What is Ontology? - Example

The Challenge

Question: "Find affordable gaming computers"

Problem: What does "affordable" and "gaming" mean?

- Different people have different definitions
- Hard-coded filters in every query
- Not reusable or maintainable

Solution: Define these concepts **once** in an ontology, then query using those concepts!

Our Data: 5 Computers

Brand	Model	Price	RAM	GPU
ASUS	TUF Gaming A15	$899	16GB	RTX4060
CyberPowerPC	Gamer Xtreme	$949	16GB	GTX1660
Alienware	M15 R7	$1499	32GB	RTX4060
HP	Pavilion 15	$599	16GB	Intel UHD
Dell	Inspiron 15	$449	8GB	Intel UHD

Which are "affordable gaming computers"?

Traditional Approach: Manual Filtering

SQL (Database):

```sql
SELECT * FROM computers
WHERE price < 1000
  AND ram >= 16
  AND gpu_type = 'dedicated'
```

SPARQL (Manual Filter):

```
SELECT ?computer WHERE {
  ?computer :hasPrice ?price .
  ?computer :hasRAM ?ram .
  ?computer :hasGPU ?gpu .
  ?gpu rdf:type :DedicatedGPU .
  FILTER(?price < 1000 && ?ram >= 16)
}
```

Problem: Rules are hard-coded in every query!

Ontology Approach: Define the Concepts

Step 1: Define what "affordable" and "gaming" mean in the ontology

```
:AffordableComputer ≡ Computer with price < 1000
:GamingComputer ≡ Computer with RAM ≥ 16 AND has DedicatedGPU
```

Step 2: Query using these concepts directly!

```
SELECT ?computer WHERE {
  ?computer rdf:type :AffordableComputer .
  ?computer rdf:type :GamingComputer .
}
```

Benefit: No manual filtering! The ontology knows what these mean!

The Ontology Structure

```
@prefix : <http://example.org/computers#> .

# Basic Classes
:Computer a owl:Class .
:Laptop rdfs:subClassOf :Computer .
:GPU a owl:Class .
:DedicatedGPU rdfs:subClassOf :GPU .

# Properties
:hasPrice a owl:DatatypeProperty .
:hasRAM a owl:DatatypeProperty .
:hasGPU a owl:ObjectProperty .
:hasBrand a owl:DatatypeProperty .
:hasModel a owl:DatatypeProperty .
```

Simple structure - just like a database schema!

Defining Semantic Classes with OWL

AffordableComputer = Computer with price < 1000

```
:AffordableComputer a owl:Class ;
    owl:equivalentClass [
        owl:intersectionOf (
            :Computer
            [
                owl:onProperty :hasPrice ;
                owl:someValuesFrom [
                    owl:onDatatype xsd:integer ;
                    owl:withRestrictions (
                        [ xsd:maxExclusive 1000 ]
                    )
                ]
            ]
        )
    ] .
```

This is **declarative** - we define what it **means**, not how to find it!

Defining Gaming Computer

GamingComputer = Computer with RAM ≥ 16 AND Dedicated GPU

```
:GamingComputer a owl:Class ;
    owl:equivalentClass [
        owl:intersectionOf (
            :Computer
            [
                owl:onProperty :hasRAM ;
                owl:someValuesFrom [
                    owl:onDatatype xsd:integer ;
                    owl:withRestrictions (
                        [ xsd:minInclusive 16 ]
                    )
                ]
            ]
            [
                owl:onProperty :hasGPU ;
                owl:someValuesFrom :DedicatedGPU
            ]
        )
```

Adding Computer Data

```
# GPU Instances
:RTX4060 a :DedicatedGPU .
:IntelUHD a :GPU .

# Computer Instance
:Computer1 a :Laptop ;
    :hasBrand "ASUS" ;
    :hasModel "TUF Gaming A15" ;
    :hasPrice 899 ;
    :hasRAM 16 ;
    :hasGPU :RTX4060 .
```

Just facts - no need to manually mark it as "affordable" or "gaming"!

The Magic: OWL Reasoning

Before Reasoning:

- Computer1 is a Laptop
- Computer1 has price 899, RAM 16, GPU RTX4060
- RTX4060 is a DedicatedGPU

OWL Reasoner checks the definitions:

1. Does Computer1 have price < 1000? ✓ YES → Computer1 IS AN AffordableComputer
2. Does Computer1 have RAM ≥ 16? ✓ YES
3. Does Computer1 have DedicatedGPU? ✓ YES → Computer1 IS A GamingComputer

After Reasoning:

- Computer1 is also an AffordableComputer
- Computer1 is also a GamingComputer

The Query (Semantic!)

```
PREFIX : <http://example.org/computers#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?brand ?model ?price ?ram
WHERE {
    # No manual filters - just query the concepts!
    ?computer rdf:type :AffordableComputer .
    ?computer rdf:type :GamingComputer .

    ?computer :hasBrand ?brand .
    ?computer :hasModel ?model .
    ?computer :hasPrice ?price .
    ?computer :hasRAM ?ram .
}
ORDER BY ?price
```

Clean, semantic, maintainable!

Running the Example

```
# Install dependencies
pip install rdflib owlrl

# Run the demo
python3 run.py
```

Output:

```
2. Applying OWL reasoning...
   ✓ After reasoning: 147 triples (inferred new facts!)

Results:
  1. ASUS TUF Gaming A15
     Price: $899, RAM: 16GB

  2. CyberPowerPC Gamer Xtreme
     Price: $949, RAM: 16GB

Total: 2 computer(s) found
```

Why These Two? Let's Check!

Computer	Price < 1000?	RAM ≥ 16?	Dedicated GPU?	Match?
ASUS TUF Gaming	✓ ($899)	✓ (16GB)	✓ (RTX4060)	**YES**
CyberPowerPC	✓ ($949)	✓ (16GB)	✓ (GTX1660)	**YES**
Alienware	× ($1499)	✓ (32GB)	✓ (RTX4060)	NO
HP Pavilion	✓ ($599)	✓ (16GB)	× (Intel UHD)	NO
Dell Inspiron	✓ ($449)	× (8GB)	× (Intel UHD)	NO

Perfect! Only 2 computers match **both** criteria.

Comparison: Manual vs Semantic

Manual Filtering (Traditional)

```
FILTER(?price < 1000 && ?ram >= 16)
```

- Rules in every query
- Hard to maintain
- Must understand the domain

Semantic Classes (Ontology)

```
?computer rdf:type :AffordableComputer .
?computer rdf:type :GamingComputer .
```

- Rules defined once in ontology
- Easy to maintain (change once!)
- Query using domain concepts

The Power of Ontology Reasoning

Scenario: Company decides "affordable" now means < $1200

Traditional Approach

```
-- Must update EVERY query (maybe 100s of queries!)
WHERE price < 1200  -- was 1000
  AND ram >= 16
```

Ontology Approach

```
# Change ONCE in the ontology
owl:withRestrictions ( [ xsd:maxExclusive 1200 ] )
```

All queries using `:AffordableComputer` automatically use new definition!

Real-World Applications

E-commerce:

- `PremiumProduct`, `BudgetProduct`, `PopularProduct`
- Query: "Find premium laptops on sale"

Healthcare:

- `HighRiskPatient`, `ChronicCondition`, `UrgentCase`
- Query: "Find high-risk patients with chronic conditions"

Smart Cities:

- `AccessibleVenue`, `FamilyFriendly`, `PetFriendly`
- Query: "Find accessible family-friendly restaurants"

Key Takeaways

1. **Ontologies encode domain knowledge**
    - Not just data, but what the data **means**
2. **OWL reasoning infers new facts**
    - Computers automatically become AffordableComputer or GamingComputer
    - Based on their properties, not manual labeling
3. **SPARQL queries become semantic**
    - Query using concepts, not filters
    - More maintainable and clear
4. **Change once, benefit everywhere**
    - Definitions in one place (ontology)

Try It Yourself!

Exercise 1: Add a new computer

```
:Computer6 a :Laptop ;
    :hasBrand "Lenovo" ;
    :hasModel "Legion 5" ;
    :hasPrice 1099 ;
    :hasRAM 16 ;
    :hasGPU :RTX4060 .
```

Will it be found? (No - price ≥ 1000!)

Exercise 2: Define `BudgetComputer` (price < $500)

Exercise 3: Write a query to find all affordable computers (gaming or not)

Summary

Question: "Find affordable gaming computers"

Traditional: Hard-code filters everywhere

```
FILTER(?price < 1000 && ?ram >= 16)
```

Ontology: Define once, query semantically

```
?computer rdf:type :AffordableComputer .
?computer rdf:type :GamingComputer .
```

This is the power of Semantic Web! 🚀

The ontology **knows** what these concepts mean and can **reason** about them!