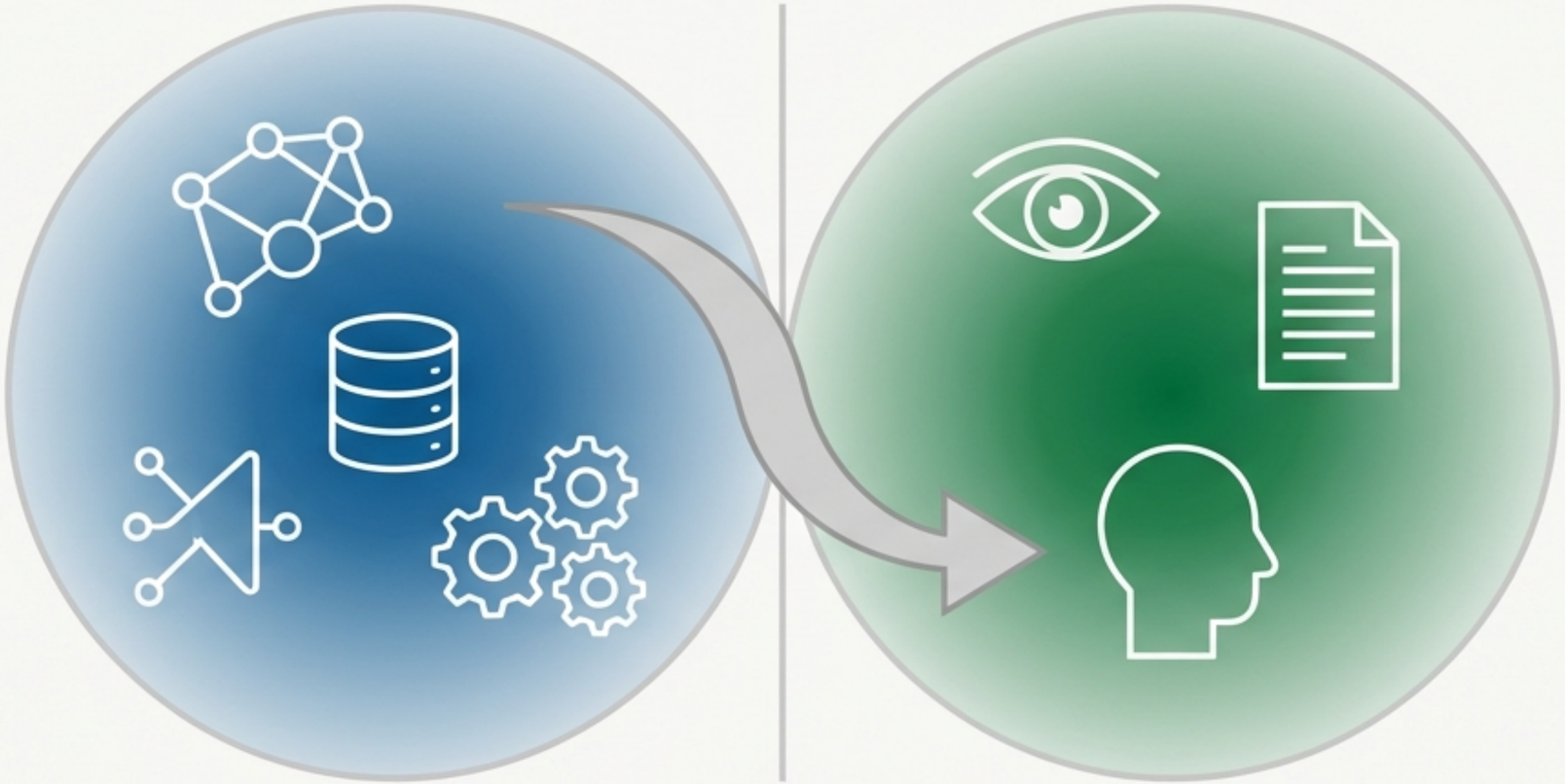
Our Data Has Two Audiences. It’s Time Our Architecture Did Too.



Introducing the Projected Data Architecture

We Face a Fundamental Conflict: Machine Integrity vs. Human Readability.

For Machines: We Need Referential Integrity.

For robust graph operations, data storage, and reliability, we must use ID-based references. They are **unambiguous**, **efficient**, and machine-friendly.

```
{
  "ontology_id": "7b4e9f12...",
  "node_type_id": "9e8d7c6b...",
  "predicate_id": "3c1a0f5e..."
}
```

For Humans: We Need to Understand Our Data.

For debugging, visualisation, and quick comprehension, we need **human-readable names and relationships.** IDs are opaque and meaningless to a person.

```
{
  "ontology": "python_code",
  "node_type": "class",
  "edge_type": "calls"
}
```

Relying on IDs Alone Makes Our Data Opaque and Hard to Debug.

What ontology is `7b4e9f12`?

What node type is `9e8d7c6b`?

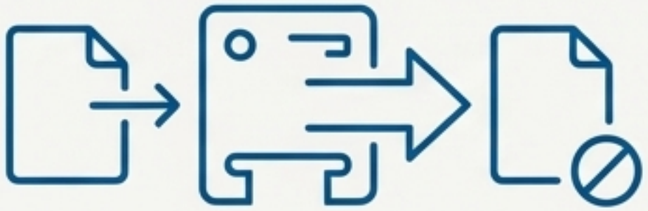
```
{
  "graph_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
  "ontology_id": "7b4e9f12-c3d4-e5f6-7890-1234567890ab",
  "nodes": [
    {
      "node_id": "f0e9d8c7-b6a5-4321-fedc-ba9876543210",
      "node_type_id": "9e8d7c6b-a5b4-c3d2-e1f0-9876543210fe"
    },
    {
      "node_id": "12345678-90ab-cdef-0123-456789abcdef",
      "node_type_id": "9e8d7c6b-a5b4-c3d2-e1f0-9876543210fe"
    }
  ],
  "edges": [
    {
      "from_node": "f0e9d8c7-b6a5-4321-fedc-ba9876543210",
      "to_node": "12345678-90ab-cdef-0123-456789abcdef",
      "predicate_id": "3c1a0f5e-d4c3-b2a1-0f9e-8d7c6b5a4321"
    }
  ]
}
```

What does this edge represent?

How can I possibly visualise this?

This forces us to constantly cross-reference and hold complex mappings in our heads, slowing down development and making debugging a chore.

To Resolve This Conflict, We Established a New Philosophy.

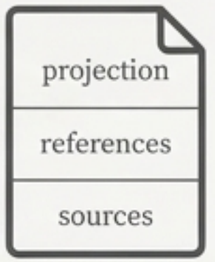


Principle: Projections are Generated, Not Edited.

A projection is a disposable report generated from the master data. You read it; you don't edit it. This eliminates sync complexity and preserves a single source of truth.

Principle: Human Readability Is the Primary Goal.

The projection exists to be understood by people. Every decision, from naming to structure, prioritises clarity.
No machine-centric IDs are allowed in in the human-facing section.



Principle: Separation of Concerns via Three Sections.

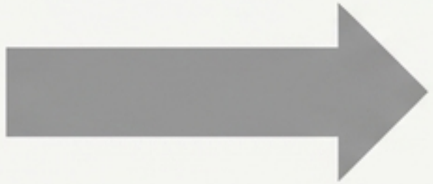
A projected file cleanly separates what humans read (`projection`), what tools use (`references`), and what debuggers need (`sources`). Each audience gets exactly what it needs.

Our Solution: A Two-Layer Architecture that Serves Both Needs.

Schema__
(The Master Layer)

- Source of Truth
- ID-Based
- Graph-Friendly
- Stored in MGraph-DB

The Projector



Projected__
(The Derived Layer)

- Human-Readable
- Ref-Based
- Generated for Understanding
- Disposable Artifact

Key Insight: This is **not** a round-trip transformation. The projection is a one-way view designed for human consumption. To modify data, you always work with the `Schema__` layer.

Every Projected__ File Has an Identical Three-Section Anatomy

`projection` [redacted]

For Whom: **Humans**
Purpose: To understand the graph's nodes, edges, and relationships in plain English.

`references` [redacted]

For Whom: **Tools**
Purpose: To provide a correlation map from human-readable `refs` back to their machine `IDs`.

`sources` [redacted]

For Whom: **Debugging & Auditing**
Purpose: To provide provenance, showing exactly which `Schema__` graph was used to generate this projection.

The `projection` Section Distils Everything to Simple Nodes and Edges.

Nodes are Things

A node has a type (`ref`) and an instance identity (`name`).

```
{
  "ref": "class",
  "name": "MyClass"
}
```

Read as: A **class** named **MyClass**.

Edges are Relationships

An edge connects two nodes by their `name` and has a relationship type (`ref`).

```
{
  "from_name": "MyClass",
  "to_name": "helper_func",
  "ref": "calls"
}
```

Read as: **MyClass calls helper_func**.

The Result: From a Cryptic Mess to a Readable Story.

BEFORE: Machine-Optimised Truth

```
{
  "nodes": [
    {
      "node_id": "f0e9d...",
      "node_type_id": "9e8d7..."
    }
  ],
  "edges": [
    {
      "from_node": "f0e9d...",
      "to_node": "12345...",
      "predicate_id": "3c1a0..."
    }
  ]
}
```

**No IDs are needed to understand this graph. That is the entire point.

AFTER: Human-Readable Projection

```
{
  "projection": {
    "nodes": [
      { "ref": "class", "name": "MyClass" }
    ],
    "edges": [
      {
        "from_name": "MyClass",
        "to_name": "helper_func",
        "ref": "calls"
      }
    ]
  }
}
```

To Enable This, We First Normalised Our Schema Definitions

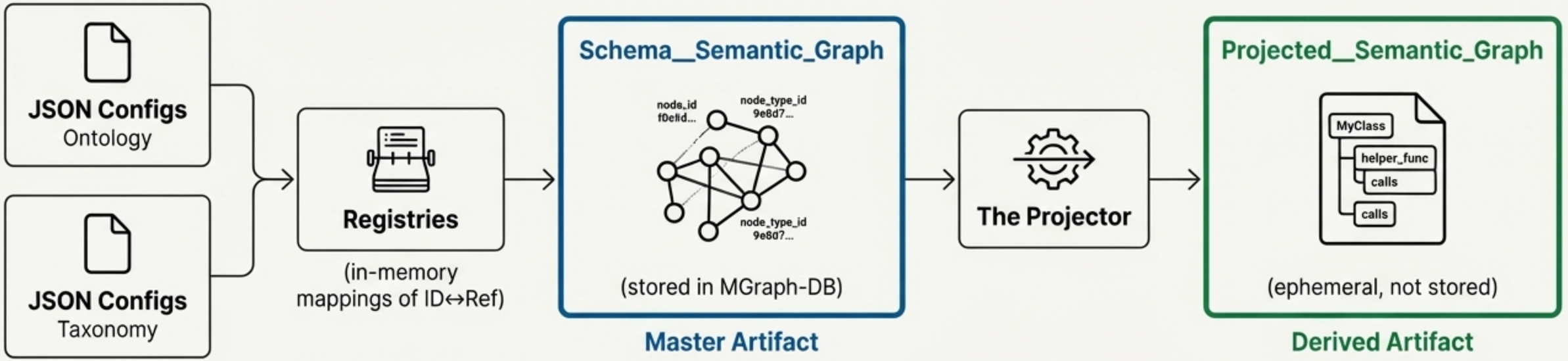
We now define predicates as first-class entities, not as repeated strings.

BEFORE (Denormalised)	AFTER (Normalized)

<pre>"node_types": {
    "class": {
        "relationships": {
            "calls": {
                "target_type": "function"
            }
        }
    }
}</pre>
	<pre>"predicates": {
	    "p-1": { "ref": "calls", "id": "..." }
	},
	"edge_rules": [
	    {
	      "source_type_id": "nt-1",
	      "predicate_id": "p-1",
	      "target_type_id": "nt-2"
	    }
	]</pre>

Issues: • Fragile to renaming • Risk of inconsistency • No single source of truth	**Benefits:** • Defined once, change in one place • Guaranteed consistency • Enables `references` mapping

The Complete Architecture: How All the Pieces Fit Together



What Goes Where

Artifact	Layer	Contains	Stored?
JSON configs	Definition	Ontology, taxonomy definitions	Yes (files)
Schema__Semantic_Graph	Instance	ID-based graph data	Yes (MGraph-DB)
Projected__Semantic_Graph	Projection	Ref-based human view	No (regenerate)

A Principled Design Delivers Tangible Benefits.

Key Design Decisions

- **`Projected__` not `View__`**: Implies one-way generation.
- **Three Sections**: Separates content, correlation, and provenance.
- **`ref` for Types, `name` for Instances**: Provides clear, consistent identity.
- **No IDs in `projection`**: Prioritises human readability above all.
- **No Round-Trip**: Reinforces that Schema__ is the single source of truth.

Benefits Achieved

- **Graph-Friendly**: `Schema__` uses IDs compatible with MGraph-DB.
- **Referential Integrity**: All cross-references in `Schema__` are robust.
- **Human-Readable**: `Projected__` is clear and requires no lookups.
- **Debuggable & Traceable**: The structure enables easy correlation and auditing.
- **Single Source of Truth**: Schema__ is master, Projected__ is derived.