What is Ontology?

Understanding Knowledge Representation

The Restaurant Menu Problem

Imagine you're building a food delivery app:

You have data from 100 restaurants:

- "Margherita Pizza"
- "Cheese Pizza"
- "Pizza with Tomato and Mozzarella"

Problem: Are these the same? Different? Related?

Without structure, computers can't tell!

The Solution: Define Relationships

Instead of just listing items, define what they ARE:

```
Margherita Pizza:
  - IS A Pizza
  - HAS Tomato Sauce
  - HAS Mozzarella Cheese
  - HAS Basil

Pizza:
  - IS A Food
  - HAS Crust
  - HAS Toppings
```

This is ontology: Defining what things are and how they relate

What is Ontology?

> **Ontology** = A formal way to represent knowledge about a domain

- CD in D - Conceptual Description in a Domain
- It is about schema, structure, and meaning, not just data
- Think of it as a class to define a concept, not an instance (fact) in OOP

Three key components:

1. **Concepts** → What types of things exist?
2. **Relationships** → How are things connected?
3. **Rules** → What constraints must be satisfied?

Real-World Example: University Domain

Concepts:

- Person
- Student (is a Person)
- Professor (is a Person)
- Course

Relationships:

- Student TAKES Course
- Professor TEACHES Course
- Course HAS Prerequisites

Rules:

- A student must take prerequisites before advanced courses
- A professor cannot teach more than 4 courses per semester

Why Do We Need Ontology?

Problem 1: Data Integration

You have student data from 3 systems:

```
System A: "firstName", "lastName"
System B: "name_first", "name_last"
System C: "givenName", "familyName"
```

Ontology Solution: Map all to standard concept `Person.firstName`

Problem 2: Reasoning

```
Data says:
- Alice is a Student
- Students are Persons
- Persons have Birthdate

Question: "Does Alice have a birthdate?"
```

Without Ontology: Don't know

With Ontology: Yes! (inferred from hierarchy)

Ontology vs Database

Feature	Database	Ontology
Focus	Store data	Model knowledge
Structure	Tables & columns	Concepts & relationships
Reasoning	No	Yes
Meaning	Implicit	Explicit
Change	Hard to modify	Flexible

Key difference: Ontology makes meaning explicit

Simple Example: Animal Ontology

```
Animal
├── Mammal
│   ├── Dog
│   ├── Cat
│   └── Human
└── Bird
    ├── Eagle
    └── Sparrow

Relationships:
- Dog HAS owner (Person)
- Dog EATS food (Food)
- Mammal HAS warmBlooded = True
```

The Power of Inference

What you define:

```
Rex → type → Dog
Dog → subClassOf → Mammal
Mammal → warmBlooded → True
```

What computer can infer:

```
Rex is a Mammal ✓
Rex is warmBlooded ✓
Rex is an Animal ✓
```

You describe once, get many facts for free!

Ontology Languages

Different ways to write ontologies:

1. **OWL** (Web Ontology Language) → Most powerful
2. **RDF** (Resource Description Framework) → Basic triples
3. **RDFS** (RDF Schema) → RDF + hierarchy
4. **JSON-LD** → JSON format for web

We'll mainly focus on OWL and RDF

RDF Triple Format to desecribe Facts

In Ontology, facts are expressed as: Subject - Predicate - Object

```
Student    rdf:type        owl:Class
Course     rdf:type        owl:Class
takes      rdf:type        owl:ObjectProperty
takes      rdfs:domain     Student
takes      rdfs:range      Course
hasCredit  rdf:type        owl:DatatypeProperty
hasCredit  rdfs:domain     Course
hasCredit  rdfs:range      xsd:integer
```

Ontology uses the same triple format to describe concepts and relationships.

Example Facts (Not Ontology Level Definition)

We can define facts about Student that was defined in the ontology above using the same triple format:

```
Alice    rdf:type       Student
Alice    takes          CSC101
CSC101   rdf:type       Course
CSC101   taughtBy       DrSmith
CSC101   hasCredit      3
```

- We don't need to define everything upfront, we reuse existing concepts such as `rdf:type`.
- `rdf:type` is in the ontology of RDF standard ontology (vocabulary).

When we use Turtle syntax, the above triples look like:

```
:Alice  rdf:type   :Student ;
        :takes     :CSC101 .

:CSC101 rdf:type   :Course ;
        :taughtBy  :DrSmith ;
        :hasCredit 3 .
```

Knowlede Graph: Visual Representation

```
      type
Alice -----> Student
  |
  | takes
  ↓
CSC101
  |
  | taughtBy
  ↓
Dr.Smith
```

Knowledge graph = Connected facts

Ontology Behind the Scenes

```
takes:
   domain → Student
   range  → Course

taughtBy:
   domain → Course
   range  → Instructor
```

This schema sits “under” the graph.

Building an Ontology: Step-by-Step

Problem: Model a library system

Step 1: Identify concepts

- Book, Author, Member, Loan

Step 2: Define hierarchy

- Book → FictionBook, NonFictionBook
- Member → Student, Faculty, Staff

Step 3: Define properties

```
Book:
  - hasTitle (string)
  - hasISBN (string)
  - publishedYear (integer)

Author:
  - hasName (string)
  - wrote (Book)
```

Step 4: Add constraints

- A loan cannot exceed 30 days
- A student can borrow max 5 books

Ontology Design Patterns

1. Hierarchy Pattern

```
Vehicle → Car → SportsCar
```

2. Part-Whole Pattern

```
Car HAS_PART Engine
Car HAS_PART Wheel
```

3. Attribute Pattern

```
Car HAS_COLOR Color
Car HAS_YEAR Year
```

Common Mistakes to Avoid

❌ **Too complex:** Don't model everything in detail

✓ **Start simple:** Model what you need

❌ **Inconsistent naming:** Car, automobile, vehicle

✓ **Consistent terms:** Choose one and stick to it

❌ **Missing constraints:** Allowing impossible states

✓ **Add validation:** Age must be > 0, etc.

Tools for Creating Ontologies

1. **Protégé** → Visual ontology editor (free)
2. **TopBraid Composer** → Professional tool
3. **WebProtégé** → Web-based version
4. **VS Code + Plugins** → For code-first approach

We'll use Protégé for hands-on practice

Ontology in the Real World

1. **Google Knowledge Graph**
   - Understands entities and relationships
   - Powers search results
2. **Healthcare (SNOMED CT)**
   - Medical terminology ontology
   - Ensures consistent diagnosis
3. **E-commerce (schema.org)**

version

4. **VS Code + Plugins** → For code-first approach

We'll use Protégé for hands-on practice

Ontology in the Real World

1. **Google Knowledge Graph**
   - Understands entities and relationships
   - Powers search results
2. **Healthcare (SNOMED CT)**
   - Medical terminology ontology
   - Ensures consistent diagnosis
3. **E-commerce (schema.org)**
   - Product descriptions
   - Better search results

When to Use Ontology?

Good fit:

✓ Complex domain knowledge

✓ Need for reasoning

✓ Data integration from multiple sources

✓ Requirement for explanation/traceability

Probably overkill:

✕ Simple CRUD application

✕ Single data source

✕ No complex relationships

Key Concepts Summary

Ontology provides:

1. **Structure** → Organize knowledge
2. **Semantics** → Define meaning
3. **Reasoning** → Infer new facts
4. **Integration** → Connect different systems
5. **Validation** → Ensure consistency

It's about making knowledge machine-understandable

Next Steps

In upcoming lectures we'll learn:

1. How to create ontologies in Protégé
2. How to write SPARQL queries
3. How to integrate ontology with applications
4. How to use ontology with AI systems (RAG)

Today's foundation will make those topics easier!

Key Takeaways

✓ Ontology = Structured knowledge representation

✓ Uses concepts, relationships, and rules

✓ Enables reasoning and inference

✓ Different from databases (meaning vs storage)

✓ Practical applications in AI, search, and integration

Think of ontology as a "knowledge blueprint" for computers