Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №3

«Построение онтологической модели»

Вариант №2

Выполнил студент

группы ИВТИИбд-12

Абрамушкин В. Ю.

Преподаватель

Хайрулин И.Д.

Ульяновск, 2025

**1. Структурное описание онтологической модели**

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

**- Фильмы (Film):** Отдельные фильмы

**- Актеры (Actor):** Актеры ролей, играющие в фильме

**- Режиссеры (Director):** Люди снявшие фильм

**- Жанры (Genre):** Жанры кино (фильмов)

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:

**- Фильмы (Film):** Отдельные фильмы

**- Актеры (Actor):** Актеры ролей, играющие в фильме

**- Режиссеры (Director):** Люди снявшие фильм

**- Жанры (Genre):** Жанры кино (фильмов)

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 | Actor, Director | Имя актёра, режиссера |
| hasDuration | Film | Продолжительность фильма |
| hasRoles | Actor | Роли которые играл актёр |
| hasTitle | Film, Genre | Название фильма, жанра |

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** |
| --- | --- |
| hasFilm | Вселенная содержит фильм |
| hasRoles | Актёр имеет роли |
| hasCreator | Снят режиссером |

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

**2.3 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: Film, Actor, Director, Genre.
* Видно, как классы структурированы и какие объекты могут быть связаны между собой.

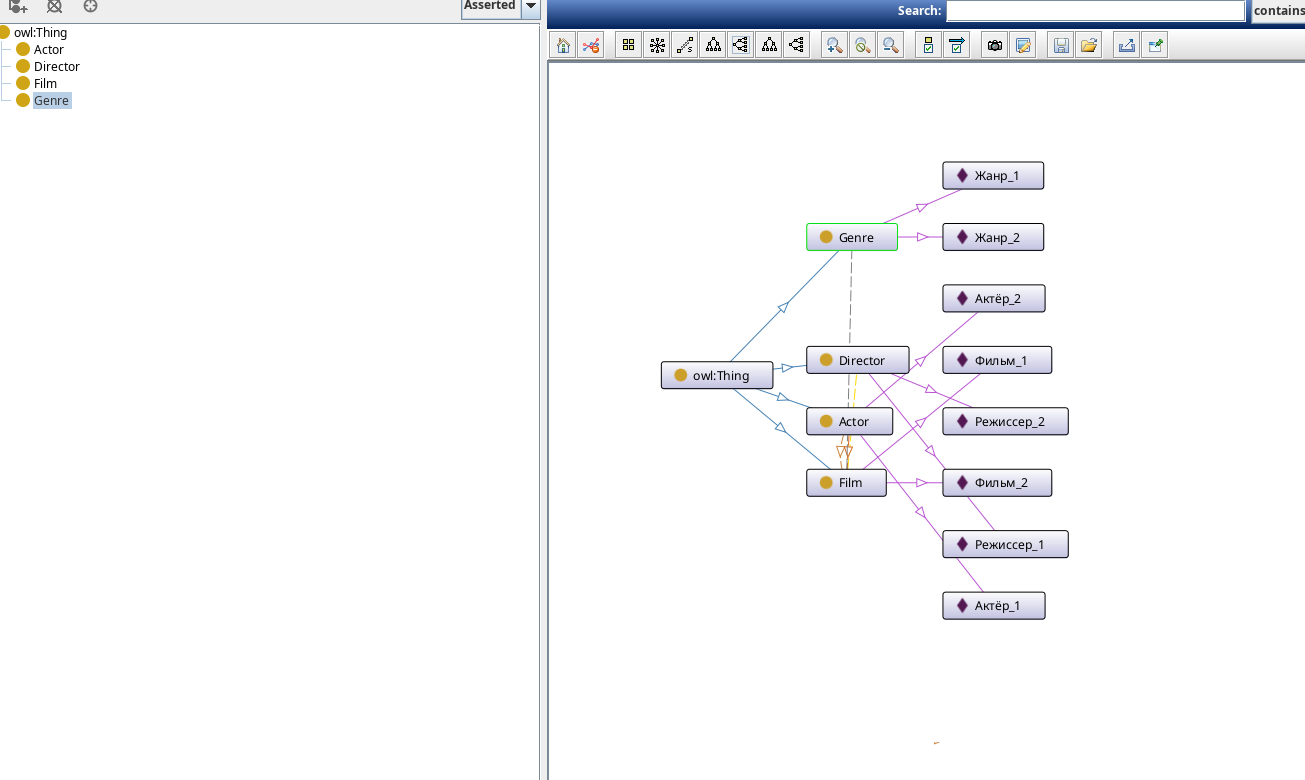


Рисунок 1 – Дерево классов

**Рисунок 2 – Object Properties**

Показывает все свойства объектов:

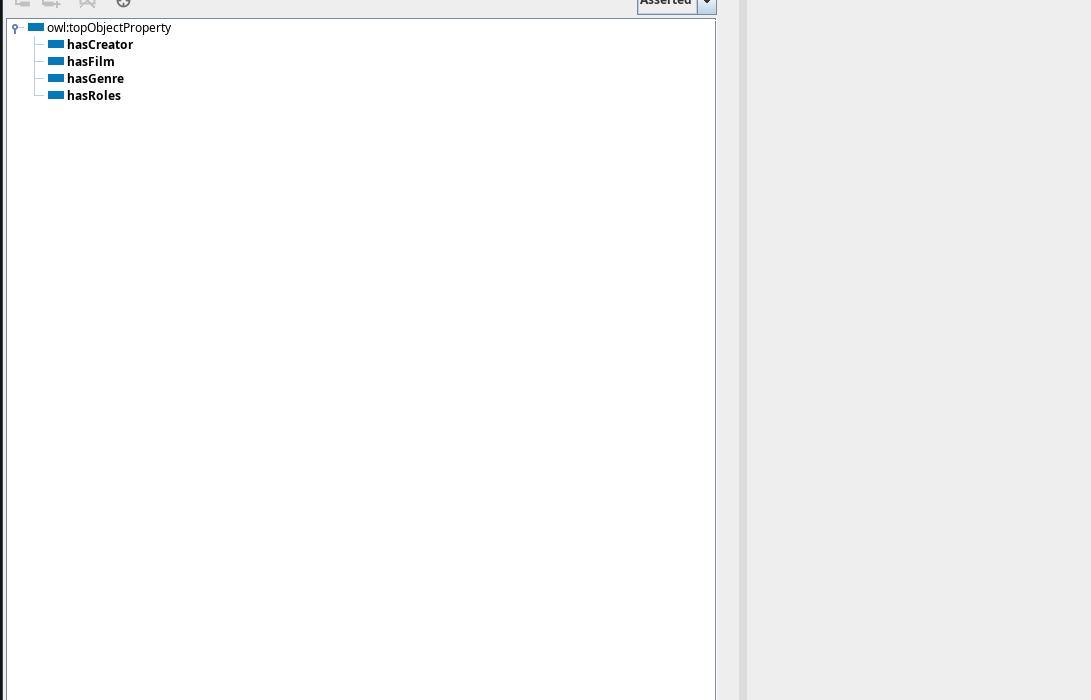


Рисунок 2 – Object Properties

**Рисунок 3 – Individuals**

Отображает конкретные экземпляры классов (индивиды):

* Фильмы: Фильм\_1, Фильм\_2
* Актёры: Актёр\_1, Актёр\_2
* Жанры: Жанр\_1, Жанр\_2
* Режиссёры: Режиссёр\_1, Режиссёр\_2

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

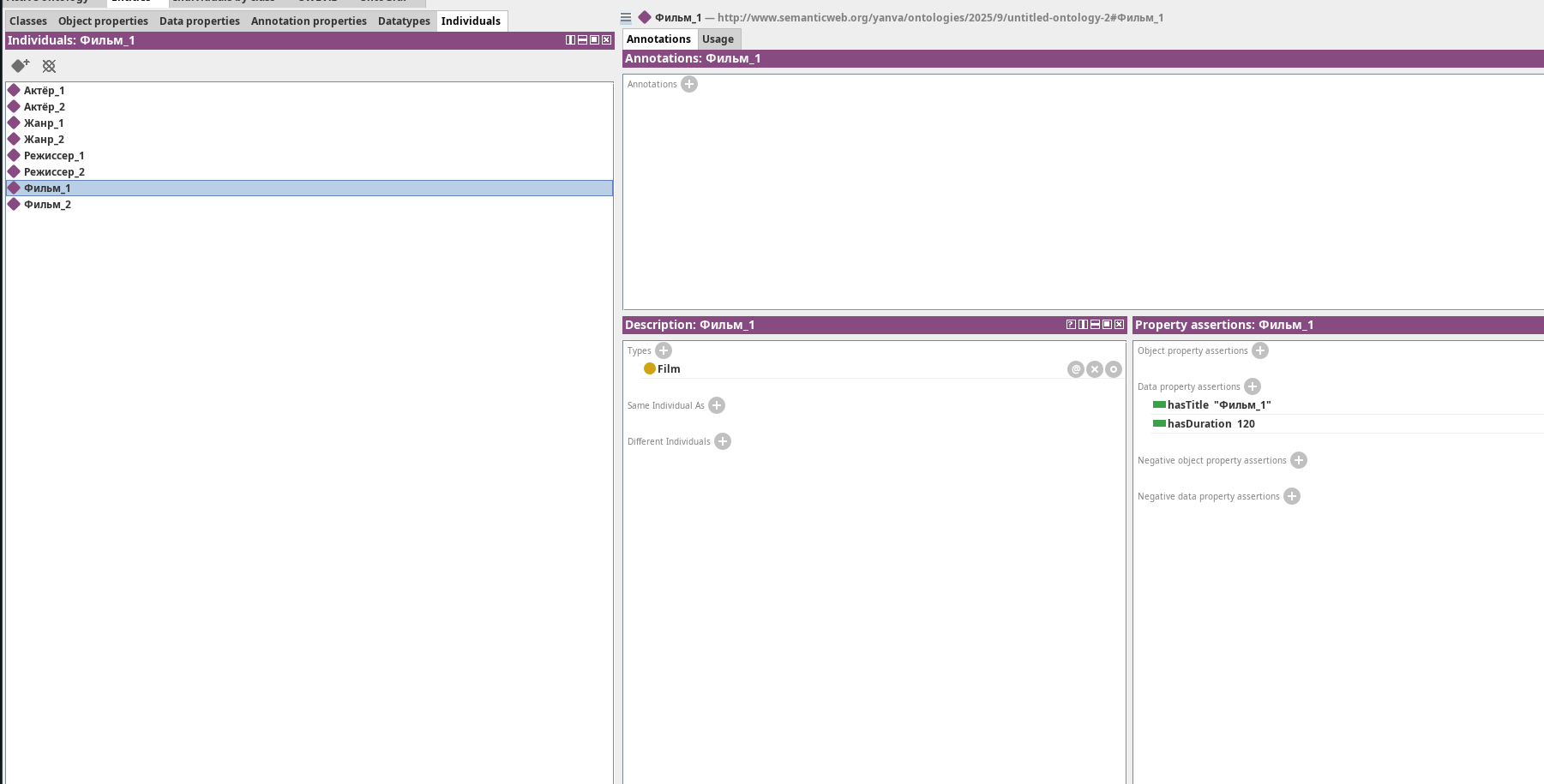


Рисунок 3 – Individuals

# 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:Film {hasTitle: "Фильм 1", hasDuration: 120}),

(f2:Film {hasTitle: "Фильм 2", hasDuration: 90}),

(act1:Actor {hasName: "Актер 1", hasRoles: ["Роль А", "Роль Б"]}),

(act2:Actor {hasName: "Актер 2", hasRoles: ["Роль В"]}),

(d1:Director {hasName: "Режиссер 1"}),

(d2:Director {hasName: "Режиссер 2"}),

(g1:Genre {hasTitle: "Жанр 1"}),

(g2:Genre {hasTitle: "Жанр 2"})

* MATCH (f:Film {hasTitle: "Фильм 1"}), (d:Director {hasName: "Режиссер 1"})

CREATE (f)-[:hasCreator]->(d)

MATCH (f:Film {hasTitle: "Фильм 2"}), (d:Director {hasName: "Режиссер 2"})

CREATE (f)-[:hasCreator]->(d)

MATCH (a:Actor {hasName: "Актер 1"}), (f:Film {hasTitle: "Фильм 1"})

CREATE (a)-[:hasFilm]->(f)

MATCH (a:Actor {hasName: "Актер 2"}), (f:Film {hasTitle: "Фильм 2"})

CREATE (a)-[:hasFilm]->(f)

MATCH (a:Actor {hasName: "Актер 1"}), (f:Film {hasTitle: "Фильм 2"})

CREATE (a)-[:hasFilm]->(f)

MATCH (f:Film {hasTitle: "Фильм 1"}), (g:Genre {hasTitle: "Жанр 1"})

CREATE (f)-[:hasGenre]->(g)

MATCH (f:Film {hasTitle: "Фильм 2"}), (g:Genre {hasTitle: "Жанр 2"})

CREATE (f)-[:hasGenre]->(g)

MATCH (f:Film {hasTitle: "Фильм 1"}), (g:Genre {hasTitle: "Жанр 2"})

CREATE (f)-[:hasGenre]->(g)

**Рисунок 5 – Демонстрация связей**

* Визуализация графа показывает все узлы и их связи.
* Легко видеть, какие актеры играли в фильму, а кто автор фильма.

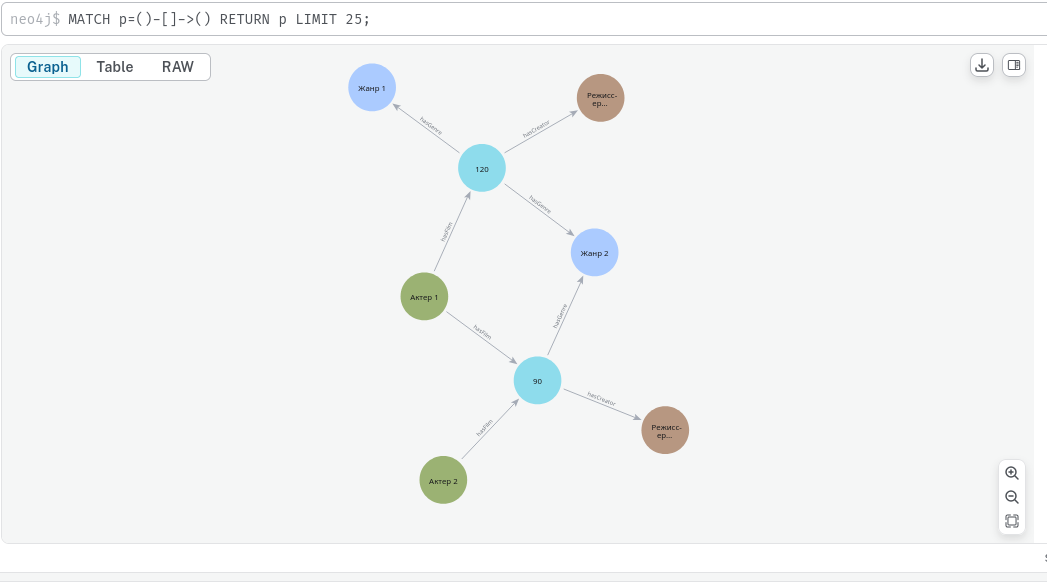


Рисунок 4 – Демонстрация связей

**5. Вывод**

В работе создана онтологическая модель музыкального сервиса, объединяющая **фильмы, актёров, режиссеров и жанры**.

**Cypher-запросы** позволяют автоматически выводить обратные и косвенные связи.

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

Практическая ценность: отслеживание взаимодействий между фильмами, актёрами, режиссёрами и жанрами, а также возможность расширения модели и анализа данных для киновселенной.