

Understand Microsoft Fabric IQ fundamentals

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/1-introduction>

Introduction

Completed

Imagine you're a data analyst at Lamna Healthcare, responsible for helping clinical operations teams understand patient care patterns across your facility.

Patient records sit in lakehouse tables while vital signs stream continuously from ICU monitoring equipment into an eventhouse. When hospital administrators ask questions like "Which patients in the ICU have elevated vital signs?" or "How many beds are occupied on the surgical floor?", you need to manually join lakehouse tables with eventhouse streams, translate business terms into technical column names, and write complex queries.

Business users can't explore the data themselves—they depend on you to write queries each time they have a question. By the time you deliver answers, clinical conditions may have already changed.

Fabric IQ solves this challenge by letting you define business vocabulary in an ontology, then bind those concepts to your data sources in OneLake. You define concepts such as Patient, Department, and Room with their properties and relationships, creating a semantic layer. Business users can then ask questions in natural language through data agents or visually explore relationships through Graph in Microsoft Fabric—exploring the data themselves without needing you to write queries.

In this module, you'll discover what Fabric IQ is and how it works. You'll explore the components that work together—ontology items, data agents, Graph in Microsoft Fabric, and Power BI semantic models—and learn when to use each one. You'll also see how ontology modeling shifts your approach from use-case-driven thinking to concept-driven thinking, fundamentally changing how teams collaborate around data.

Important

Fabric IQ is currently in [preview](#).

2. Get started with Fabric IQ

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/2-get-started-with-fabric-iq>

Get started with Fabric IQ

Completed

Fabric IQ is a workload in Microsoft Fabric for creating ontologies that define your business vocabulary. It sits alongside other Fabric workloads like Data Engineering, Data Factory, Data Science, Data Warehouse, Real-Time Intelligence, and Power BI. Within the IQ workload, you create **ontology items**—Fabric artifacts that contain your ontology definitions and data bindings.

An ontology is a shared vocabulary of your business. It's made up of the things in your environment (represented as entity types), their facts (represented as properties of entity types), and the ways they connect (represented as relationships).

You can also think of an ontology like a business context layer, containing:

- A catalog of concepts (like Hospital, Patient, Department) with their properties and relationships
- Data bindings to your lakehouse tables and eventhouse streams
- A graphical representation that links related concepts for navigation and analysis
- A query surface that lets you ask questions about concepts (not just tables), supporting federated queries across sources

Instead of requiring data experts to translate business questions into SQL queries, you model data using business concepts that everyone understands.

The ontology provides a single definition of each concept that can be used by data agents and Graph in Microsoft Fabric for querying and visualization.

Where Fabric IQ fits in the data platform

Fabric IQ builds on Microsoft Fabric's unified data platform. Here's how it connects to other capabilities:

Ingest and store: Fabric IQ works with data you already have in lakehouse tables and eventhouse streams. It doesn't move or duplicate your data—it creates a semantic layer that references your existing data sources.

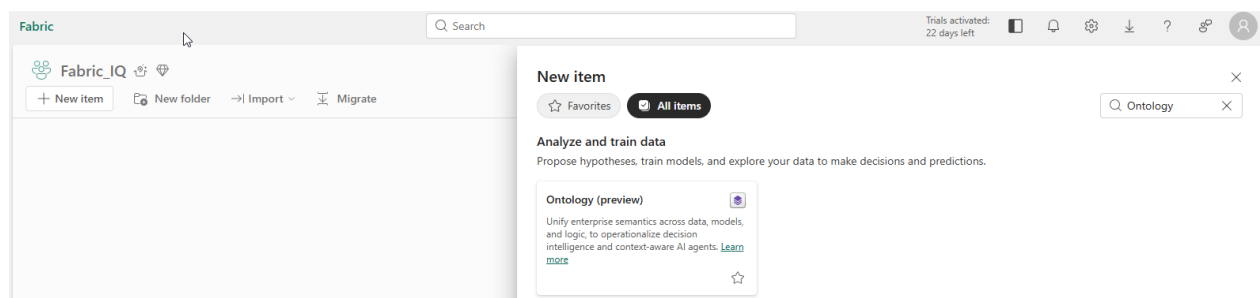
Model and represent semantics: The ontology item offers modeling capabilities by defining entity types, properties, and relationship types. You can generate an ontology structure from existing Power BI semantic models, or create your own from scratch. Then bind these definitions to your data and explore them in a navigable graph that builds automatically.

Analyze and visualize: The ontology item integrates with Graph in Microsoft Fabric to provide a visual graph and query experience based on your business concepts. You can use the ontology to ground data agents with business context.

Access Fabric IQ in your workspace

You create ontology items the same way you create other Fabric items:

1. Navigate to your Fabric workspace
2. Select **+ New item**
3. Search for and select **Ontology (preview)**
4. Enter a name for your ontology (use numbers, letters, and underscores—no spaces or dashes)
5. Select **Create**



The ontology opens when it's ready. You'll see two main areas: the configuration canvas where you define entity types and relationships, and the preview experience where you explore your data.

Important

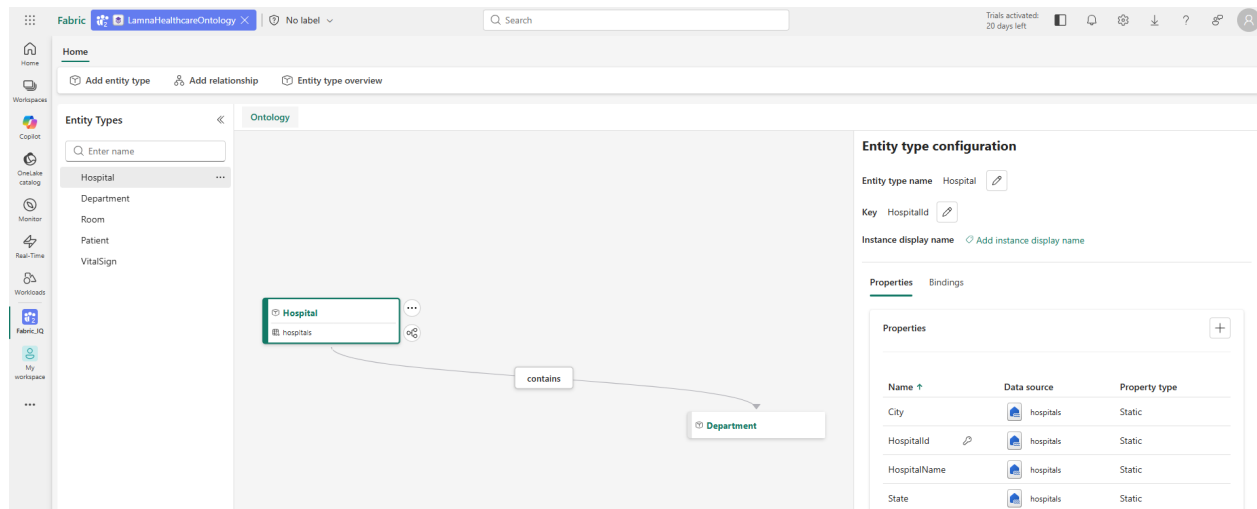
Your Fabric administrator needs to enable certain tenant settings before you can create ontology items. For more information, see [Ontology \(preview\) required tenant settings](#).

Explore the ontology interface

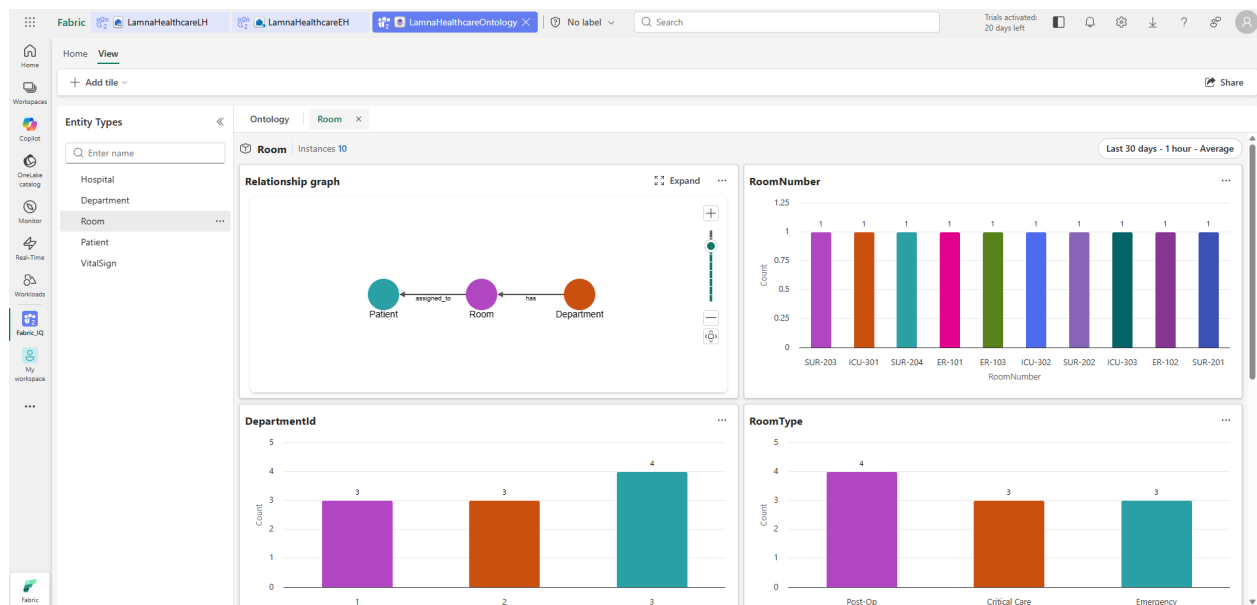
The ontology item has two primary views:

Configuration canvas: This is where you build your ontology. You create entity types (like Hospital, Department, Room, Patient), define properties on those entities (like FirstName, DateOfBirth,

AdmissionDate), and establish relationship types between entities (like "contains" or "assigned to").



Preview experience: This view shows your instantiated ontology. You can see entity instances (specific rooms, departments, and patients), explore relationships in a graph visualization, and query your data using business language instead of SQL. The preview experience integrates with Graph in Microsoft Fabric to provide rich visual exploration.



Understand the build-bind-query workflow

Creating an ontology in Fabric IQ follows three main steps:

Build: Define your business vocabulary by creating entity types, properties, and relationship types. You're modeling the concepts that matter to your business. For example, in a healthcare scenario, this means defining what a Patient is, what properties describe a patient (like name, date of birth, admission date), and how patients relate to other entities (like rooms and departments).

Bind: Connect your ontology definitions to data sources. This includes lakehouse tables for static data (like patient records and room assignments) and eventhouse streams for time-series data (like vital signs monitoring). Data binding maps your source data columns to ontology properties.

Query: Once your ontology is bound to data, you can query it using business concepts instead of database tables. Use Graph in Microsoft Fabric to visualize relationships and traverse connections. Use Query Builder to filter and explore entity instances without writing SQL. Or connect AI agents that can answer natural language questions using your business vocabulary.

This workflow separates business meaning from physical data structures.

How Fabric IQ connects to OneLake

Fabric IQ doesn't move or duplicate your data. Instead, it creates a semantic layer that references existing data sources:

- **Lakehouse tables** contain static data (patient records, hospital information, room assignments)
- **Eventhouse streams** contain time-series data (continuous vital signs from medical monitoring equipment)

When you query your ontology, Fabric IQ automatically sends your queries to the most efficient system to get results quickly. For graph traversals, it uses GQL for Graph in Microsoft Fabric. For time-series queries, it uses KQL for Eventhouse. This federated query capability means you can ask business-level questions that span multiple data sources without knowing the technical details of where data lives.

Two paths to create an ontology

Fabric IQ offers two approaches for creating ontologies:

Generate from Power BI semantic model: If you already have a well-structured Power BI semantic model, you can automatically generate an initial ontology structure from it. Fabric IQ creates entity types matching your tables, properties matching your columns, and relationship types following your model relationships. You then refine this generated ontology by renaming entity types, verifying keys and bindings, and enhancing it with additional data sources like time-series eventhouse streams.

Build from OneLake data: If you don't have a semantic model, or want full control over ontology design, you can build directly from lakehouse and eventhouse data. You manually create entity types, define properties, and establish relationships. This approach gives you complete control over how you model your business vocabulary.

Both paths lead to the same result: a complete ontology that defines your business concepts.

3. Explore Microsoft Fabric IQ components

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/3-explore-fabric-iq-components>

Explore Microsoft Fabric IQ components

Completed

Microsoft Fabric IQ brings together several components that work as an integrated ecosystem. Each component serves a specific role in how you define, query, analyze, and visualize your business data. Understanding these components helps you choose the right tool for each task and leverage their combined strengths.

Fabric IQ includes four core components:

- **Ontology items** define your business vocabulary and bind concepts to data sources
- **Data agents** answer natural language questions across multiple data sources
- **Graph** stores and queries connected data using relationships
- **Semantic models** provide a starting point for generating ontologies from existing Power BI models

Ontology items: Define your business vocabulary

An **ontology item** is where you build your shared business vocabulary. You define the business concepts that matter to your organization, then bind them to actual data sources in OneLake. If you were working with healthcare data, for example, you might bind a Patient concept to a lakehouse table containing patient records, and a VitalSign concept to an eventhouse stream capturing real-time monitoring data.

Consider a scenario where a healthcare data team defines Hospital, Department, Room, Patient, and VitalSign as entity types with properties like hospital ID, department name, and room number. Relationships like "Department contains Room" and "Patient occupies Room" connect these concepts into a coherent business vocabulary.

The ontology defines your business concepts that can be used by Graph in Microsoft Fabric for visualization and traversal, and by data agents for answering natural language questions.

Data agents: Query data with natural language

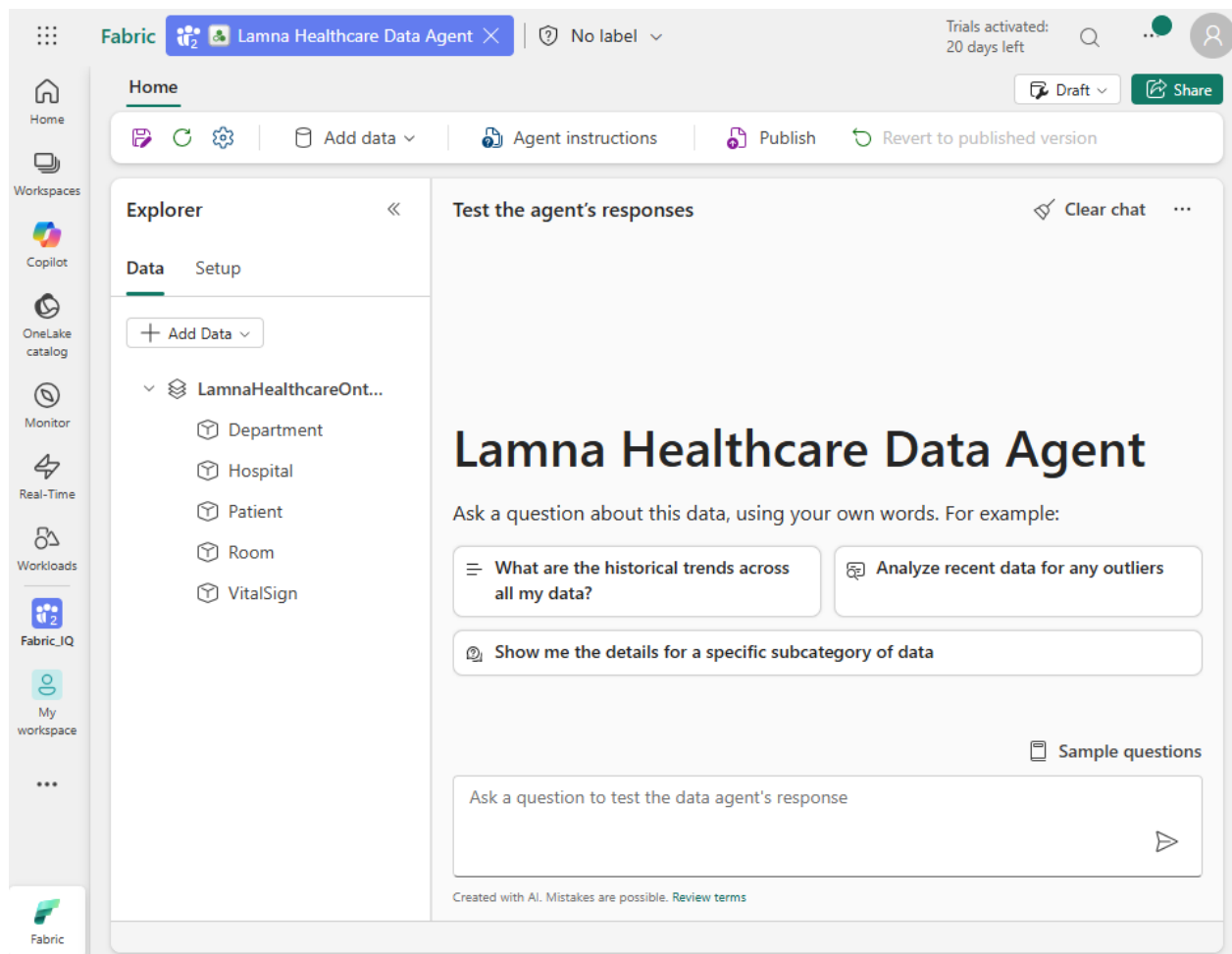
A **Fabric data agent** is a conversational Q&A system powered by generative AI. You configure it with up to five data sources in any combination (lakehouses, warehouses, KQL databases, Power BI semantic models, or ontologies), and it uses Azure OpenAI Assistant APIs to process natural language questions.

When you ask a question, the data agent parses it, identifies the most relevant data source, and generates the appropriate query. For lakehouses and warehouses, it generates SQL. For Power BI semantic models, it generates DAX. For KQL databases, it generates KQL. For ontologies, it uses the defined business vocabulary to understand context.

Consider a scenario where a nurse asks "Which patients in cardiology have elevated heart rates right now?" The data agent understands "cardiology" from the ontology, queries the lakehouse for patient assignments, and queries the eventhouse for current vital sign readings—all without the nurse writing SQL or KQL.

You enhance data agent accuracy by providing **data agent instructions** (guidance on which data source to use for specific question types) and **example queries** (sample question-query pairs that illustrate expected responses). This customization aligns the data agent with your organization's specific terminology and data access patterns.

Data agents enforce read-only access and apply security protocols to ensure users only see data they have permission to access. You can publish data agents to Microsoft 365 Copilot or integrate them with Microsoft Copilot Studio to extend their reach beyond Fabric.



Graph in Microsoft Fabric: Visualize and traverse relationships

Graph in Microsoft Fabric offers native graph storage and compute for connected data. Unlike relational databases that require complex joins to navigate relationships, it uses a **labeled property graph model** where nodes (entities) and edges (relationships) carry labels and properties that make connections explicit and easy to traverse.

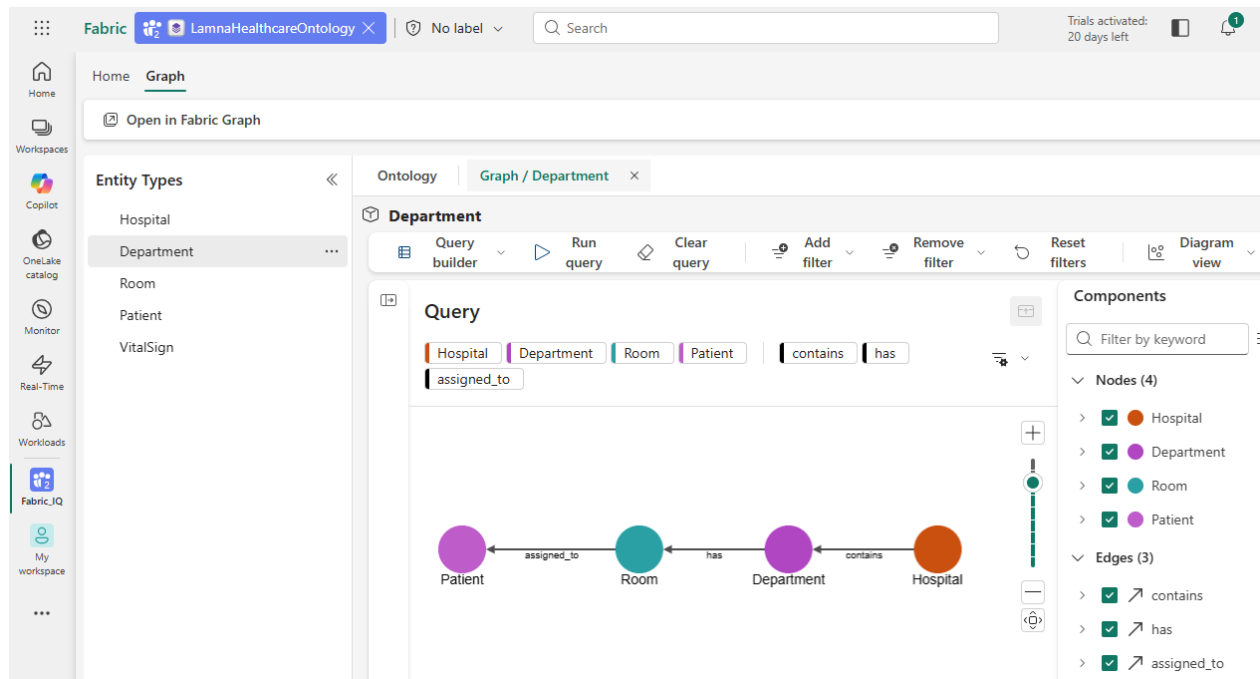
When you create an ontology item, a managed graph is automatically created from your ontology's entity types and relationships. You query it using **GQL (Graph Query Language)**, an international standard for graph queries. Graph excels at answering relationship-heavy questions like "Which patients are assigned to surgical floor rooms?" or "Show me the department hierarchy for a specific hospital."

Consider a scenario where the ontology automatically creates a graph with nodes for hospitals, departments, rooms, and patients, connected by edges representing their relationships. To investigate patient flow patterns, you can traverse from a hospital → its departments → rooms → currently assigned patients.

Graph in Microsoft Fabric operates directly on OneLake without requiring data duplication or ETL. Its scale-out architecture can handle large-scale graph structures with many relationships. You can

explore graph data visually through the graph interface or write GQL queries for more complex analysis.

Ontology and Graph in Microsoft Fabric work together seamlessly. The ontology declares your business concepts and relationships, then automatically creates a graph structure. Graph in Microsoft Fabric stores and computes the traversals, enabling visual exploration and advanced queries over your connected data.



Semantic models: Generate ontologies from existing data models

A **Power BI semantic model** provides a structured representation of your data with tables, columns, relationships, and business logic already defined. In the context of Fabric IQ, semantic models serve as an excellent starting point for creating ontologies.

When you generate an ontology from an existing semantic model, Fabric IQ automatically creates:

- **Entity types** matching your semantic model tables (like Hospital, Department, Room, Patient)
- **Properties** matching your table columns (like HospitalId, DepartmentName, Floor)
- **Relationship types** based on your semantic model relationships (like Hospital contains Department, Room assigned to Patient)
- **Keys** identifying unique instances of each entity type

This approach is significantly faster than building an ontology from scratch. Instead of manually creating each entity type and defining every property and relationship, you start with a complete structure that reflects your existing data model.

Consider a scenario where a data team generates an ontology from their existing semantic model with tables for hospitals, departments, rooms, and patients. This automatically creates the entity

types and relationships, which they could then enhance by adding time-series vital sign data from eventhouse.

After generating an ontology, you refine it by:

- Verifying that entity type keys are correctly identified
- Confirming that data bindings map to the right source columns
- Adding additional entity types from other data sources (like eventhouse streams)
- Enhancing relationships with binding information that links to actual data

This workflow lets you leverage existing modeling work while extending it with Fabric IQ's cross-domain reasoning and graph capabilities.

4. Understand the ontology modeling paradigm

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/4-understand-ontology-modeling-paradigm>

Understand the ontology modeling paradigm

Completed

Ontology modeling in Fabric IQ defines business concepts independent of specific analytical use cases. This unit explains how the ontology approach differs from traditional analytical data modeling.

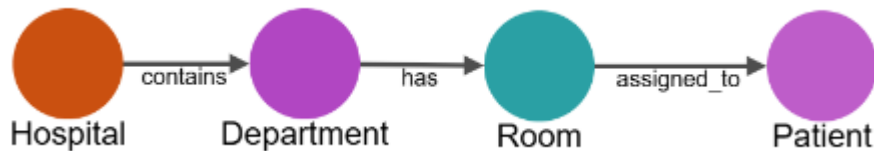
Recognize business concepts over table schemas

Traditional analytical data modeling designs tables optimized for specific reporting and analytics needs. You start with questions like "What reports do we need to support? Which dimensions and facts are required?" and design star schemas or dimensional models accordingly.

Ontology modeling inverts this. You start by asking: "What are the core concepts in our business? How do they relate? What facts matter about each concept?" Analytical considerations come after you've captured business meaning.

Consider a scenario where you're working with healthcare data. A traditional approach might create PatientDim, RoomDim, and DepartmentDim tables in a star schema with abbreviated column names like "pt_id," "rm_num," and "dept_id." When different teams build separate data marts independently, they may develop overlapping but inconsistent definitions of "patient" or "department."

With ontology modeling, you define Patient, Room, and Department as entity types using business language: PatientName, DateOfBirth, RoomNumber, DepartmentName. You establish named relationships: Department has Room, Room assigned to Patient. These definitions become the foundation for AI agents and graph queries that need consistent business context.



This ontology model shows Hospital, Department, Room, and Patient entities connected by named relationships ("contains", "has", "assigned_to") in Fabric IQ's graph interface. Unlike foreign keys that require JOIN statements, these relationships have explicit names and can be queried directly.

Model entity types as reusable concepts

Entity types are conceptual definitions that you create before binding them to data, unlike database tables that combine schema definition with data storage.

An entity type standardizes the name, description, identifiers, and properties for a business concept. By defining what "patient" means at the entity type level—including which properties exist and what the identifier is—you create a definition that can then be bound to a data source. This separates the conceptual model from the underlying table structure.

Define properties with semantic meaning

Properties provide a way to standardize names and data types at the conceptual level, which you then map to actual columns during data binding.

Traditional database columns often vary: "temp," "temperature," "temp_reading," or "body_temp" for the same concept across different tables. Each table defines its own column names independently.

In ontology modeling, you define a standard property name like "Temperature" at the entity type level. When you bind to data, you map this property to whatever column name exists in your table - whether it's "temp," "temperature," or "body_temp." Tools querying the ontology always see the standardized "Temperature" property, regardless of the underlying column name.

Entity Types

- Hospital
- Department
- Room
- Patient
- VitalSign** ...

Ontology

Entity type configuration

Entity type name

VitalSign

Key

ReadingId

Instance display name

[Add instance display name](#)

Properties

Bindings

Properties

Name ↑	Data source	Property type
HeartRate	 VitalSignsReadings	Timeseries
OxygenSaturation	 VitalSignsReadings	Timeseries
PatientId	 VitalSignsReadings	Static
ReadingId 	 VitalSignsReadings	Static
RespiratoryRate	 VitalSignsReadings	Timeseries
RoomId	 VitalSignsReadings	Static

This VitalSign entity type shows standardized property names like HeartRate, OxygenSaturation, and RespiratoryRate. When bound to data, these properties map to the actual column names in the source table.

Properties can be marked as identifiers to signal which values uniquely identify entity instances. Marking PatientID as an identifier establishes that this value uniquely represents each patient, ensuring consistent entity resolution across data sources.

Establish relationships between concepts

Relationships in ontology modeling are named, directional connections between entity types. Unlike foreign keys that are implicit until you write JOIN statements, these relationships are explicit concepts that tools can query and visualize.

Consider a scenario where a "Department has Room" relationship links hospital departments to their physical spaces. A "Room assigned to Patient" relationship tracks current patient assignments. You

can traverse these relationships in the graph interface or query them using GQL to perform dependency analysis—when you need to find all patients in a specific department, you can traverse "Department has Room" followed by "Room assigned to Patient."

Bind concepts to data without duplication

Data binding connects your entity types to actual data sources without copying or moving data. The data stays in place—in lakehouse tables or eventhouse streams in OneLake. The ontology creates a semantic layer that references this data.

Consider a scenario where a Patient entity type binds to a lakehouse table containing demographic information, while the VitalSign entity type binds to an eventhouse stream providing real-time vital signs. Each entity type binds to its appropriate data source using entity-specific keys. When you query across these concepts, the ontology federates queries across the lakehouse and eventhouse, returning integrated results without having moved any data.

The screenshot shows the 'Configure data binding' window for the 'LamnaHealthcareEH' data source. The interface is divided into two main sections: 'Binding type' and 'Bind your properties'.

Binding type: The 'Timeseries' option is selected. Below it, the 'Source data timestamp column' is set to 'Timestamp'.

Bind your properties: This section is divided into two tabs: 'Static' and 'Timeseries'.

- Static tab:** It shows three properties being mapped: 'ReadingId' (source column: 'ReadingId'), 'PatientId' (source column: 'PatientId'), and 'RoomId' (source column: 'RoomId').
- Timeseries tab:** It shows three properties being mapped: 'HeartRate' (source column: 'HeartRate'), 'OxygenSaturation' (source column: 'OxygenSaturation'), and 'RespiratoryRate' (source column: 'RespiratoryRate').

Each property mapping includes a source column dropdown, a property name input, and a delete icon. There are also buttons to 'Add static property' and 'Add timeseries property'.

This data binding configuration shows timeseries properties from an eventhouse mapped to entity type properties. Source columns (HeartRate, OxygenSaturation, RespiratoryRate) bind to standardized property names without copying data—the ontology references the eventhouse data directly.

Data binding creates a semantic layer over your existing data sources without duplication, enabling AI agents to query using business terminology instead of table-specific column names.

5. Module assessment

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/5-knowledge-check>

Module assessment

Completed

6. Summary

<https://learn.microsoft.com/en-us/training/modules/understand-fabric-iq-fundamentals/6-summary>

Summary

Completed

Microsoft Fabric IQ lets you define business vocabulary once in ontologies, enabling natural language queries and graph visualization of your data. In this module, you learned what Fabric IQ is, how to access it, and how it fits within Microsoft Fabric's data platform.

You explored the build-bind-query workflow for creating ontologies: defining entity types and relationships, binding them to lakehouse tables and eventhouse streams, and querying them through Graph or data agents. You also saw how to generate ontologies from existing Power BI semantic models or build them from scratch using OneLake data.

You examined four components that work together: ontology items define your business vocabulary, data agents answer natural language questions across multiple data sources, Graph in Microsoft Fabric visualizes and analyzes relationships, and Power BI semantic models can generate initial ontology structures.

Finally, you learned how ontology modeling differs from traditional analytical modeling. Instead of starting with specific reporting needs and designing tables optimized for queries, ontology modeling starts with core business concepts and their relationships. This concept-driven approach creates reusable definitions that both data agents and Graph can use, enabling business users to explore data using familiar terminology.

