**Министерство науки и высшего образования Российской Федерации**

**ФГБОУ ВО Ульяновский государственный технический университет**

**Кафедра «Вычислительная техника»**

**Лабораторная работа №3**

**«Построение онтологической модели»**

**Вариант №21**

**Выполнил студент**

**группы ИВТИИбд-11**

**Родионов М.Ю.**

**Преподаватель**

**Хайруллин И.Д.**

**Ульяновск, 2025**

## ****1. Структурное описание онтологической модели****

**Онтологическая модель описывает систему управления рестораном и формализует ключевые объекты и их взаимосвязи. Основные сущности (классы) модели включают:**

* **Официанты (Waiter): сотрудники, принимающие заказы и обслуживающие гостей.**
* **Повара (Chef): сотрудники, ответственные за приготовление блюд.**
* **Заказы (Order): объекты, представляющие заказы гостей, содержащие одно или несколько блюд.**
* **Блюда (Dish): позиции из меню ресторана, которые могут быть отнесены к категориям (основные блюда, закуски, десерты, напитки).**
* **Столы (Table): объекты, представляющие столы в зале ресторана.**
* **Статусы (Status): этапы обработки заказа (принят, готовится, готов, подан, оплачен).**

**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****

### ****2.1 Классы (Classes)****

**Модель содержит следующие основные классы:**

* **Waiter (Официант): сотрудник, принимающий заказы и обслуживающий гостей.**
* **Chef (Повар): сотрудник, готовящий блюда.**
* **Order (Заказ): объект, содержащий одно или несколько блюд.**
* **Dish (Блюдо): позиция из меню ресторана.**
* **Table (Стол): стол в зале ресторана.**
* **Status (Статус): этап обработки заказа.**

### ****2.2 Свойства данных (Data Properties)****

| **Свойство** | **Принадлежит классу** | **Описание** |
| --- | --- | --- |
| **hasName** | **Staff** | **Имя сотрудника** |
| **hasPrice** | **Dish** | **Цена блюда** |
| **hasCookingTime** | **Dish** | **Время приготовления в минутах** |
| **hasOrderTime** | **Order** | **Время создания заказа** |
| **hasTableNumber** | **Table** | **Номер стола** |
|  |  |  |

### ****2.3 Свойства объектов (Object Properties)****

| **Свойство** | **Описание** |
| --- | --- |
| **hasWaiter** | **Заказ обслуживается официантом** |
| **hasChef** | **Блюдо готовится поваром** |
| **containsDish** | **Заказ содержит блюдо** |
| **hasStatus** | **Заказ имеет статус** |
| **assignedToTable** | **Заказ привязан к столу** |
| **preparesDish** | **Повар готовит блюдо** |

### ****2.4 Аксиомы (SWRL-правила) для Protégé****

1. **Связь заказа с официантом:  
   Order(?o) ^ Waiter(?w) ^ hasWaiter(?o, ?w) → servedBy(?w, ?o)**
2. **Связь повара с блюдом:  
   Chef(?c) ^ Dish(?d) ^ preparesDish(?c, ?d) → cooks(?c, ?d)**
3. **Автоматическое обновление статуса заказа:  
   Order(?o) ^ Status(?s) ^ hasStatus(?o, ?s) → nextStatus(?o, ?s)**

### ****2.5 Аналог аксиом для Neo4j (Cypher-запросы)****

1. **Связь официанта с заказом:**

**cypher**

**MATCH (o:Order)-[:HAS\_WAITER]->(w:Waiter)**

**MERGE (w)-[:SERVED\_ORDER]->(o);**

1. **Связь повара с блюдом:**

**cypher**

**MATCH (c:Chef)-[:PREPARES\_DISH]->(d:Dish)**

**MERGE (c)-[:COOKS]->(d);**

1. **Обновление статуса заказа:**

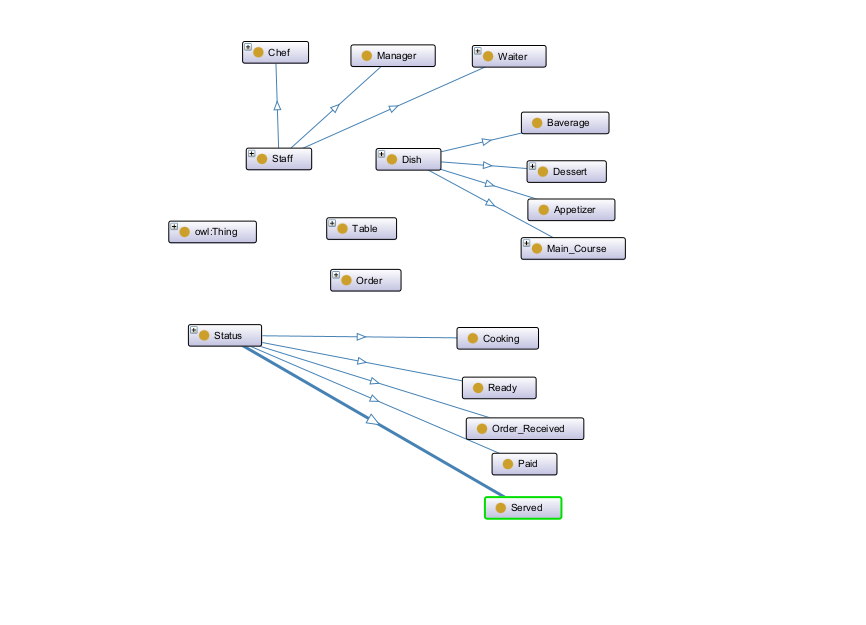
**cypher**

**MATCH (o:Order)-[r:HAS\_STATUS]->(s:Status)**

**WHERE s.name = "Готовится"**

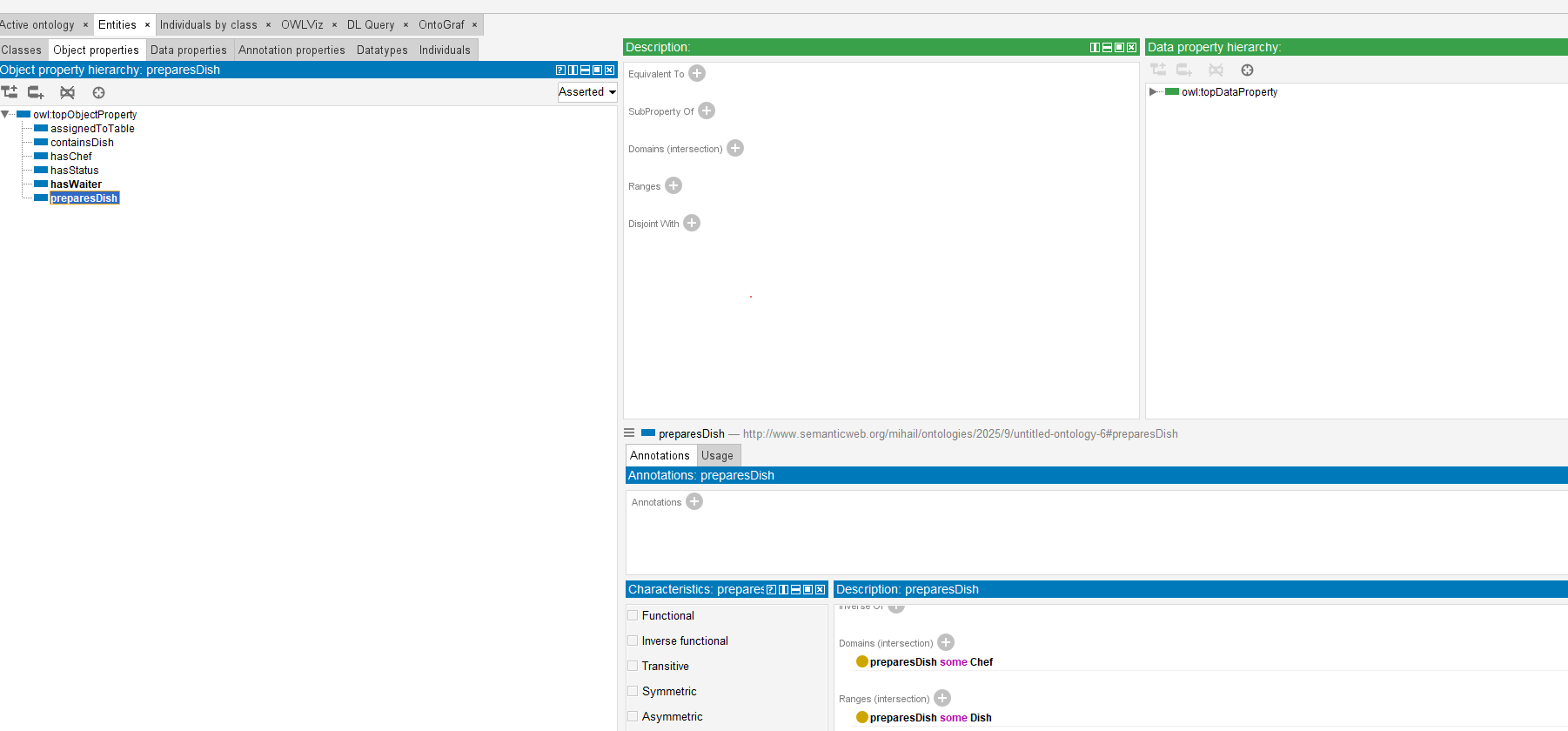
**MERGE (o)-[:NEXT\_STATUS]->(s);**

* **Отображает иерархию всех классов модели: Staff, Waiter, Chef, Order, Dish, Table, Status.**
* **Класс Staff является родительским для Waiter и Chef.**

****

**Рисунок 1 – Дерево классов**

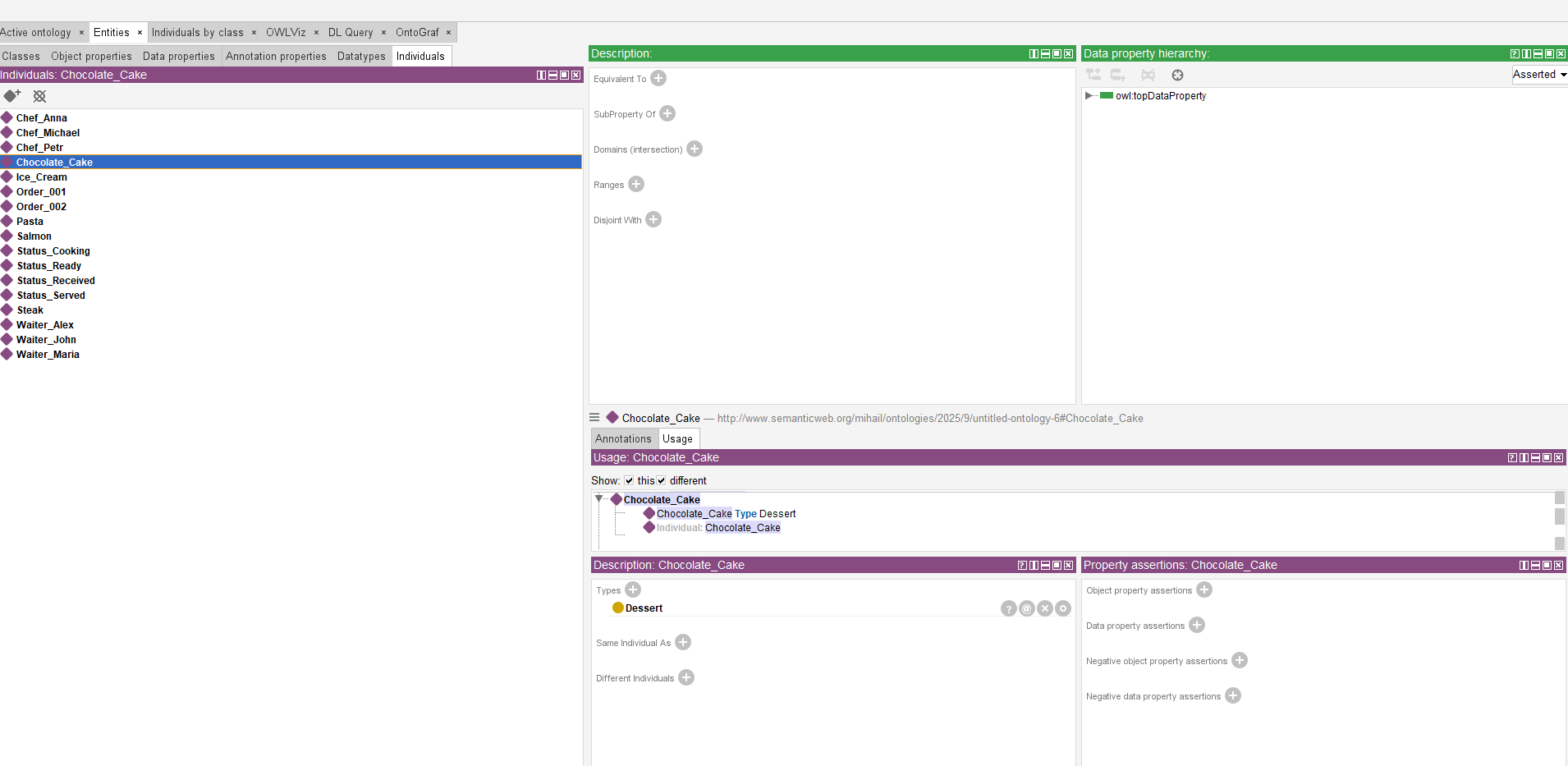
**Показывает все свойства объектов и их направление, включая: hasWaiter, hasChef, containsDish, hasStatus, assignedToTable, preparesDish.**

****

**Рисунок 2 – Object Properties**

### ****Individuals****

* **Конкретные экземпляры классов:**
  + **Официанты: Иван Петров, Мария Сидорова.**
  + **Повара: Петр Иванов, Анна Козлова.**
  + **Заказы: Order\_001, Order\_002.**
  + **Блюда: Стейк Рибай, Паста Карбонара, Салат Цезарь.**
  + **Столы: Table\_1, Table\_2.**
  + **Статусы: Принят, Готовится, Готов.**

****

**Рисунок 3 – Individuals**

## ****4. Реализация в Neo4j****

### **Создание узлов и связей (Cypher):**

**cypher**

**CREATE**

**(waiter1:Waiter:Staff {id: 1, name: "Иван Петров", experience: 3}),**

**(waiter2:Waiter:Staff {id: 2, name: "Мария Сидорова", experience: 2}),**

**(chef1:Chef:Staff {id: 3, name: "Петр Иванов", specialization: "Горячий цех"}),**

**(chef2:Chef:Staff {id: 4, name: "Анна Козлова", specialization: "Холодный цех"}),**

**(dish1:Dish {id: 101, name: "Стейк Рибай", price: 25.50, cooking\_time: 15}),**

**(dish2:Dish {id: 102, name: "Паста Карбонара", price: 18.00, cooking\_time: 12}),**

**(dish3:Dish {id: 103, name: "Салат Цезарь", price: 12.00, cooking\_time: 8}),**

**(table1:Table {id: 1, number: 1, capacity: 4, location: "У окна"}),**

**(table2:Table {id: 2, number: 2, capacity: 2, location: "В центре зала"}),**

**(status1:Status {name: "Принят"}),**

**(status2:Status {name: "Готовится"}),**

**(status3:Status {name: "Готов"}),**

**(order1:Order {id: 1001, order\_time: timestamp(), total\_amount: 45.50, guest\_count: 2}),**

**(order2:Order {id: 1002, order\_time: timestamp(), total\_amount: 34.50, guest\_count: 4}),**

**(order1)-[:HAS\_WAITER]->(waiter1),**

**(order2)-[:HAS\_WAITER]->(waiter2),**

**(order1)-[:ASSIGNED\_TO\_TABLE]->(table1),**

**(order2)-[:ASSIGNED\_TO\_TABLE]->(table2),**

**(order1)-[:HAS\_STATUS]->(status1),**

**(order2)-[:HAS\_STATUS]->(status2),**

**(order1)-[:CONTAINS\_DISH]->(dish1),**

**(order1)-[:CONTAINS\_DISH]->(dish3),**

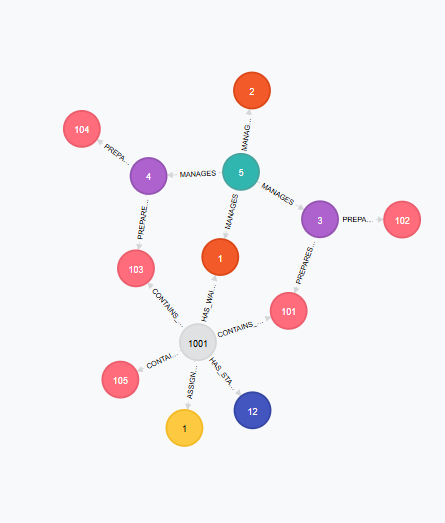
**(order2)-[:CONTAINS\_DISH]->(dish2),**

**(chef1)-[:PREPARES\_DISH]->(dish1),**

**(chef1)-[:PREPARES\_DISH]->(dish2),**

**(chef2)-[:PREPARES\_DISH]->(dish3);**

### **Демонстрация связей**

* **Визуализация графа показывает все узлы и их связи.**
* **Легко видеть, какие заказы обслуживают официанты, какие блюда готовят повара, какие столы заняты, и какие статусы имеют заказы.**
* ****

**Рисунок 5 – Демонстрация связей**

**5. Вывод**

В работе создана онтологическая модель управления рестораном, объединяющая **официантов, поваров, заказы, блюда, столы и статусы**.

**SWRL-правила** и **Cypher-запросы** позволяют автоматически выводить обратные и косвенные связи:

* официанты ↔ заказы,
* повара ↔ блюда,
* заказы ↔ статусы.

Модель наглядно реализована в **Protégé** и **Neo4j**, что подтверждается визуализацией и скриншотами.

Практическая ценность: отслеживание рабочих процессов в ресторане, управление заказами, анализ эффективности персонала и возможность расширения модели для автоматизации управления рестораном.