Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №3

«Построение онтологической модели»

Вариант №4

Выполнил студент

группы ИВТИИбд-11

Богданов Р.М.

Преподаватель

Хайрулин И.Д.

Ульяновск, 2025

**1. Структурное описание онтологической модели**

Онтологическая модель описывает музыкальный сервис и формализует ключевые объекты и их взаимосвязи. Основные сущности или классы модели включают:

* **Исполнители (Artist):** объекты, тех или иных музыкальных исполнителей в первую очередь групп.
* **Альбомы (Album):** созданная исполнителем коллекция треков, часто завязанная на общей тематике.
* **Треки (Track):** музыкальные произведения, созданные исполнителями, попадающие в плейлисты, и часто являющиеся частью альбомов тех или иных исполнителей.
* **Плейлисты (Playlist):**. коллекция музыкальных произведений часто имеющая общий жанр или настроение.

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:

* **Исполнители (Artist):** представляет литературные произведения, которые могут быть написаны авторами и опубликованы издательствами.
* **Альбомы (Album):** лица, создающие книги и участвующие в процессе публикации.
* **Треки (Track):** пользователи библиотеки, которые берут книги на время и взаимодействуют с другими сущностями через взятые книги.
* **Плейлисты (Playlist):** организации, выпускающие книги, связанные с авторами и читателями через свои публикации.

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** |
| --- | --- | --- |
| name | Artist | Имя исполнителя |
| title | Album | Название Альбома |
| title | Track | Название трека |
| name | Playlist | Название плейлиста |

**2.3 Свойства объектов (Object Properties)**

Свойства объектов определяют связи между различными объектами модели:

| **Свойство** | **Описание** |
| --- | --- |
| genre | Жанр альбома или трека |
| mood | Настроение плейлиста |
| release date | Время выхода альбома |
| duration | Продолжительность трека |
| main artist | Кем создан альбом |
| contain | Содержит альбом или является частью альбома |
| performs in | Входит в плейст или является его частью |

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. Если год больше 1992 создано КиШ**:**

untitled-ontology-4:Альбом(?t) ^ untitled-ontology-4:Год\_выпуска(?t, ?d) ^ swrlb:lessThan(?d, 1992) -> untitled-ontology-4:создает(untitled-ontology-4:КиШ, ?t)

Если год создания альбома больше 1992 то исполнителем является КиШ.

1. КиШ Русский рок:

untitled-ontology-4:Трек(?x) ^ untitled-ontology-4:создает(?x, untitled-ontology-4:КиШ) -> untitled-ontology-4:входит\_в\_альбом(untitled-ontology-4:Русский\_рок, ?x)

Если трек создан КиШ то жанр альбома русский рок.

# 2.5 Аналог аксиом для Neo4j (Cypher-запросы)

Для графовой модели логические выводы реализуются с помощью запросов:

1. Выводит если плейлист содержит больше 2 треков с жанром русский рок то его тематика русский рок**:**

MATCH (p:Playlist)-[:INCLUDES]->(t:Track)

WHERE t.genre = "Русский рок"

WITH p, COUNT(t) AS rus\_rock\_tracks

WHERE rus\_rock\_tracks >= 2

RETURN p.name AS thematic\_playlist, rus\_rock\_tracks AS matching\_tracks;

1. Выводит альбомы где число треков с продолжительностью меньше 4 минут :

MATCH (alb:Album)-[:CONTAINS]->(t:Track)

WHERE t.duration < 240

WITH alb, COUNT(t) AS short\_tracks

WHERE short\_tracks >= 2

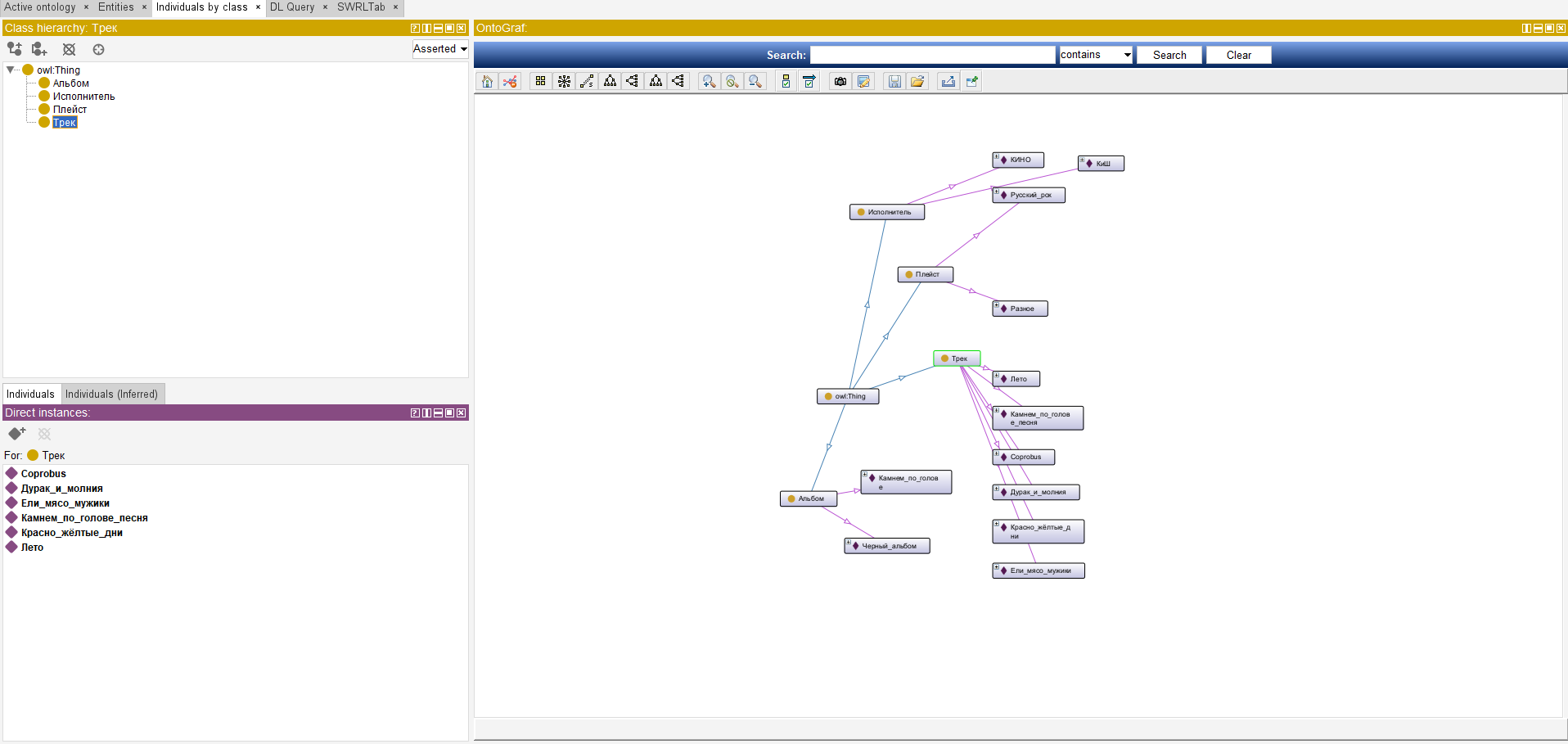
RETURN alb.title AS album, short\_tracks AS tracks\_under\_4min;

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.

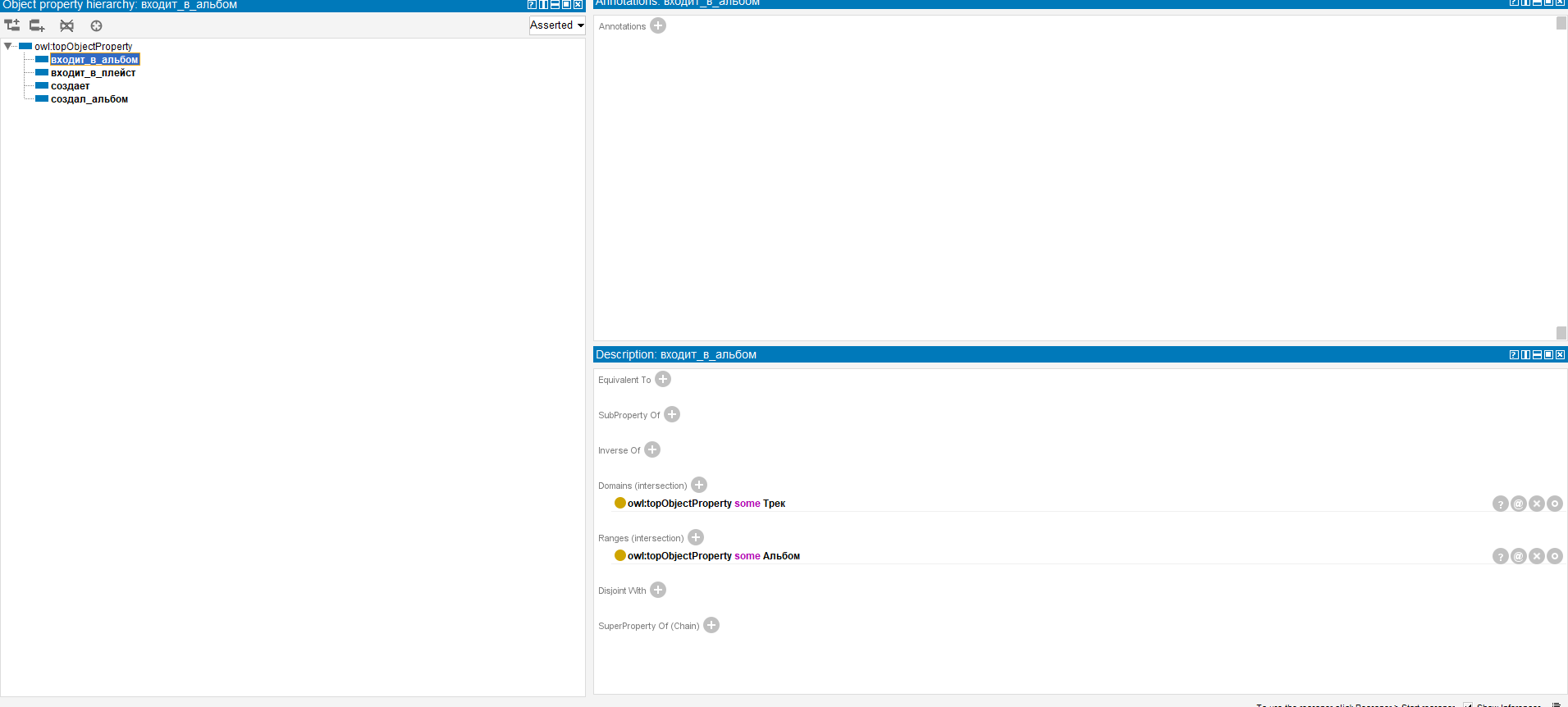
**Рисунок 1 – Дерево классов**

* Отображает иерархию всех классов модели: Исполнитель, Альбом, Трек, Плейст.
* Видно, как классы структурированы и какие объекты могут быть связаны между собой.



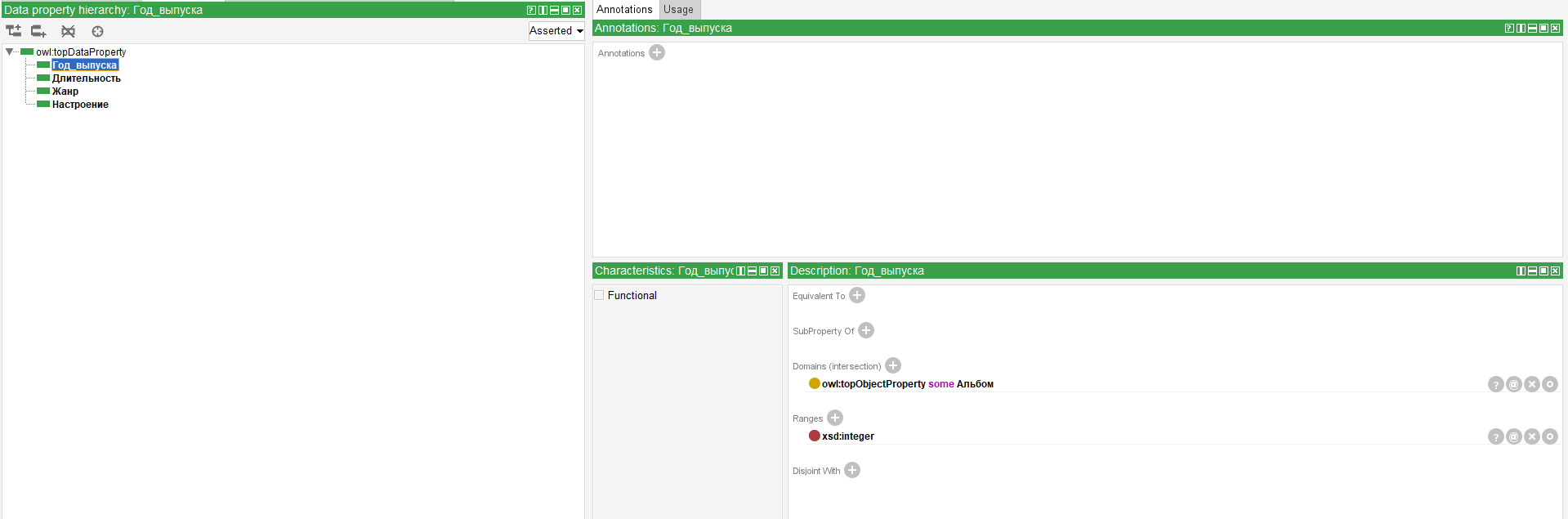
**Рисунок 2 – Object Properties**

Показывает все связи объектов и их направление, включая:



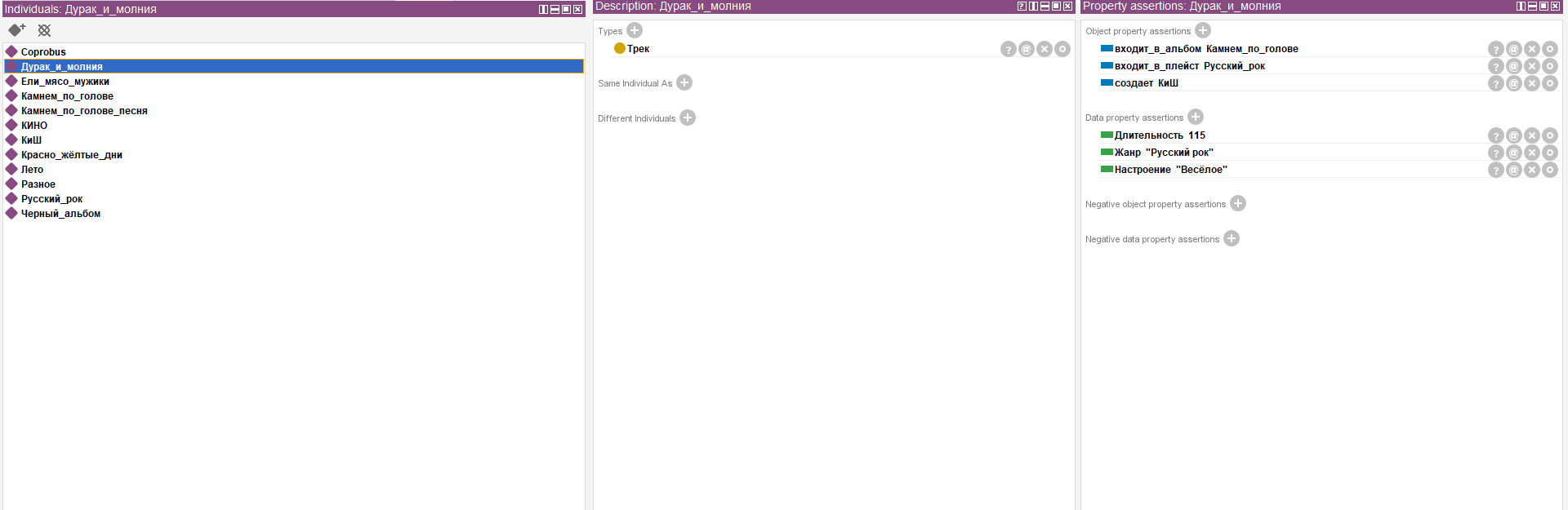
**Рисунок 3 – Data properties**

Свойства объектов:

****

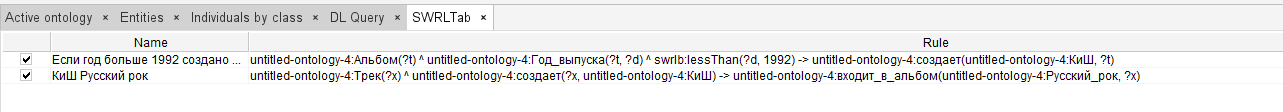
**Рисунок 4 – Individuals**

Отображает конкретные экземпляры классов (индивиды):



**Рисунок 5 – SWRLTab с правилами**

Показывает правила логического вывода



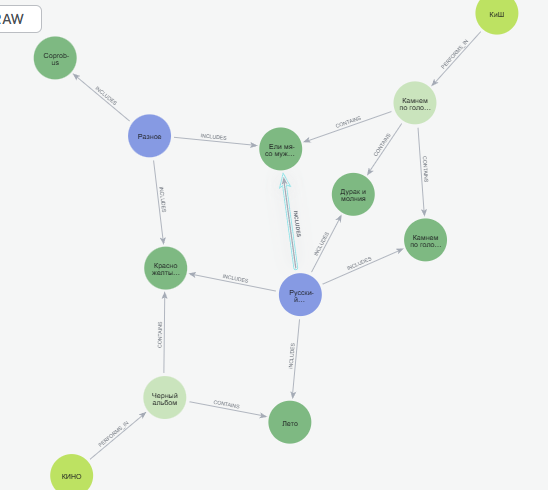
# 5. Реализация в Neo4j

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

* MATCH (n) WHERE n.title IS NOT NULL RETURN DISTINCT "node" as entity, n.title AS title LIMIT 25 UNION ALL MATCH ()-[r]-() WHERE r.title IS NOT NULL RETURN DISTINCT "relationship" AS entity, r.ti...
* MATCH p=()-[]->() RETURN p LIMIT 25;
* MATCH (n) RETURN n LIMIT 25;
* MATCH (n:Track) RETURN n LIMIT 25;
* MATCH (alb:Album)-[:CONTAINS]->(t:Track) WHERE t.duration < 240 WITH alb, COUNT(t) AS short\_tracks WHERE short\_tracks >= 2 RETURN alb.title AS album, short\_tracks AS tracks\_under\_4min;
* MATCH (p:Playlist)-[:INCLUDES]->(t:Track) WHERE t.genre = "Русский рок" WITH p, COUNT(t) AS rus\_rock\_tracks WHERE rus\_rock\_tracks >= 2 RETURN p.name AS thematic\_playlist, rus\_rock\_tracks AS matchin...
* MATCH p=()-[:INCLUDES]->() RETURN p LIMIT 25;
* MATCH p=()-[:CONTAINS]->() RETURN p LIMIT 25;
* MATCH (p:Playlist {name: "Русский рок"}), (t1:Track {title: "Ели мясо мужики"}), (t2:Track {title: "Камнем по голове"}), (t3:Track {title: "Красно желтые дни"}), (t4:Track {title: "Лето"}), (t5:Tra...
* MATCH (p:Playlist {name: "Разное"}), (t1:Track {title: "Coprobus"}), (t2:Track {title: "Красно желтые дни"}), (t3:Track {title: "Ели мясо мужики"}) CREATE (p)-[:INCLUDES {added\_date: date('2026-10-...
* MATCH (p:Playlist {name: "Разное"}), (t1:Track {title: "Coprobus"}), (t2:Track {title: "Красно желтые дни"}), (t3:Track {title: "Ели мясо мужики"}) CREATE (p)-[:INCLUDES {added\_date: date('2026-10-...
* MATCH (n:Artist) RETURN n LIMIT 25;
* MATCH (n:Playlist) RETURN n LIMIT 25;
* MATCH (alb:Album {title: "Черный альбом"}), (t1:Track {title: "Лето"}), (t2:Track {title: "Красно желтые дни"}) CREATE (alb)-[:CONTAINS {track\_number: 1}]->(t1), (alb)-[:CONTAINS {track\_number: 2}]...
* MATCH (n:Album) RETURN n LIMIT 25;
* MATCH (alb:Album {title: "КИНО"}), (t1:Track {title: "Лето"}), (t2:Track {title: "Красно желтые дни"}) CREATE (alb)-[:CONTAINS {track\_number: 1}]->(t1), (alb)-[:CONTAINS {track\_number: 2}]->(t2)
* MATCH (alb:Album {title: "Камнем по голове"}), (t1:Track {title: "Камнем по голове"}), (t2:Track {title: "Дурак и молния"}), (t3:Track {title: "Ели мясо мужики"}) CREATE (alb)-[:CONTAINS {track\_num...
* MATCH ()-[r:INCLUDES]-() DELETE r
* MATCH p=()-[:PERFORMS\_IN]->() RETURN p LIMIT 25;
* MATCH (alb:Album {title: "Камнем по голове"}), (t:Track {title: "Ели мясо мужики"}) CREATE (alb)-[:INCLUDES {position: 3}]->(t)
* MATCH (alb:Album {title: "Камнем по голове"}), (t:Track {title: "Дурак и молния"}) CREATE (alb)-[:INCLUDES {position: 2}]->(t)
* MATCH (alb:Album {title: "Камнем по голове"}), (t:Track {title: "Камнем по голове"}) CREATE (alb)-[:INCLUDES {position: 1}]->(t)
* MATCH (a:Artist {name: "КиШ"}), (alb:Album {title: "Камнем по голове"}) CREATE (a)-[:PERFORMS\_IN {role: "main\_artist"}]->(alb)
* MATCH (a:Artist {name: "КИНО"}), (alb:Album {title: "Черный альбом"}) CREATE (a)-[:PERFORMS\_IN {role: "main\_artist"}]->(alb)
* MATCH (n) WHERE n.visualisation IS NOT NULL RETURN DISTINCT "node" as entity, n.visualisation AS visualisation LIMIT 25 UNION ALL MATCH ()-[r]-() WHERE r.visualisation IS NOT NULL RETURN DISTINC...
* CREATE (p1:Playlist {name: "Разное", mood: "странное"}), (p2:Playlist {name: "Русский рок", mood: "Заряженное"})
* CREATE (t1:Track {title: "Лето", duration: 354, genre: "Русский рок", mood: "Грустное"}), (t2:Track {title: "Красно желтые дни", duration: 350, genre: "Русский рок", mood: "Ностальгическое"}...
* CREATE (alb1:Album {title: "Камнем по голове", release\_date: date('1996-03-24'), genre: "Русский рок"}), (alb2:Album {title: "Черный альбом", release\_date: date('1991-01-12'), genre: "Русски...
* CREATE (a1:Artist {name: "КИНО", genre: "Русский рок"}), (a2:Artist {name: "КиШ", genre: "Русский рок"})

**Рисунок 5 – Демонстрация связей**

Визуализация графа показывает все узлы и их связи.



**5. Вывод**

В работе создана онтологическая модель музыкального сервиса, объединяющая **исполнителей, альбомы, треки и плейлисты**.

**SWRL-правила** и **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 и взаимосвязей со свойствами.