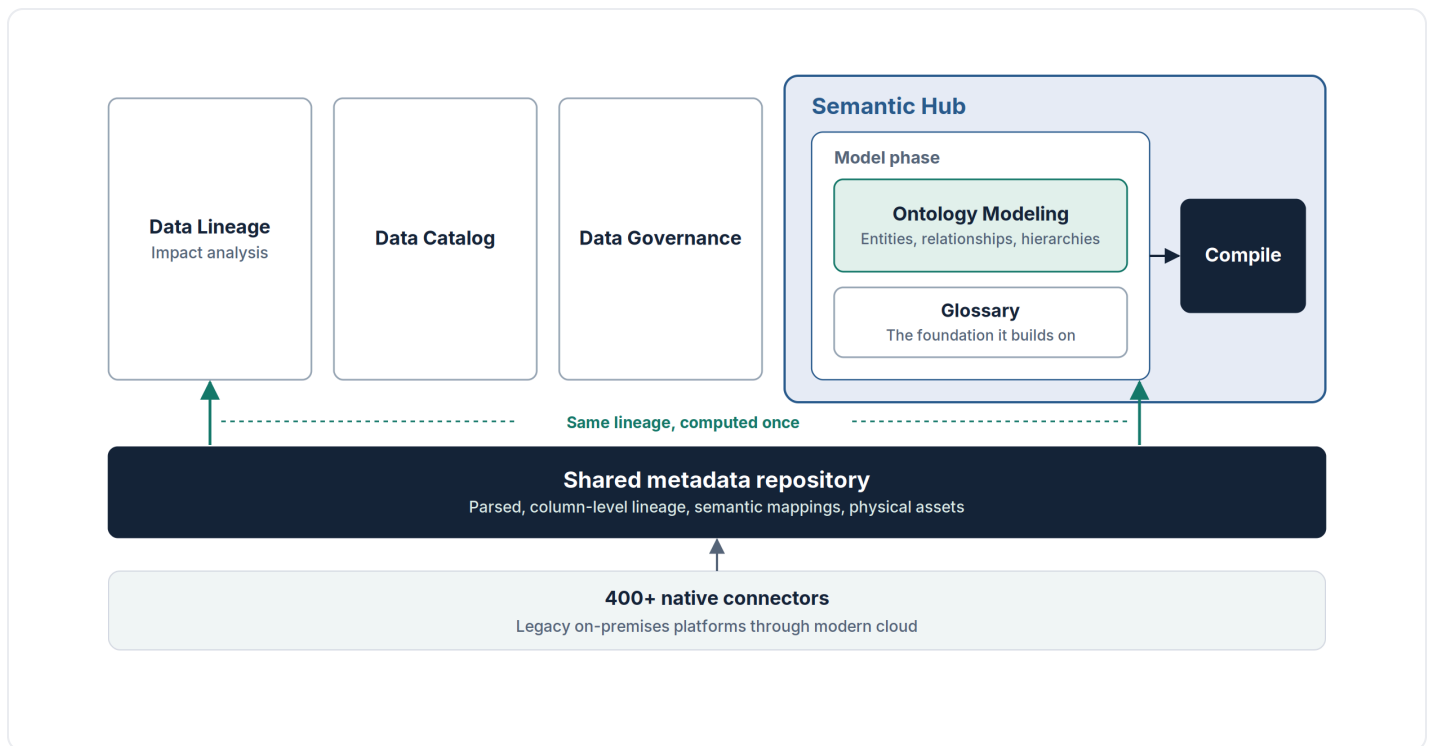


That reconciliation is what makes the ontology auditable. Every concept traces to an asset. Every change to that concept is versioned, attributed, and reversible. When an auditor asks what "eligible member" meant in Q2 and which report used it, the answer is a query.

Where this lives

Ontology Modeling is a feature inside Semantic Hub, built on MetaKarta's Glossary. It sits in Semantic Hub's Model phase, where semantic definitions get standardized, versioned, and assigned stewardship before anything compiles.

It resides in the same shared metadata repository as Data Lineage, Data Catalog, and Data Governance. The lineage that feeds the ontology and the lineage that answers an impact analysis question are the same lineage, computed once.



Bottom-up construction works because that repository already holds a parsed, column-level picture of the estate. The ontology is the business-readable view of something the platform already knows.

Where to start

Take one term your organization argues about. "Eligible member," "active account," "net revenue," whichever one has a meeting attached to it.

Ask three questions about it: 1) which physical columns implement it today, 2) which reports and models depend on those columns, and 3) who approved the current version, and when.

If your ontology can answer all three, it was built from the estate. If it can answer none of them, it was built from documents, and your agents are reasoning against prose.

MetaKarta v12 is available now. Request a demo to learn how to build ontology models from your existing data lineage.

[Request Demo](#)

About Meta Integration Technology

Meta Integration Technology, Inc. (MITI) is a pioneer in metadata integration, semantic modeling, and data lineage with decades of experience serving global enterprises and technology providers. MITI's technology is embedded in products from leading data and analytics vendors and powers MetaKarta, its enterprise platform for metadata-driven data management. MetaKarta delivers data lineage, data cataloging, data governance, and semantic enforcement through a single integrated system, connecting the full enterprise data estate across 400+ native connectors, from legacy to modern and on-premises to the cloud.

650 Castro Street, Suite 120-431, Mountain View, California 94041-2055 USA
Phone: 650.429.8853

info@metakarta.com
www.metakarta.com