Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №3

«Построение онтологической модели»

Вариант № 21

Выполнила студентка

группы ИВТИИБД-11

Полина.П.П

Преподаватель

Хайруллин. И. Д

1. **Структурное описание онтологической модели**

Онтологическая модель описывает систему работы ресторана и формализует ключевые объекты и их взаимосвязи. Основные сущности (далее классы) модели включают:

* Официанты (Wailters) – принимают заказы
* Повара (Chefs) – готовят блюда
* Заказы (Orders) – что хочет клиент
* Блюда (Dishes) – что готовит ресторан

Для реализации модели использованы два инструмента:

1. **Protégé** — применяется для построения OWL-онтологии, добавления аннотаций и формулировки логических правил с использованием SWRL.
2. **Neo4j** — используется для создания графовой модели с узлами и связями, где логические зависимости реализуются с помощью языка запросов Cypher.

**2. Перечень классов, отношений и аксиом**

Онтологическая модель включает несколько типов элементов: классы (сущности), свойства данных и свойства объектов, а также логические правила (аксиомы), описывающие взаимосвязи между объектами.

**2.1 Классы (Classes)**

* RestaurantStaff (Персонал ресторана) - Абстрактный класс-родитель для всех сотрудников ресторана
* Waiter (Официант) - Сотрудник, принимающий заказы и обслуживающий клиентов
* Chef (Повар) - Сотрудник, отвечающий за приготовление блюд
* Order (Заказ) - Заказ от клиента, содержащий одно или несколько блюд
* Dish (Блюдо) - Блюдо из меню ресторана
* OrderStatus (Статус заказа) - Перечисление возможных статусов заказа
* DishStatus (Статус блюда) - Статус приготовления конкретного блюда в заказе

**2.2 Свойства данных (Data Properties)**

1. Свойства заказа

· hasOrderID → xsd:integer

· hasTableNumber → xsd:integer

· hasOrderTime → xsd:dateTime

· hasTotalAmount → xsd:decimal

· hasEstimatedTime → xsd:integer

2.Свойства блюда

· hasDishID → xsd:integer

· hasDishName → xsd:string

· hasPrice → xsd:decimal

· hasCookingTime → xsd:integer

· hasCategory → xsd:string

3.Свойства сотрудников

· hasEmployeeID → xsd:integer

· hasName → xsd:string

· hasExperience → xsd:string

· hasShift → xsd:string

* 1. **Свойства объектов (Object Properties)**
* takesOrder (принимает заказ) - Официант принимает заказ от клиента
* preparesDish (готовит блюдо) - Повар готовит конкретное блюдо
* containsDish (содержит блюдо) - Заказ содержит одно или несколько блюд
* hasStatus (статус заказа) – Заказ №n имеет статус «В ожидании».
  1. **Аксиомы (SWRL-правила) для Protégé**

Каждый официант принимает хотя бы один заказ

:Waiter rdfs:subClassOf

[ rdf:type owl:Restriction ;

owl:onProperty :takesOrder ;

owl:someValuesFrom :Order

] .

Каждый заказ содержит хотя бы одно блюдо

:Order rdfs:subClassOf

[ rdf:type owl:Restriction ;

owl:onProperty :containsDish ;

owl:someValuesFrom :Dish

] .

Каждый повар готовит хотя бы одно блюдо

:Chef rdfs:subClassOf

[ rdf:type owl:Restriction ;

owl:onProperty :preparesDish ;

owl:someValuesFrom :Dish

] .

* 1. **Аналог аксиом для Neo4j (Cypher-запросы)**

**Официанты принимают заказы**

**MATCH (w:Waiter {name: "Джон"}), (o:Order {id: 1})**

**CREATE (w)-[r1:ПРИНИМАЕТ\_ЗАКАЗ {time: "2024-01-15 12:30:00"}]->(o)**

**MATCH (w:Waiter {name: "Анна"}), (o:Order {id: 2})**

**CREATE (w)-[r2:ПРИНИМАЕТ\_ЗАКАЗ {time: "2024-01-15 12:45:00"}]->(o)**

**MATCH (w:Waiter {name: "Михаил"}), (o:Order {id: 3})**

**CREATE (w)-[r3:ПРИНИМАЕТ\_ЗАКАЗ {time: "2024-01-15 13:00:00"}]->(o)**

**MATCH (w:Waiter {name: "Джон"}), (o:Order {id: 4})**

**CREATE (w)-[r4:ПРИНИМАЕТ\_ЗАКАЗ {time: "2024-01-15 13:15:00"}]->(o)**

**RETURN w, o, r1, r2, r3, r4**

**Заказы содержат блюда**

**MATCH (o:Order {id: 1}), (d:Dish {name: "Пицца Маргарита"})**

**CREATE (o)-[c1:СОДЕРЖИТ {quantity: 1, notes: "без оливок"}]->(d)**

**MATCH (o:Order {id: 1}), (d:Dish {name: "Салат Цезарь"})**

**CREATE (o)-[c2:СОДЕРЖИТ {quantity: 2, notes: "дополнительный соус"}]->(d)**

**MATCH (o:Order {id: 2}), (d:Dish {name: "Лосось на гриле"})**

**CREATE (o)-[c3:СОДЕРЖИТ {quantity: 1, notes: "с лимоном"}]->(d)**

**MATCH (o:Order {id: 3}), (d:Dish {name: "Тирамису"})**

**CREATE (o)-[c4:СОДЕРЖИТ {quantity: 3, notes: ""}]->(d)**

**MATCH (o:Order {id: 4}), (d:Dish {name: "Борщ"})**

**CREATE (o)-[c5:СОДЕРЖИТ {quantity: 1, notes: "со сметаной"}]->(d)**

**MATCH (o:Order {id: 4}), (d:Dish {name: "Пицца Маргарита"})**

**CREATE (o)-[c6:СОДЕРЖИТ {quantity: 1, notes: ""}]->(d)**

**RETURN o, d, c1, c2, c3, c4, c5, c6**

**Повара готовят блюда**

**MATCH (c:Chef {name: "Мария"}), (d:Dish {name: "Салат Цезарь"})**

**CREATE (c)-[p1:ГОТОВИТ {start\_time: "2024-01-15 12:35:00", end\_time: "2024-01-15 12:45:00"}]->(d)**

**MATCH (c:Chef {name: "Алексей"}), (d:Dish {name: "Пицца Маргарита"})**

**CREATE (c)-[p2:ГОТОВИТ {start\_time: "2024-01-15 12:40:00", end\_time: "2024-01-15 13:00:00"}]->(d)**

**MATCH (c:Chef {name: "Алексей"}), (d:Dish {name: "Лосось на гриле"})**

**CREATE (c)-[p3:ГОТОВИТ {start\_time: "2024-01-15 12:50:00", end\_time: "2024-01-15 13:05:00"}]->(d)**

**MATCH (c:Chef {name: "Ольга"}), (d:Dish {name: "Тирамису"})**

**CREATE (c)-[p4:ГОТОВИТ {start\_time: "2024-01-15 13:05:00", end\_time: "2024-01-15 13:10:00"}]->(d)**

**MATCH (c:Chef {name: "Мария"}), (d:Dish {name: "Борщ"})**

**CREATE (c)-[p5:ГОТОВИТ {start\_time: "2024-01-15 13:20:00", end\_time: "2024-01-15 13:45:00"}]->(d)**

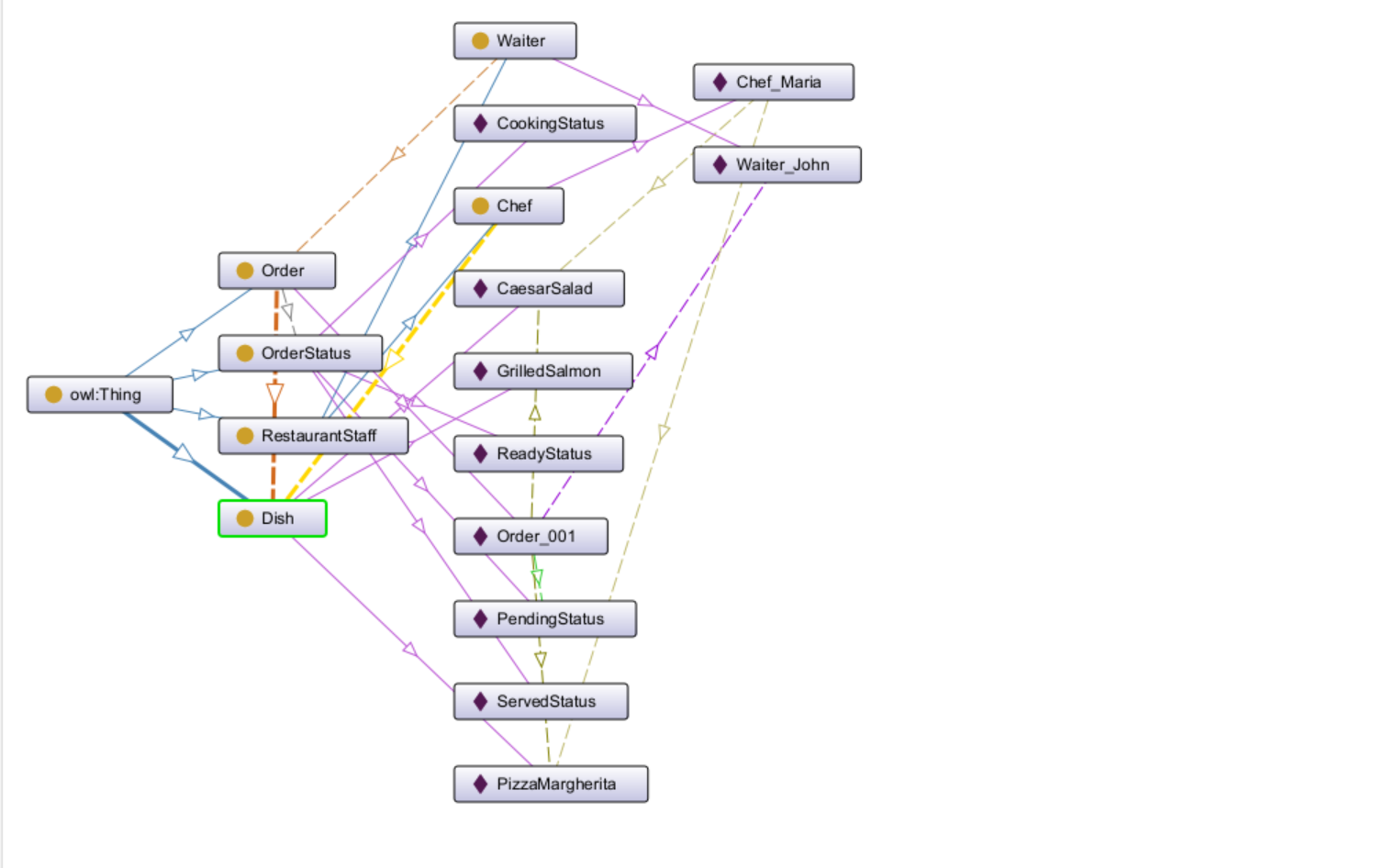
**RETURN c, d, p1, p2, p3, p4, p5**

**Эти запросы автоматически создают новые связи в графовой базе, аналогично SWRL-правилам в Protégé, что позволяет выполнять логический вывод и анализ взаимодействий между сущностями.**

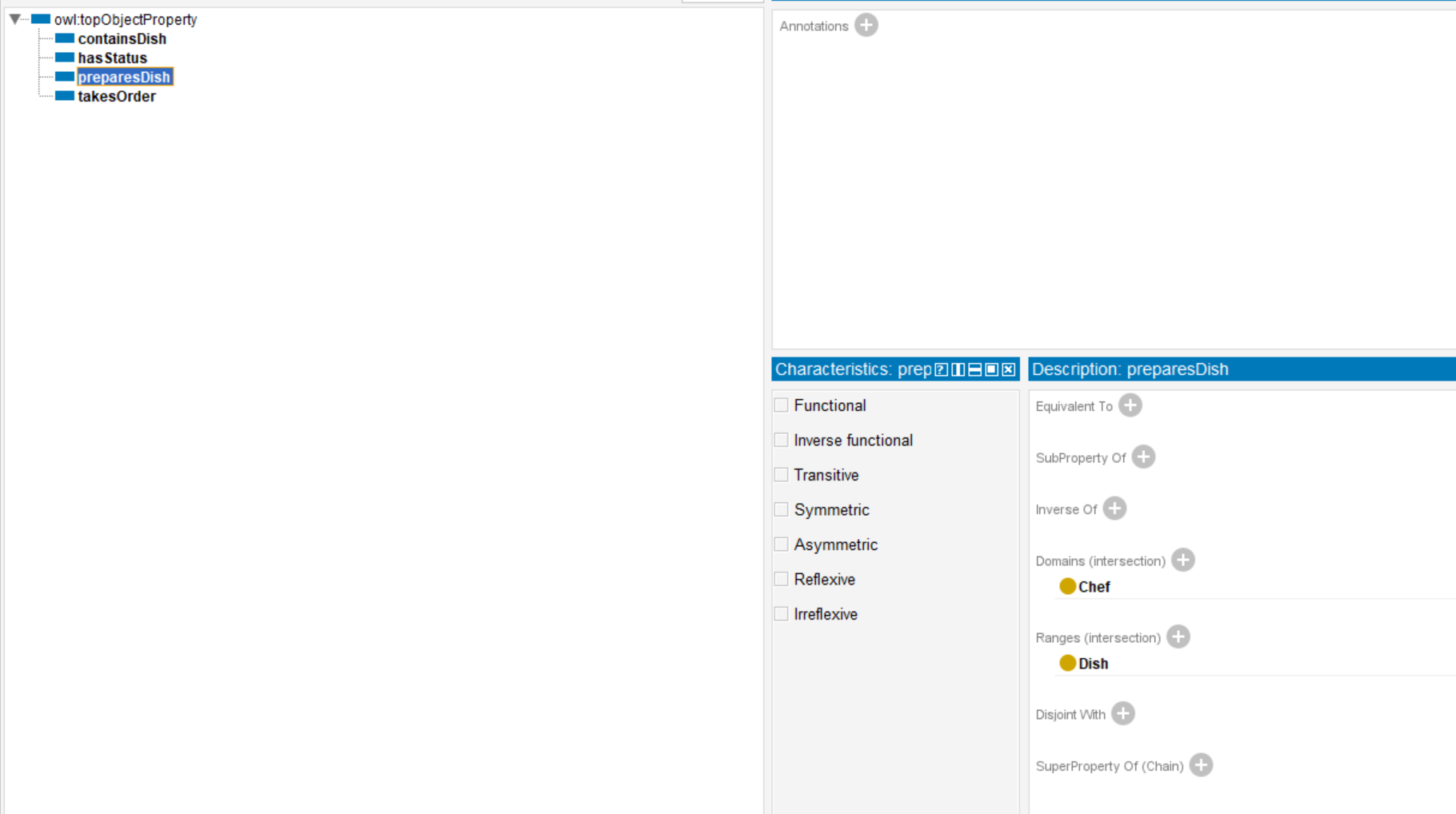
1. **Реализация в Protégé**

**Для построения онтологической модели использовался инструмент Protégé, который позволяет создавать OWL-онтологии, задавать свойства и правила, а также выполнять логический вывод.**

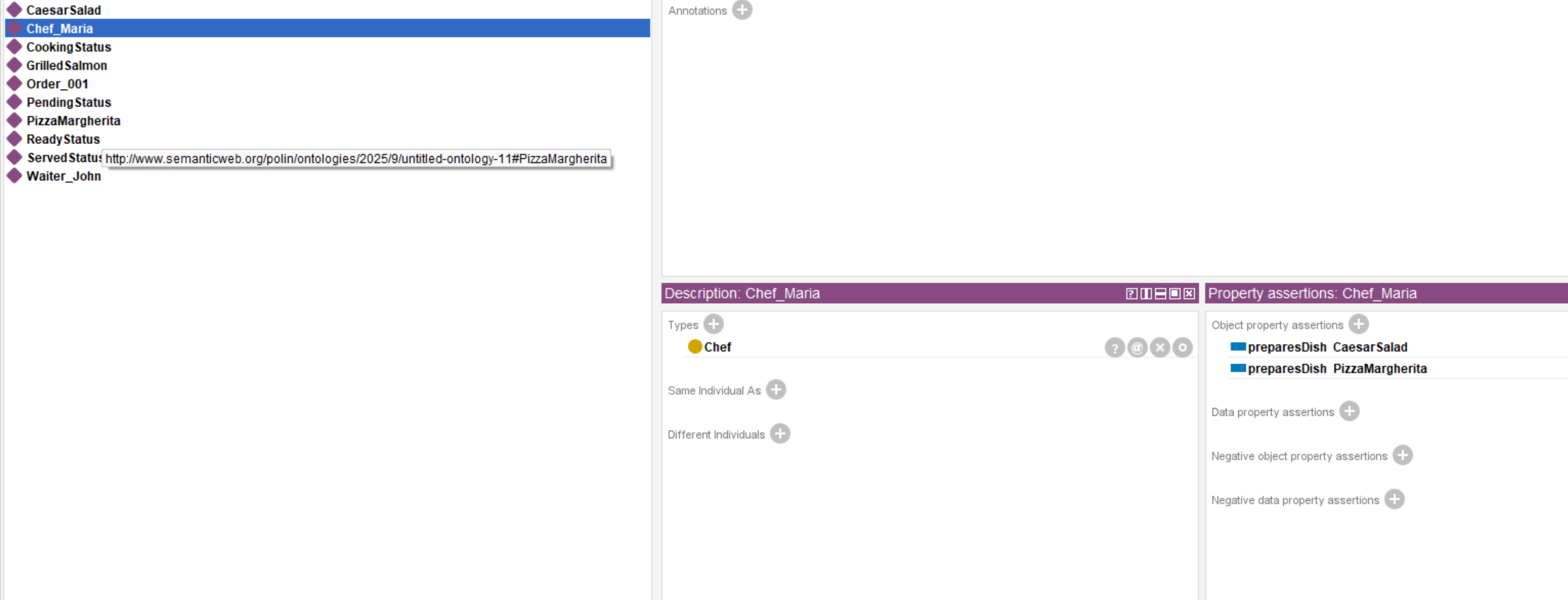
**Рисунок 1 – Дерево классов**

****

**Рисунок 2 – Object Properties**



**Рисунок 3 – Individuals**



**Рисунок 5 – Демонстрация связей**



**5.Вывод**

В работе создана онтологическая модель работы ресторана, объединяющая официантов, клиентов с их заказами, поваров и блюда