

## **(Optional) Ontology Examples: Diet Restrictions**

A practical ontology example demonstrating how to solve the problem of matching foods with dietary restrictions.

# The Problem

**Scenario:** Restaurant recommendation system

You have customers with different dietary restrictions:

- **Alice:** Vegetarian (cannot eat meat)
- **Bob:** Peanut allergy (cannot eat peanuts)
- **Carol:** Lactose intolerant (cannot eat dairy)
- **David:** Halal diet (cannot eat pork)
- **Eve:** Vegan (cannot eat meat, dairy, eggs)

**Question:** Which menu items can each person safely eat?

**Traditional Approach:** Hard-coded if-statements (nightmare to maintain!)

# The Ontology Solution

Instead of hard-coding rules, we model the domain knowledge explicitly:

1. Define classes (Person, Food, Ingredient, Restriction)
2. Express relationships (containsIngredient, incompatibleWith)
3. Query using SPARQL
4. Get explainable results!

# Files

- `diet.ttl` - Complete ontology with people, foods, and restrictions
- `query_alice.sparql` - What can Alice eat?
- `query_bob.sparql` - What can Bob eat?
- `query_why_not.sparql` - Why can't someone eat something?
- `query_safe_for_all.sparql` - Foods safe for everyone
- `query_compatibility_matrix.sparql` - Show all person-food matches
- `run.py` - Python demo script
- `run.sh` - Quick start script
- `shacl_shapes.ttl` - Data validation rules
- `validate.py` - SHACL validation demo

# Quick Start

```
cd topics/ontology/code/ontology/diet/  
  
# Install dependencies  
pip install rdflib  
  
# Run the demo  
chmod +x run.sh  
./run.sh  
  
# Or directly  
python3 run.py  
  
# Optional: Run SHACL validation  
pip install pyshacl  
python3 validate.py
```

# Expected Output

## Diet Restrictions Ontology Demo

---

### People and Their Restrictions

---

- Alice Johnson: Vegetarian
- Bob Smith: Peanut Allergy
- Carol Lee: Lactose Intolerance
- David Ahmed: Halal
- Eve Chen: Vegan

### Restaurant Menu

---

#### Beef Burger

##### Ingredients:

- Wheat Bread
- Beef
- Cheddar Cheese

#### Caesar Salad

##### Ingredients:

- Romaine Lettuce
- Cheddar Cheese
- Chicken

[... more foods ...]

```
=====
Query 1: What can Alice (Vegetarian) eat?
=====
```

1. Cheese Pizza
2. Garden Salad
3. Vegetable Stir Fry

Total: 3 result(s)

```
=====
Query 2: What can Bob (Peanut Allergy) eat?
=====
```

1. Beef Burger
2. Caesar Salad
3. Cheese Pizza
4. Garden Salad
5. Pork Chop
6. Vegetable Stir Fry

Total: 6 result(s)

```
=====
Query 3: What can EVERYONE eat? (Safe for all restrictions)
=====
```

1. Garden Salad
2. Vegetable Stir Fry

Total: 2 result(s)



# The Data Model

## Class Hierarchy

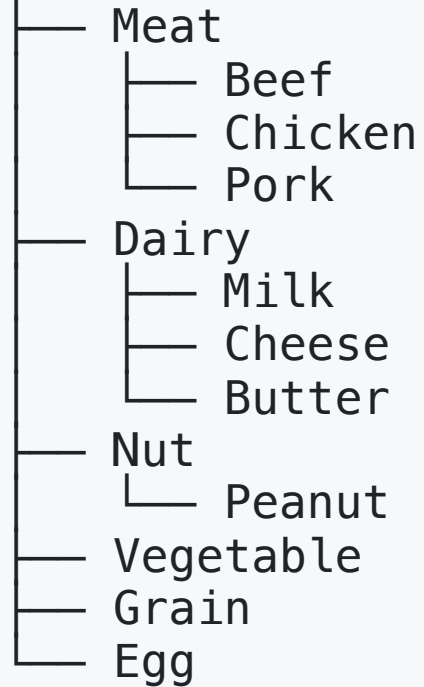
Person

- hasName (string)
- hasRestriction (Restriction)

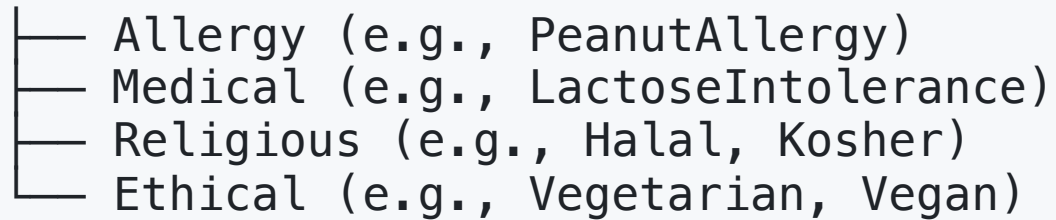
Food

- foodName (string)
- containsIngredient (Ingredient)

## Ingredient



## Restriction



## Restriction Rules

```
# Vegetarian cannot eat meat
diet:VegetarianRestriction diet:incompatibleWith diet:Meat .

# Vegan cannot eat meat, dairy, or eggs
diet:VeganRestriction diet:incompatibleWith diet:Meat ;
                    diet:incompatibleWith diet:Dairy ;
                    diet:incompatibleWith diet:Egg .

# Lactose intolerance incompatible with dairy
diet:LactoseIntoleranceRestriction diet:incompatibleWith diet:Dairy .

# Halal restrictions
diet:HalalRestriction diet:incompatibleWith diet:Pork .

# Peanut allergy
diet:PeanutAllergyRestriction diet:incompatibleWith diet:Peanut .
```

# Example Queries

## Query 1: What Can Alice Eat?

```
SELECT ?foodName
WHERE {
    diet:Alice diet:hasRestriction ?restriction .
    ?food rdf:type diet:Food ;
        diet:foodName ?foodName .

    # Filter out incompatible foods
    FILTER NOT EXISTS {
        ?food diet:containsIngredient ?ingredient .
        ?restriction diet:incompatibleWith ?category .
        ?ingredient rdf:type ?category .
    }
}
```

**Result:** Cheese Pizza, Garden Salad, Vegetable Stir Fry

## Query 2: Why Can't Alice Eat Beef Burger?

```
SELECT ?restrictionLabel ?ingredientLabel
WHERE {
    diet:Alice diet:hasRestriction ?restriction .
    diet:BeefBurger diet:containsIngredient ?ingredient .

    ?restriction diet:incompatibleWith ?category .
    ?ingredient rdf:type ?category .

    ?restriction rdfs:label ?restrictionLabel .
    ?ingredient rdfs:label ?ingredientLabel .
}
```

**Result:** Vegetarian restriction, Beef ingredient

**Explanation:** "Alice can't eat Beef Burger because she's Vegetarian and it contains Beef"

### Query 3: Foods Safe for Everyone

```
SELECT ?foodName
WHERE {
    ?food rdf:type diet:Food ;
        diet:foodName ?foodName .

    # Must be compatible with ALL people
    FILTER NOT EXISTS {
        ?person diet:hasRestriction ?restriction .
        ?food diet:containsIngredient ?ingredient .
        ?restriction diet:incompatibleWith ?category .
        ?ingredient rdf:type ?category .
    }
}
```

**Result:** Garden Salad, Vegetable Stir Fry

# SHACL Validation

The ontology includes SHACL shapes to ensure data quality:

## Validation Rules

1. **Person must have a name** (non-empty string)
2. **Person must have at least one restriction**
3. **Food must have a name**
4. **Food must contain at least one ingredient**
5. **Restriction must define incompatibilities**

## Example: Invalid Data

```
# Invalid – Person without name
diet:InvalidPerson a diet:Person ;
    diet:hasRestriction diet:VegetarianRestriction .
# ✗ Violation: Person must have a name

# Invalid – Food without ingredients
diet:MysteryDish a diet:Food ;
    diet:foodName "Mystery Dish" .
# ✗ Violation: Food must contain at least one ingredient
```

Run `python3 validate.py` to check data quality!



# Key Benefits

## 1. Explicit Domain Knowledge

```
# Clear definition of what restrictions mean  
diet:VegetarianRestriction diet:incompatibleWith diet:Meat .
```

## 2. Flexible Queries

Can ask any question:

- What can X eat?
- Why can't X eat Y?
- What's safe for everyone?
- Show all compatibility pairs

### 3. Explainable Results

Alice can't eat Beef Burger because:

- Alice has Vegetarian restriction
- Vegetarian is incompatible with Meat
- Beef Burger contains Beef
- Beef is a type of Meat

## 4. Easy to Extend

### Add a new restriction:

```
diet:GlutenFree a diet:Medical ;  
    diet:incompatibleWith diet:Wheat .
```

### Add a new person:

```
diet:Frank a diet:Person ;  
    diet:hasName "Frank Wilson" ;  
    diet:hasRestriction diet:GlutenFreeRestriction .
```

### Add a new food:

```
diet:QuinoaBowl a diet:Food ;  
    diet:foodName "Quinoa Bowl" ;  
    diet:containsIngredient diet:Quinoa ;  
    diet:containsIngredient diet:Avocado .
```

## 5. No Hard-Coded If-Statements!

Traditional approach:

```
boolean canEat(Person p, Food f) {  
    if (p.isVegetarian() && f.hasMeat()) return false;  
    if (p.hasAllergy("peanut") && f.has("peanut")) return false;  
    if (p.isLactoseIntolerant() && f.hasDairy()) return false;  
    // 100 more if statements...  
    return true;  
}
```

## Ontology approach:

```
# Just query the knowledge graph!
SELECT ?food WHERE {
  ?person diet:hasRestriction ?r .
  ?food diet:foodName ?name .
  FILTER NOT EXISTS {
    ?food diet:containsIngredient ?i .
    ?r diet:incompatibleWith ?c .
    ?i rdf:type ?c .
  }
}
```