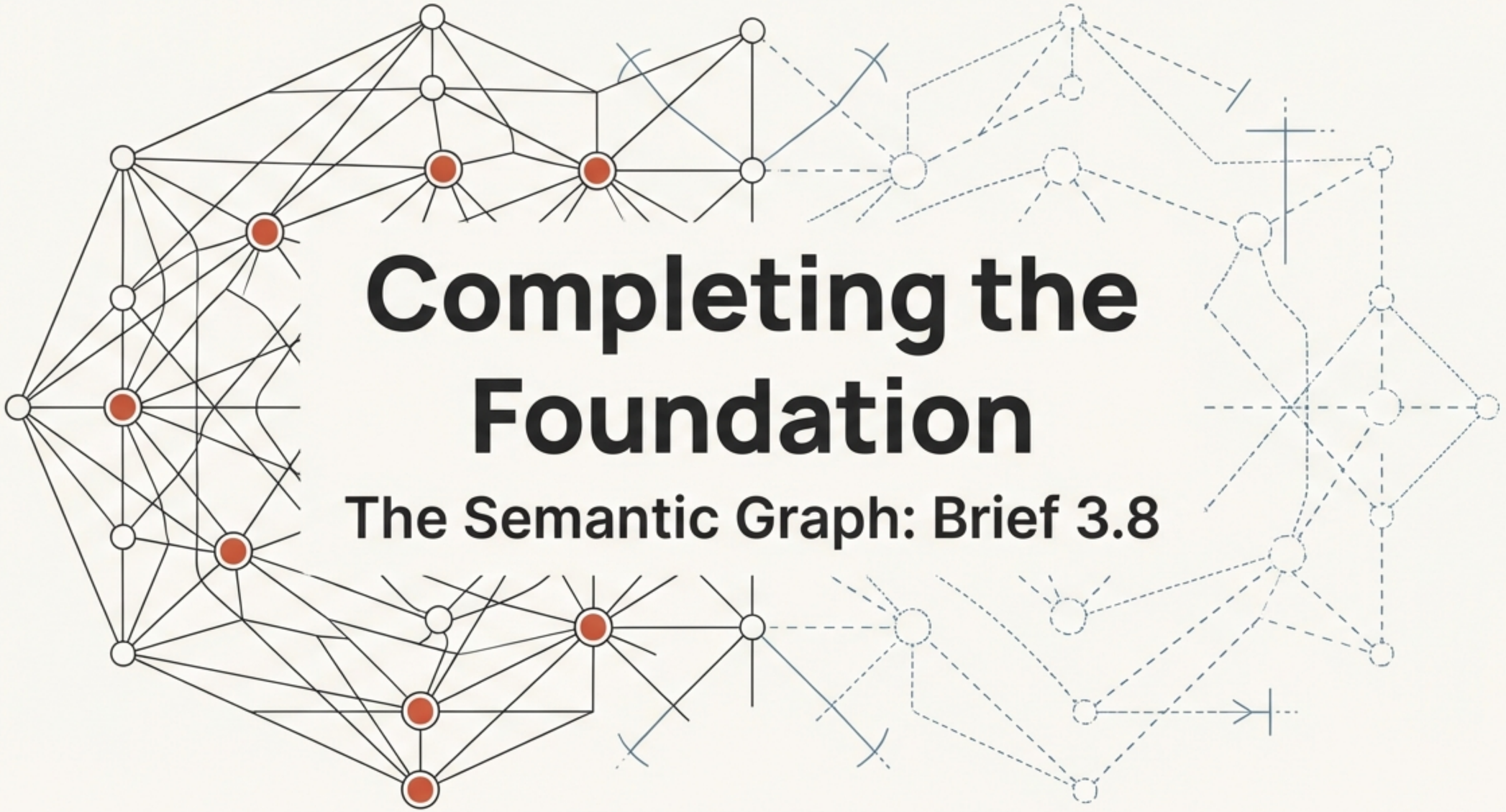


Completing the Foundation

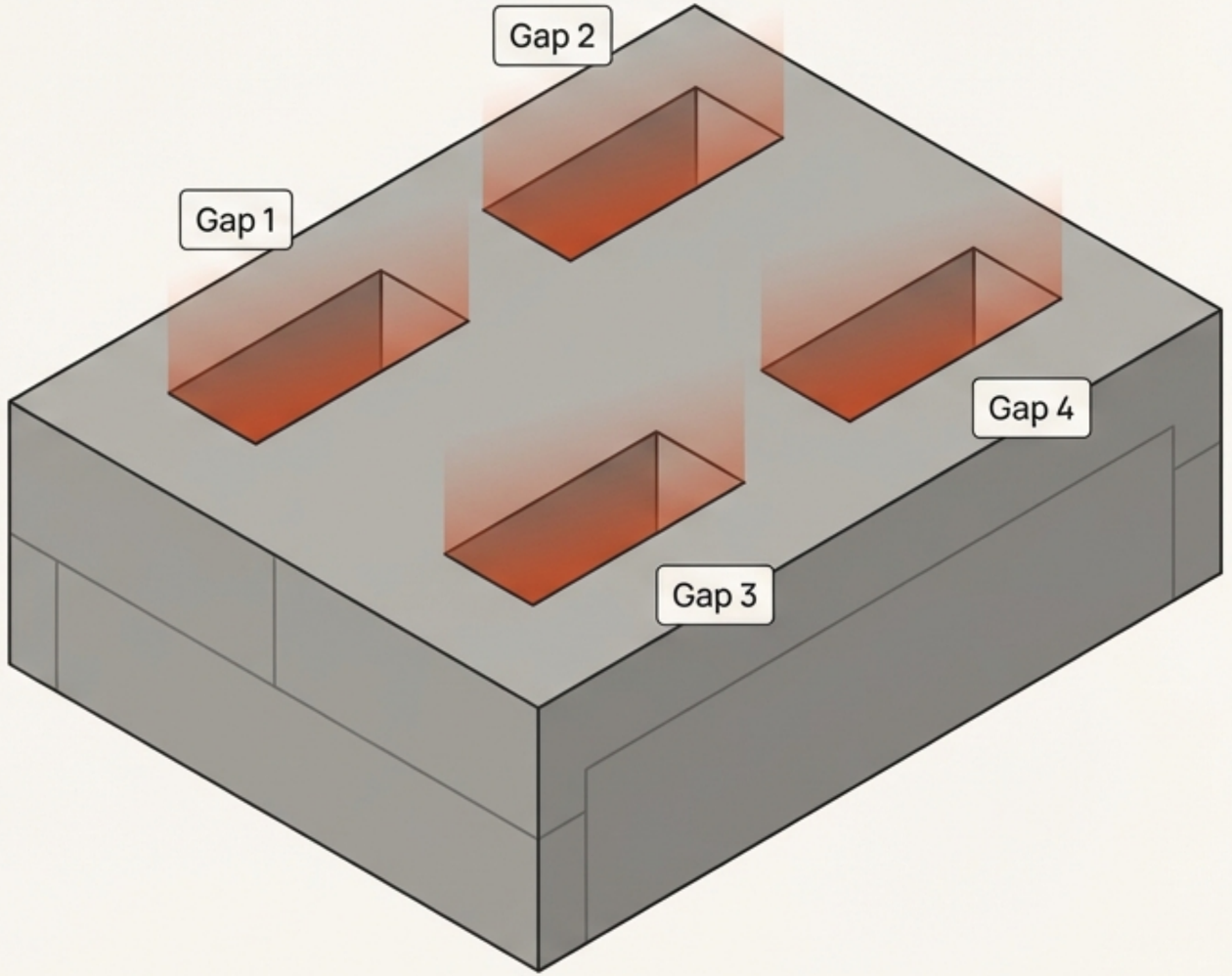
The Semantic Graph: Brief 3.8

OSBot-Utils v3.65.0 | January 2026

We built a powerful foundation with Brief 3.7.

Brief 3.7 successfully delivered our core Projected Data Architecture. It established the fundamental concepts of Nodes, Edges, and Projections. However, implementation revealed four critical gaps preventing.

However, implementation revealed four critical gaps preventing it from being a complete, standards-aligned system.



The four foundational gaps we must address.

1. Disconnected Taxonomy

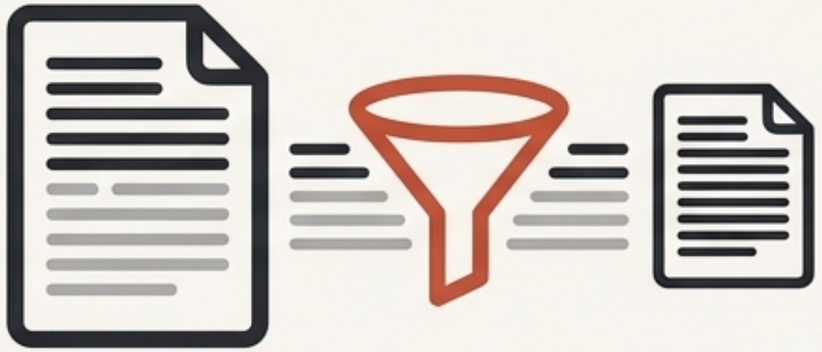
Node types have no formal link to their categories. The ontology and taxonomy exist in isolation.

2. Inconsistent References

The taxonomy uses `_ref` for internal links, which is inconsistent with the rest of the system's `_id`-based approach for referential integrity.

3. Missing Data

Nodes and edges are purely structural; they cannot store data values (properties) like line numbers or timestamps.



4. Bloated Projections

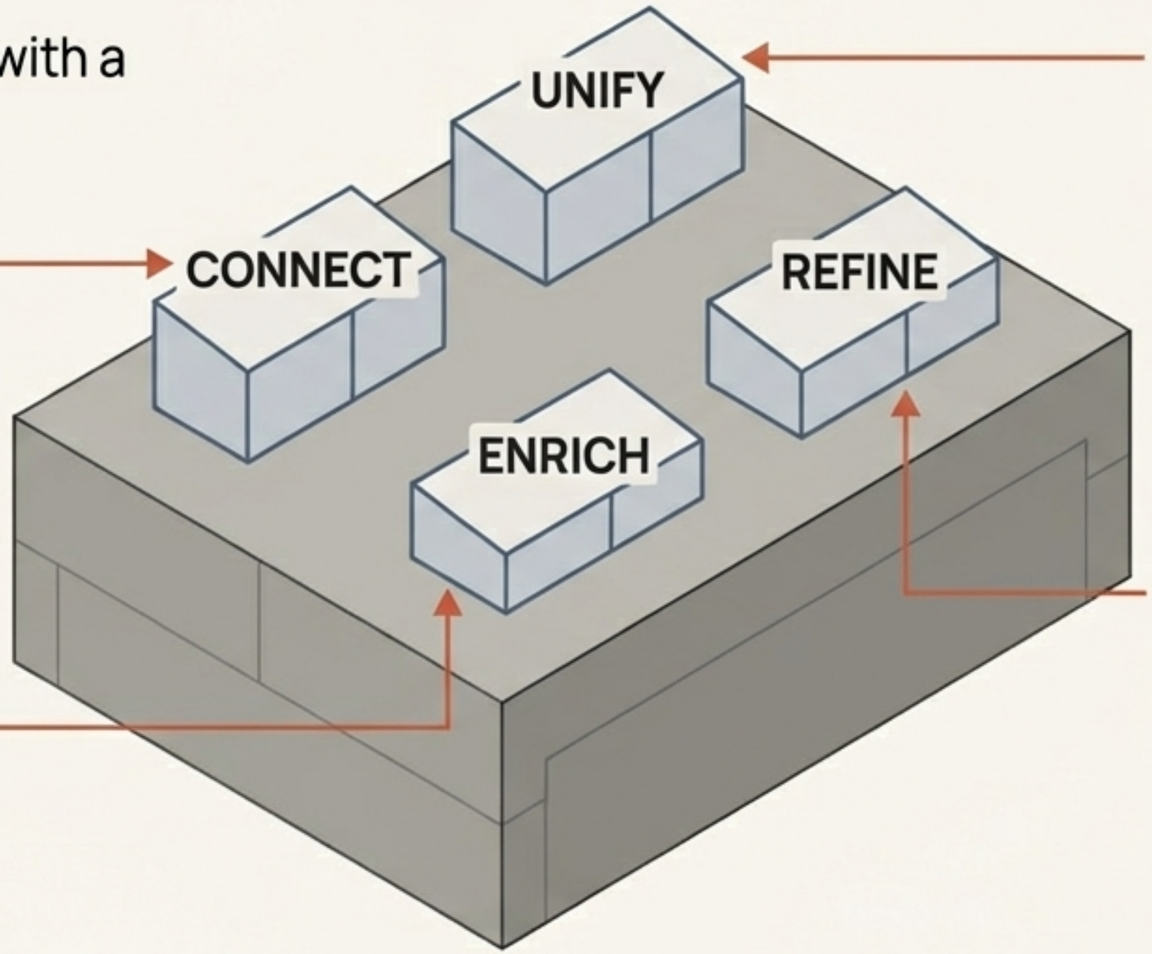
Projections are inefficient, including the entire ontology in the references section, not just the parts actively used in the graph.

Our solution: A cohesive, data-rich architecture.

We will address every gap with a targeted enhancement:

Wire the taxonomy to the ontology via a new `category_id` on Node Types.

Introduce a flexible property system for both Nodes and Edges.



Convert the taxonomy's internal structure to use ID-based references (`parent_id`, `child_ids`).

Filter projection references to include only used entities and add a dedicated `taxonomy` section to the output.

Guided by clear and consistent design principles.

1. IDs for Integrity

All internal cross-references within any schema must use IDs (e.g., `category_id`, not `category_ref`).

2. Refs for Humans

The `_ref` field is a human-readable label, defined once on the source entity, and never used as a foreign key.

3. Flexible Properties

Property values are stored as simple strings (`Safe_Str__Text`), with type validation (`property_type_id`) being an optional layer.

4. Separation of Concerns

The Ontology defines *what CAN* exist. The Rules engine constrains *what MUST* exist.

5. Minimal Projections

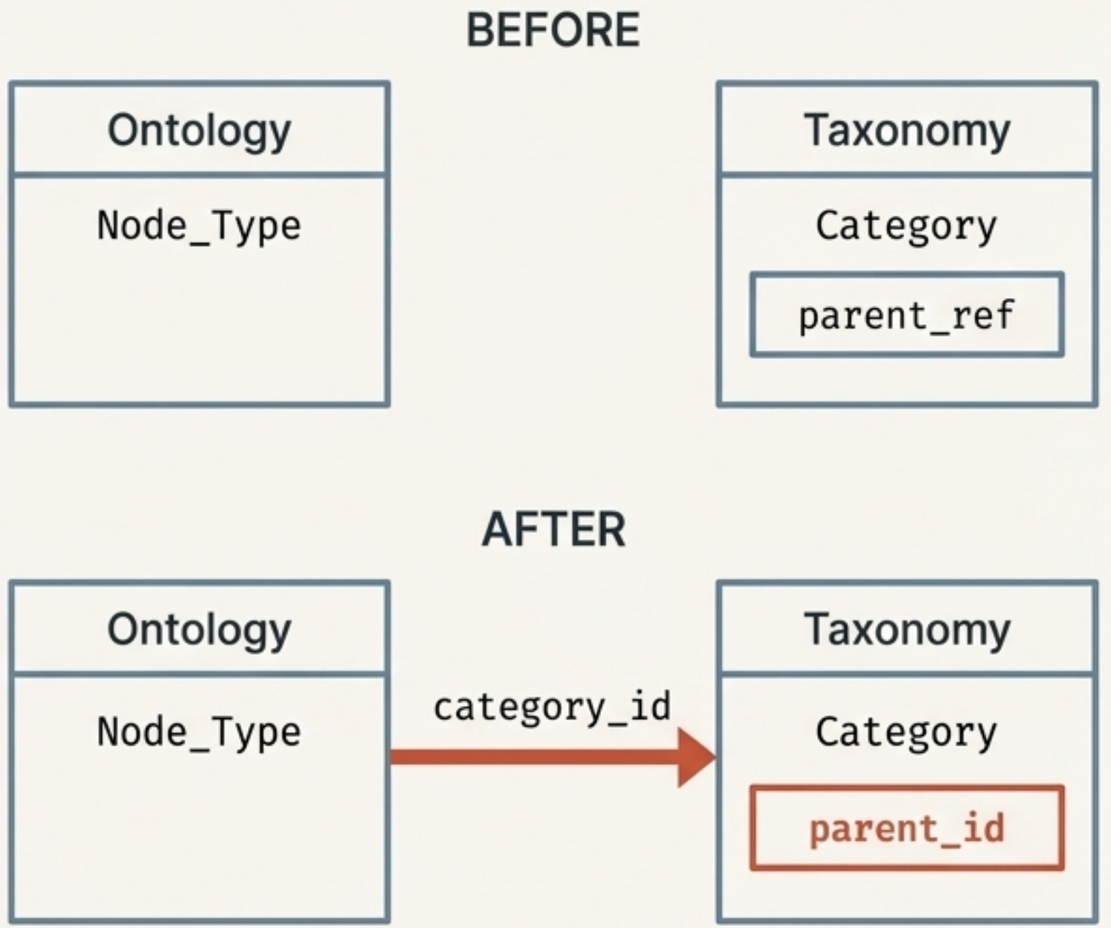
Projections must be efficient, showing only the graph data and references to entities actually used within that graph.

Pillar 1: Integrating the Taxonomy.

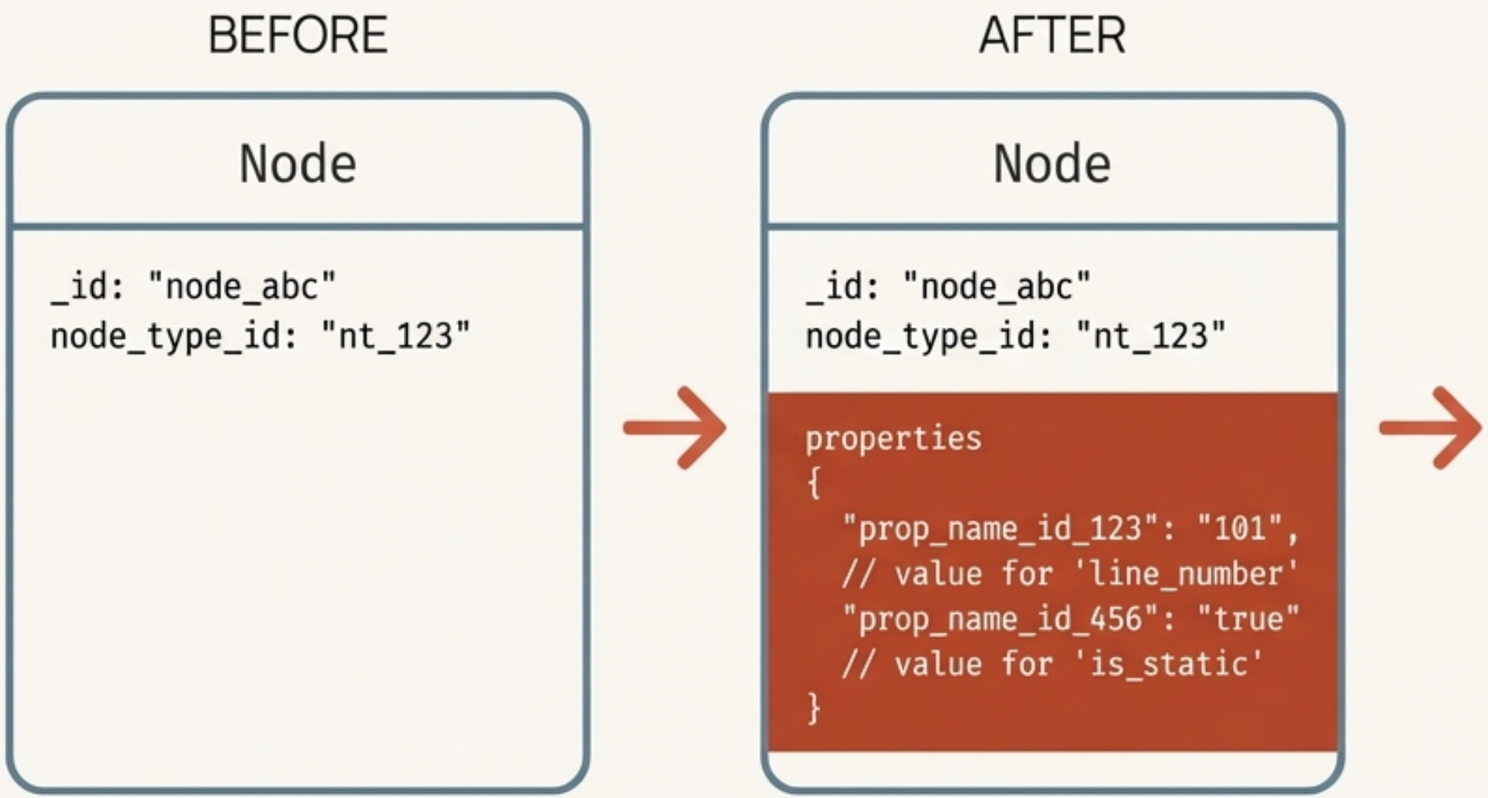
The Change: We are adding a `category_id` field to the `Node_Type` schema.

The Rationale: This formally links our ontology (what things are) to our taxonomy (how they are grouped), directly solving the 'Disconnected Taxonomy' gap.

The Consistency: The taxonomy itself is being updated to use `parent_id` and `child_ids` for its internal hierarchy, aligning with our 'IDs for Integrity' principle.



Pillar 2: Adding substance with data-rich properties.



The Change: Nodes and Edges can now hold a dictionary of key-value properties. This directly solves the 'Missing Data' gap.

The Structure: Each property has a `property_name_id` pointing to its definition in the ontology. It can also have an optional `property_type_id` for validation (e.g., `xsd:int`, `xsd:date`).

The Flexibility: This adheres to our "Flexible Properties" principle. `property_type_id = None` implies a simple string, allowing for both unstructured and strictly validated data as needed.

Our architecture maps directly to Semantic Web standards.

This design intentionally aligns with RDF, RDFS, and OWL concepts, ensuring interoperability and leveraging decades of industry best practice.

Our Concept	Industry Standard
Node Type	`rdfs:Class / owl:Class`
Node	`rdf:Resource / owl:NamedIndividual`
Predicate	`rdf:Property / owl:ObjectProperty`
Edge	RDF Triple (s → p → o)
Property Name	`rdf:Property / owl:DatatypeProperty`
Category	`owl:subClassOf` hierarchy

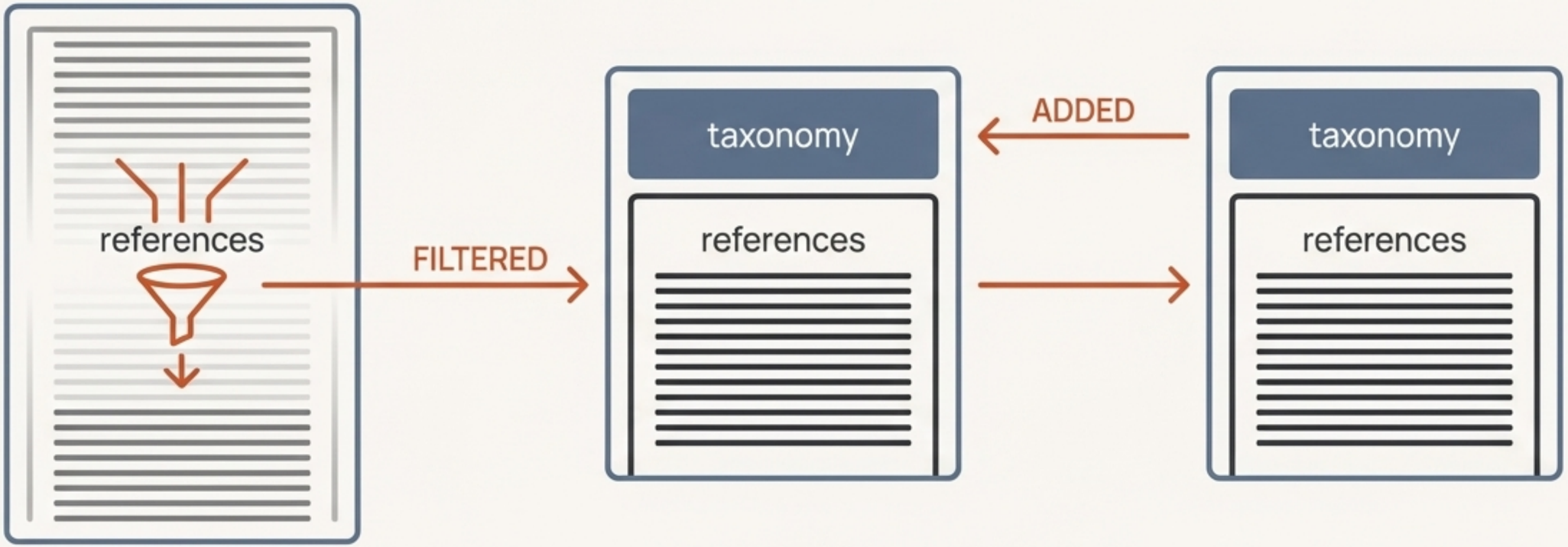
The result: Smarter, leaner, and more useful projections.

Fixed: Leaner References

The `references` section is now filtered to include only the Node Types, Predicates, Categories, and Properties actually used in the graph. This resolves the “Bloated Projections" bug.

Added: Richer Context

A new top-level `taxonomy` section is added to the projection output. This provides a human-readable hierarchy of the categories in use, enabling powerful client-side grouping and filtering without extra lookups.



Bringing it all together: A complete data flow

A ‘Code Method’ node in a software graph.

Before (Brief 3.7)

A “Code Method” node was just a type. It couldn’t store its line number, its category (“Function”) was implicit, and the projection included the entire ontology.

```
{
  "graph": {
    "nodes": [{
      "_id": "cm_1",
      "node_type_ref": "Code Method"
    }]
  },
  "references": {
    "node_types": [ "Code Method", "Code Class", "File"
    "predicates": [ "calls", "imports", "contains", …
  }
}
```

After (Brief 3.8)

The node now has a `category_id` linking to “Function” and a `line_number` property. The projection is lean and includes a `taxonomy` section.

```
{
  "taxonomy": {
    "Code_Construct": { "children": ["Function"] }
  },
  "graph": {
    "nodes": [{
      "_id": "cm_1",
      "node_type_ref": "Code Method",
      "properties": { "line_number": "101" }
    }]
  },
  "references": {
    "node_types": [ "Code Method" ],
    "categories": [ "Function", "Code_Construct" ],
    "property_names": [ "line_number" ]
  }
}
```

A structured, eight-phase implementation plan.

The work is broken down into logical phases, from foundational types to final testing.

Foundation & Taxonomy (Phases 1-2)

Create new Identifier types and refactor the Taxonomy schema to be ID-based.

Ontology & Graph Properties (Phases 3-4)

Implement the Property system in the Ontology and add property support to Graph Nodes and Edges.

Validation (Phase 5)

Extend the Rules engine to support property constraints.

Projection Engine (Phases 6-7)

Update the Projector to handle properties, build the new taxonomy section, and fix the reference filtering.

Testing & Finalisation (Phase 8)

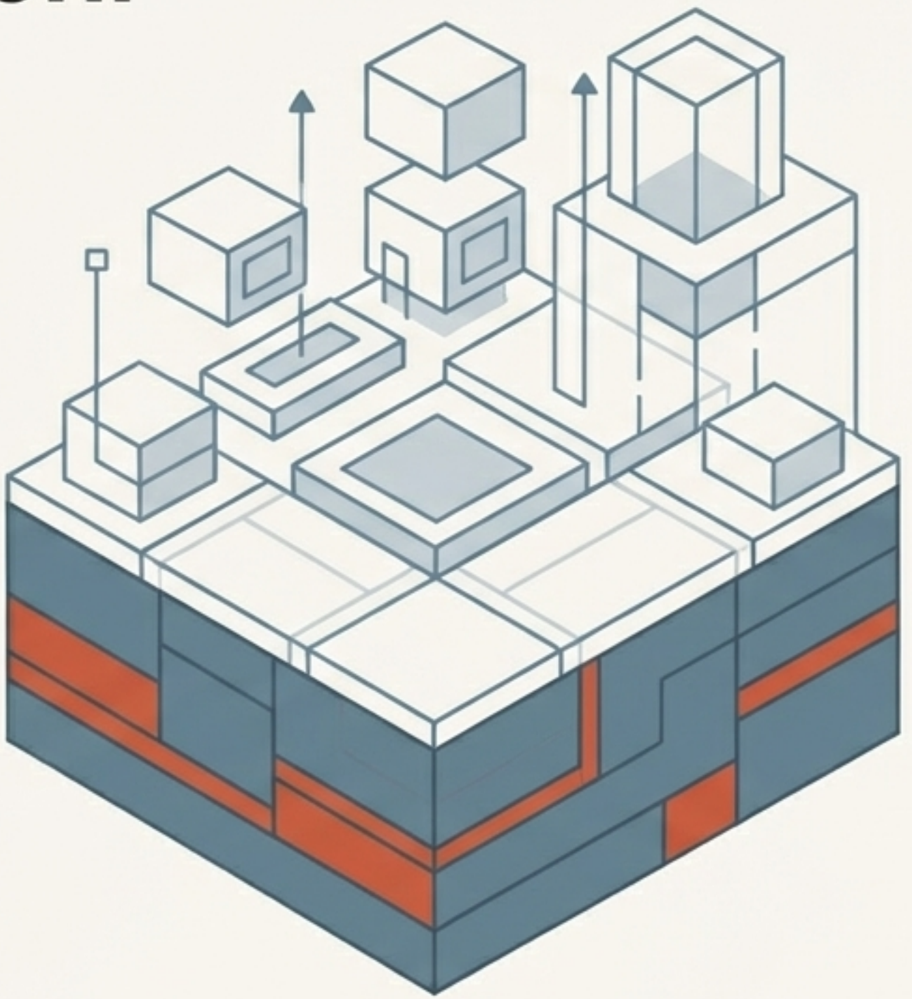
Update all test data and add new tests covering all new features.

We will know we've succeeded when...

- ...Our taxonomy is fully connected and can be used for data queries and classification.
- ...Properties can be added to nodes/edges, validated by the rules engine, and queried end-to-end.
- ...Projection outputs are measurably smaller and faster to process due to filtered references.
- ...Client applications can consume the new `taxonomy` section to build richer, more dynamic UIs for grouping and filtering.

The final result: A complete, standards-aligned foundation.

- **Full RDF/OWL Alignment:** Our architecture is now interoperable and built on industry best practices.
- **Integrated Taxonomy:** We enable powerful, data-driven classification of all nodes.
- **Data-Rich Graphs:** Nodes and edges are no longer just structural; they now carry meaningful data.
- **Principled Design:** We have enforced ID-based integrity and a clear separation between definitions and rules.
- **Optimised Projections:** Our data outputs are human-readable, minimal, and efficient.



Impact on the codebase.

The implementation will touch several core areas of `osbot_utils.helpers.semantic_graphs`.

New Files / Schemas

- Core schemas for `Property_Name` and `Property_Type`
- New collection types (e.g., `Dict__Property_Names__By_Id`)
- New rule types for required properties

Modified Files / Schemas

- Major updates to `Schema__Taxonomy` (ID-based)
- Updates to `Schema__Ontology__Node_Type`
- Updates to `Schema__Semantic_Graph__Node/Edge` (properties)
- Major refactoring of `Projector` and `Schema__Projected__Semantic_Graph`

**(A complete file list and detailed checklist is provided in Brief 3.8 for implementation.)*

Discussion & Next Steps

Brief 3.8: Completing the Semantic Graph Foundation

Feedback on the phased rollout plan?

Potential impact on downstream client applications?

Alignment with other ongoing architectural initiatives?