Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №3

«Построение онтологической модели»

Вариант №15

Выполнил студент

группы ИВТИИбд-11

Кяжкин А.А.

Преподаватель

Хайрулин И.Д.

Ульяновск, 2025

**1. Структурное описание онтологической модели**

Онтологическая модель описывает систему логистической цепочки и формализует ключевые объекты и их взаимосвязи в процессе поставок товаров от производителей к конечным потребителям. **Основные сущности (далее классы) модели включают:**

1. Заводы (Factory): производственные предприятия, создающие товары и осуществляющие поставки на склады.
2. Склады (Warehouse): объекты хранения товаров, принимающие продукцию от заводов и распределяющие ее по магазинам.
3. Магазины (Store): точки продаж, получающие товары со складов и реализующие их конечным потребителям.
4. Товары (Product): продукция, производимая заводами, хранимая на складах и продаваемая в магазинах.
5. Транспорт (Transport): средства доставки, обеспечивающие перемещение товаров между объектами цепочки.

**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)**

**Модель содержит следующие основные классы:**

1. Factory (Завод): представляет производственные предприятия, которые создают товары и осуществляют поставки.
2. Warehouse (Склад): объекты для хранения товаров с определенной вместимостью.
3. Store (Магазин): точки розничной продажи, получающие товары со складов.
4. Product (Товар): производимая продукция, которая проходит через всю логистическую цепочку.
5. Transport (Транспорт): средства доставки с определенной грузоподъемностью.

Каждый класс является фундаментальной единицей модели и служит основой для определения свойств и отношений.

**2.2 Свойства данных (Data Properties)**

Свойства данных описывают характеристики отдельных объектов (индивидуумов) классов:

|  |  |  |
| --- | --- | --- |
| **Свойство** | **Принадлежит классу** | **Описание / Комментарий** |
| hasName | Factory, Warehouse, Store | Наименование объекта |
| hasAddress | Factory, Store | Физический адрес расположения |
| hasCapacity | Factory, Warehouse | Производственная/складская |
| hasLocation | Warehouse | Местоположение склада |
| hasPrice | Product | Цена товара |
| hasWeight | Product | Вес товара |
| hasType | Store, Transport | Тип магазина/транспорта |

Эти свойства позволяют хранить конкретные данные для каждого объекта и использовать их при поиске и логическом выводе.

**2.3 Свойства объектов (Object Properties)**

**Свойства объектов определяют связи между различными объектами модели:**

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| produces | Завод производит товар |
| stores | Склад хранит товар |
| sells | Магазин продает товар |
| supplies | Объект осуществляет поставки |
| transports | Транспорт перевозит товары |
| usesTransport | Завод использует транспорт |
| deliversTo | Транспорт доставляет до объекта |

Эти свойства обеспечивают логическую структуру модели и позволяют выводить новые связи между объектами.

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

**Аксиомы формализуют правила логического вывода в модели OWL:**

**1. Автоматическое создание цепочки поставок:**

Factory(?f) ^ Product(?p) ^ Warehouse(?w) ^ produces(?f, ?p) ^ stores(?w, ?p) → supplies(?f, ?w)

Если завод производит товар, а склад хранит этот товар, то автоматически устанавливается связь поставки между заводом и складом.

**2. Связь магазина с производителем через товар:**

Store(?s) ^ Product(?p) ^ Factory(?f) ^ sells(?s, ?p) ^ produces(?f, ?p) → connectedTo(?s, ?f)

Если магазин продает товар, а завод производит этот товар, то автоматически устанавливается связь между магазином и заводом.

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

Для графовой модели логические выводы реализуются с помощью запросов:

**1. Автоматическое создание цепочек поставок:**

```cypher

MATCH (f:Factory)-[:PRODUCES]->(p:Product)<-[:STORES]-(w:Warehouse)

MERGE (f)-[:SUPPLIES]→(w);

**2. Связь магазинов с производителями:**

MATCH (s:Store)-[:SELLS]->(p:Product)<-[:PRODUCES]-(f:Factory)

MERGE (s)-[:CONNECTED\_TO]->(f);

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

**3. 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**

* Отображает иерархию всех классов модели: Logistic\_Chain, Facility, Order, Product.
* Класс Logistic\_Chain является корневым узлом, объединяющим все сущности логистической системы.
* Видно, как классы структурированы и какие объекты могут быть связаны между собой.

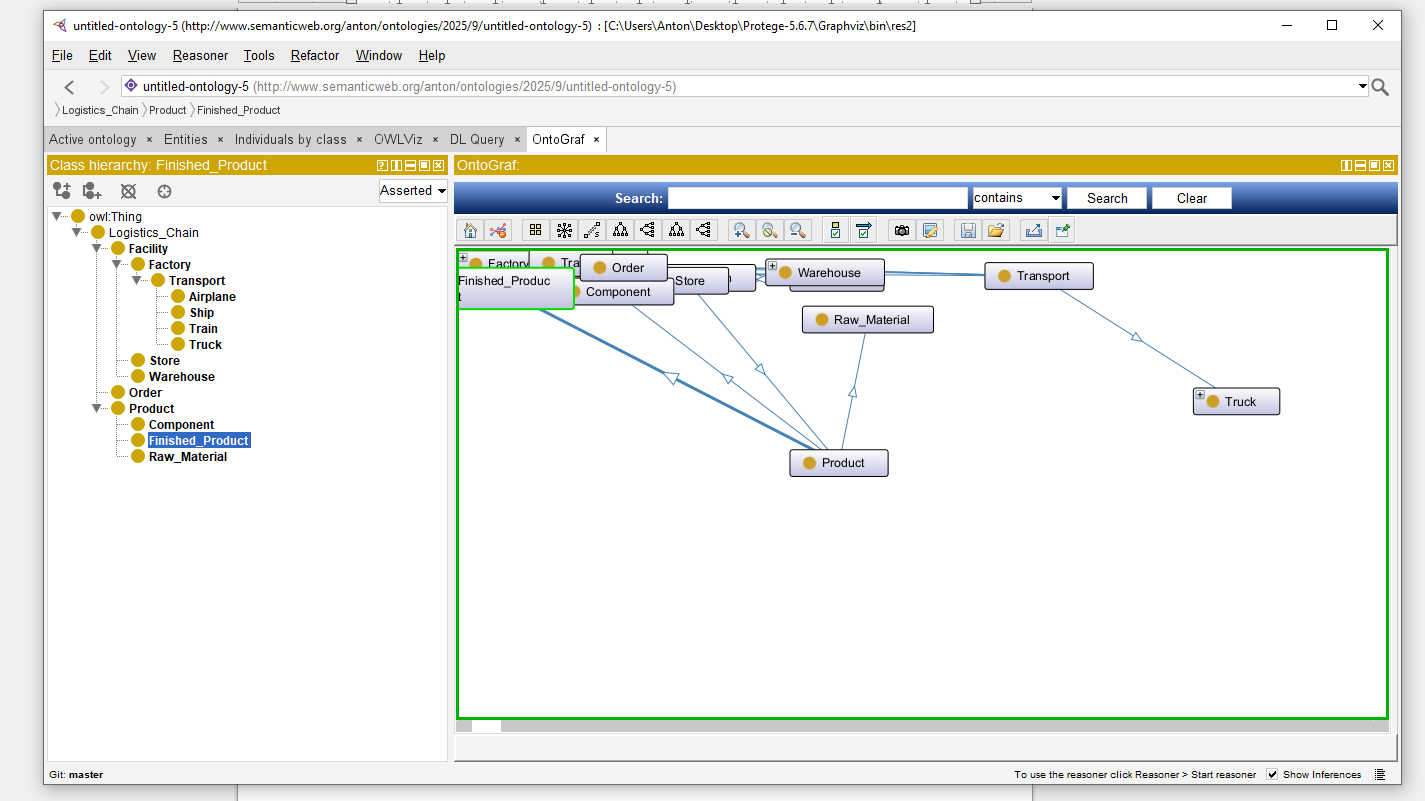


Рисунок 1. Дерево классов

**Рисунок 2. Object Properties**

Показывает все свойства объектов и их направление, включая:

* produces / stores / sells
* supplies
* transports / usesTransport / deliversTo

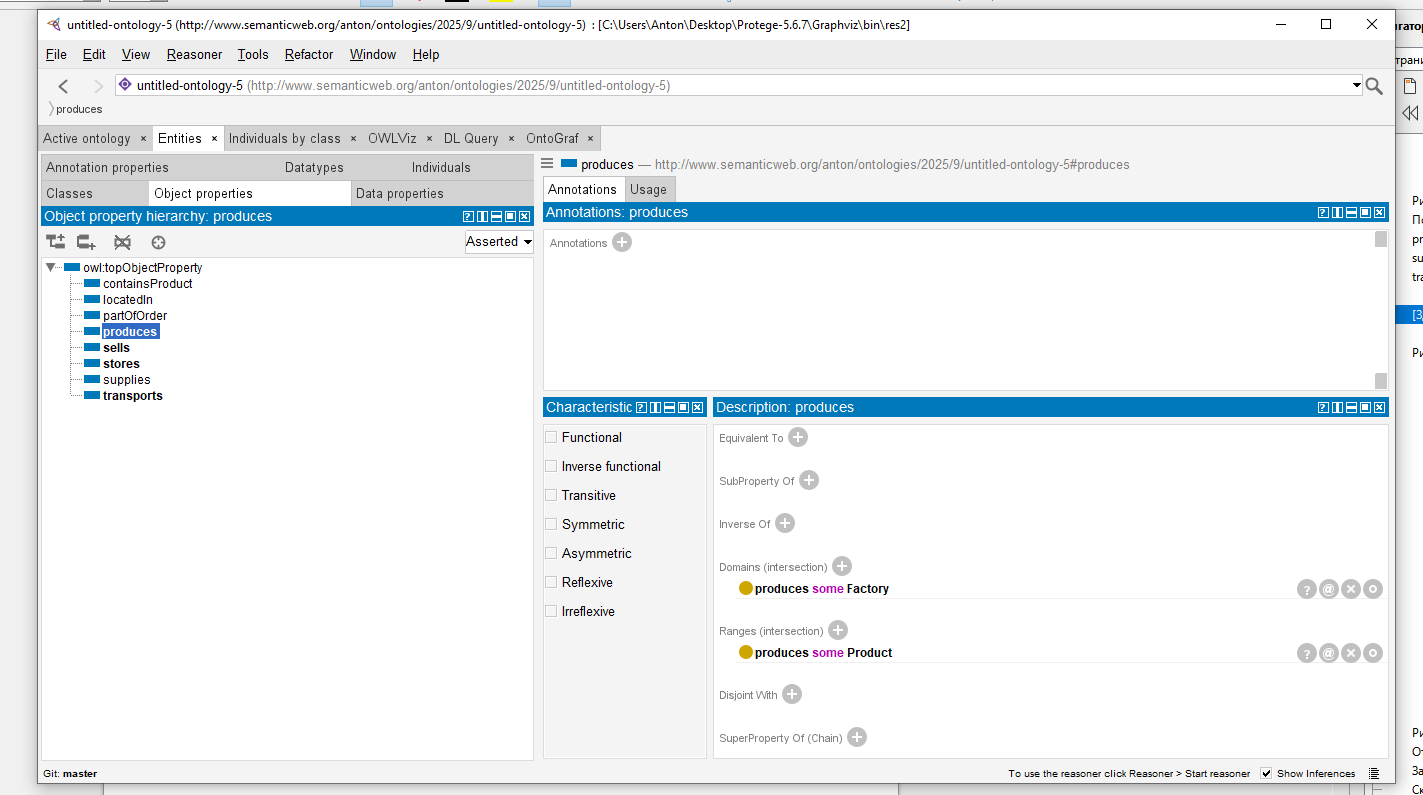


Рисунок 2. Object Properties

**Рисунок 3. Individuals**

Отображает конкретные экземпляры классов (индивиды):

* Заводы: Auto\_Plant\_1, Electronics\_Factory\_2
* Склады: Central\_Warehouse, Regional\_Distribution\_Center
* Магазины: Mail\_Store\_West, Supermarket\_Downtown
* Товары: Laptop\_Pro\_15, Smartphone\_X1
* Транспорт: Car\_Model\_S, Delivery\_Truck\_001, Refrigerated\_Truck\_002

Индивиды связаны между собой через объектные свойства, отражая реальные взаимодействия в логистической цепочке.

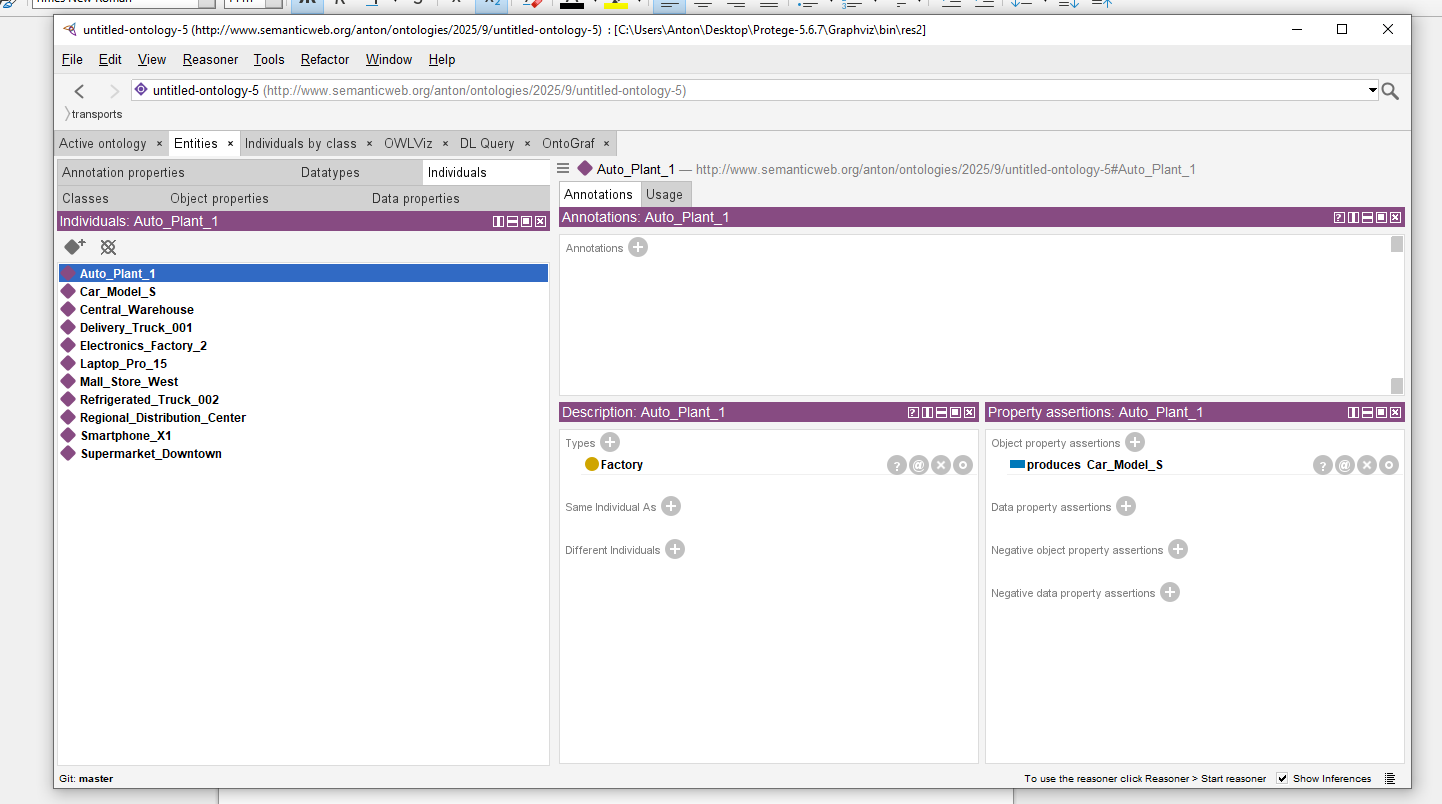


Рисунок 3. Individuals

**4. Реализация в Neo4j**

Для построения графовой модели использовалась Neo4j, позволяющая хранить данные в виде узлов и связей, а также выполнять логические запросы через Cypher.

**Создание узлов и связей (Cypher):**

CREATE

(f1:Factory {name: 'Московский\_Завод', address: 'Москва, пр. Ленина, 1', capacity: 10000}),

(f2:Factory {name: 'Петербургский\_Завод', address: 'СПб, Невский пр., 25', capacity: 8000}),

(w1:Warehouse {name: 'Центральный\_Склад', location: 'Московская область', max\_capacity: 5000}),

(w2:Warehouse {name: 'Региональный\_Склад', location: 'Ленинградская область', max\_capacity: 3000}),

(s1:Store {name: 'Московский\_Магазин\_1', address: 'Москва, ул. Тверская, 10', type: 'retail'}),

(s2:Store {name: 'Онлайн\_Магазин', address: 'online', type: 'online'}),

(p1:Product {name: 'Ноутбук\_Model\_X', price: 999.99, category: 'electronics', weight: 2.5}),

(p2:Product {name: 'Смартфон\_Model\_Y', price: 599.99, category: 'electronics', weight: 0.3}),

(p3:Product {name: 'Планшет\_Model\_Z', price: 399.99, category: 'electronics', weight: 0.7}),

(t1:Transport {id: 'TRUCK\_001', type: 'truck', capacity: 5000, status: 'available'}),

(t2:Transport {id: 'VAN\_002', type: 'van', capacity: 2000, status: 'maintenance'});

CREATE

(f1)-[:PRODUCES {volume: 1000, cost: 450.00}]->(p1),

(f1)-[:PRODUCES {volume: 2000, cost: 250.00}]->(p2),

(f2)-[:PRODUCES {volume: 1500, cost: 180.00}]->(p3),

(w1)-[:STORES {quantity: 500, last\_updated: '2024-01-15'}]->(p1),

(w1)-[:STORES {quantity: 1000, last\_updated: '2024-01-15'}]->(p2),

(w2)-[:STORES {quantity: 300, last\_updated: '2024-01-15'}]->(p3),

(s1)-[:SELLS {price: 999.99, stock: 50}]->(p1),

(s2)-[:SELLS {price: 599.99, stock: 200}]->(p2),

(s2)-[:SELLS {price: 399.99, stock: 150}]->(p3),

(f1)-[:SUPPLIES {frequency: 'weekly', lead\_time: 2}]->(w1),

(f2)-[:SUPPLIES {frequency: 'weekly', lead\_time: 3}]->(w2),

(w1)-[:SUPPLIES {frequency: 'daily', lead\_time: 1}]->(s1),

(w1)-[:SUPPLIES {frequency: 'daily', lead\_time: 0}]->(s2),

(f1)-[:USES\_TRANSPORT]->(t1),

(f2)-[:USES\_TRANSPORT]->(t2),

(t1)-[:DELIVERS\_TO {route: 'Москва-ЦентральныйСклад', distance: 50}]→(w1);

* Factory, Warehouse, Store, Product, Transport: создаются как отдельные узлы с соответствующими свойствами.
* Связи: PRODUCES, STORES, SELLS, SUPPLIES, USES\_TRANSPORT, DELIVERS\_TO отражают реальные отношения между объектами логистической цепочки.
* Модель позволяет выполнять дальнейший анализ и логический вывод через Cypher-запросы.

**Рисунок 5. Демонстрация связей в Neo4j**

* Визуализация графа показывает все узлы и их связи в логистической системе.
* Легко видеть, какие заводы производят какие товары, какие склады их хранят, и в какие магазины они поступают.
* Транспортные связи показывают маршруты доставки между объектами цепочки.
* Граф обеспечивает наглядность и удобство анализа структуры логистической системы.

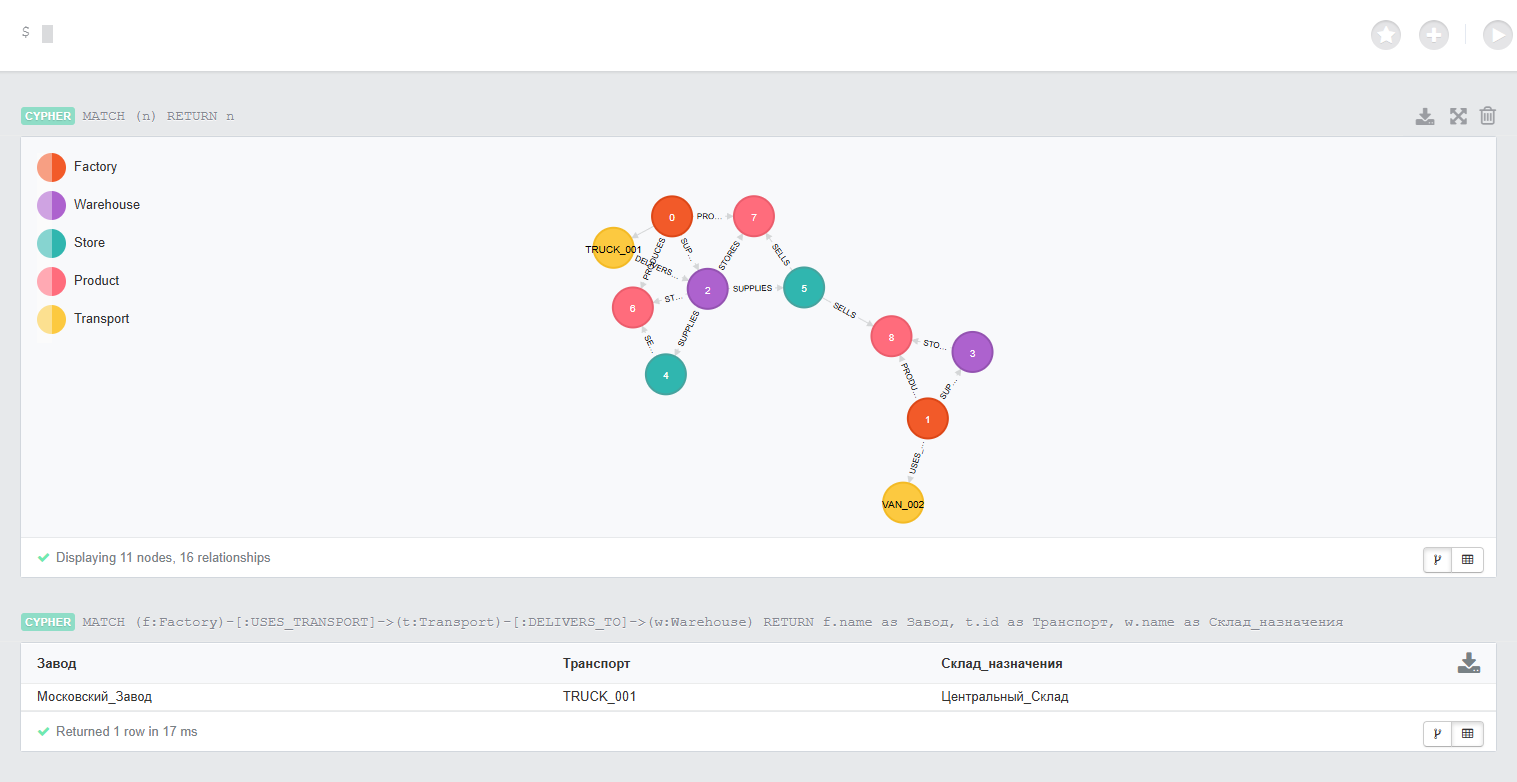


Рисунок 5. Демонстрация связей

**5. Вывод**

В работе создана онтологическая модель логистической цепочки, объединяющая заводы, склады, магазины, товары и транспорт.

Корневой класс Logistic\_Object обеспечивает структуризацию модели, связывая все сущности и упрощая анализ данных логистической системы.

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

* заводы ↔ склады через производимые/хранимые товары,
* магазины ↔ заводы через продаваемые/производимые товары.

Модель наглядно реализована в Protégé и Neo4j, что подтверждается визуализацией и созданными запросами.

Практическая ценность: отслеживание полного цикла движения товаров от производства до реализации, оптимизация логистических маршрутов, анализ взаимодействий между участниками цепочки поставок, а также возможность расширения модели для комплексного управления логистической системой.