Вінницький національний технічний університет Факультет
інтелектуальних інформаційних технологій та автоматики
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА ОНТОЛОГІЧНОЇ БАЗИ ЗНАНЬ. «МУЗИКА ТА КІНО» ЧАСТИНА 1. МУЗИЧНА ОНТОЛОГІЯ

Виконав: студент 4 курсу, групи 1КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Ковальський А. М.
(прізвище та ініціали)

Керівник: к.ф.-м.н., доцент каф. КН
Сілагін О. В.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: к.т.н., доцент каф. АІІТ
Кривогубченко С.Г
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту Зав.
кафедри КН Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський) Галузь знань – 12 Інформаційні технології Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

д.т.н., проф. Яровий А. А.

«05» Грудня 2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Ковальський Артур Миколайович

1. Тема роботи: Розробка онтологічної бази знань. «Музика та кіно» Частина 1. Музична онтологія.
   Керівник роботи: Сілагін Олексій Віталійович, к.ф.-м.н., доц.
   Затверджені наказом вищого навчального закладу «20» Березня 2025 року № 97
2. Термін подання студентом роботи 30 Грудня 2025 року
3. Вихідні дані до роботи: Об'єктно-орієнтована парадигма програмування; об'єктно-орієнтовані бази знань; моделі представлення знань – семантичні та фреймові мережі; передбачити можливість застосування запитів до бази знань; кількість основних класів – не менше 5 одиниць; кількість підкласів – не менше 10 одиниць; кількість індивідів – не менше 20 одиниць; ступінь ієрархії – не менше 2 гілки.
4. Зміст текстової частини: Вступ, обґрунтування доцільності розробки інформаційної технології онтологічного моделювання бази знань «Музична онтологія», розробка інформаційної технології онтологічного моделювання бази знань «Музична онтологія», програма реалізації інформаційної технології та онтологічне моделювання бази знань «Музична онтологія», економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):Удосконалена інформаційна технологія створення онтологічної моделі; Діаграма класів предметної області «Музична онтологія»; Діаграма предикатів предметної області «Музична онтологія»; Граф онтології предметної області «Музична онтологія»; Граф ієрархії класів онтології «Музична онтологія»; Граф індивідів онтології.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Сілагін О.В., к.ф.-м.н, доцент каф. КН		

7. Дата видачі завдання 05 травня 2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання		Примітки
		початок	закінчення	
1	Аналіз предметної галузі. Огляд чинних систем аналогів. Постановка задач дослідження	05.05.25	07.05.25	
2	Моделювання та проєктування онтологічної бази знань «Музична онтологія»)	11.05.25	14.05.25	
3	Реалізація онтологічної бази знань «Музична онтологія»	17.05.25	22.05.25	
4	Тестування онтологічної бази знань «Музична онтологія»	23.05.25	27.05.25	
5	Оформлення матеріалів до захисту БКР	30.05.25	04.06.25	

Студент _____ Ковальский А.М.

(підпис)

Керівник роботи _____ Сілагін О.В.

(підпис)

АНОТАЦІЯ

Бакалаврська дипломна робота зосереджена на розробці онтологічної бази знань стосовно теми «Музична онтологія», зосереджуючись на формуванні музичної онтології. Ключовою задачею дослідження є формалізація та організація знань про музику як комплексну культурну, художню та технічну сферу, а також взаємозв'язків між музичними творами, жанрами, виконавцями, альбомами, і саундтреками для кіно та іншими елементами кіномистецтва та музичної індустрії.

В ході роботи було проведено аналіз предметної області, виявлено ключові поняття та встановлено їх взаємозв'язки. Для реалізації бази знань використано онтологічний підхід з застосуванням мови OWL (Web Ontology Language) та програмного середовища Protégé. Створена онтологія містить в собі базові класи (Гурт/Композитор, Жанр, Альбом, Назва музичної композиції, Нагорода), їхні властивості та відношення між ними, що дає змогу забезпечити повноцінну семантичну модель інформації про музику.

Ключові слова: онтологія, база знань, музика, кіно, альбом, жанр, гурт\композитор, OWL, Protégé.

ABSTRACT

Bachelor's Thesis is focused on the development of an ontological knowledge base on the topic “Music onthology”, with a primary emphasis on the formation of a musical ontology. The key objective of the study is the formalization and organization of knowledge about music as a complex cultural, artistic, and technical domain, as well as the interconnections between musical works, genres, performers, albums, soundtracks for films, and other elements of cinematography and the music industry.

During the course of the work, an analysis of the subject area was carried out, key concepts were identified, and their relationships were established. An ontological approach was applied for the implementation of the knowledge base, using the OWL (Web Ontology Language) and the Protégé software environment. The resulting ontology includes fundamental classes (Band/Composer, Genre, Album, Music Track Title, Award), their properties, and relationships, which enables the construction of a comprehensive semantic model of music-related information.

Keywords: ontology, knowledge base, music, cinema, album, genre, band/composer, OWL, Protégé.

ЗМІСТ

ВСТУП

Актуальність теми: У сучасному світі обсяги інформації у сфері культури, зокрема музики, стрімко зростають. Щодня публікуються нові композиції, альбоми, з’являються нові виконавці, експериментальні жанри, музичні платформи, фестивалі, нагороди тощо. Усе це породжує складне інформаційне середовище, де ручне опрацювання даних стає все менш ефективним.

Традиційні способи збереження й пошуку музичної інформації — енциклопедії, бази даних, медіаплатформи — здебільшого не забезпечують семантичного представлення знань і не виявляють глибинних зв’язків між елементами (наприклад, між жанром, інструментами, епохою або виконавцем). У зв’язку з цим онтологічне моделювання постає як сучасний і перспективний підхід для організації, формалізації та інтелектуальної обробки музичних знань.

Онтологічна база знань дозволяє не тільки зберігати інформацію, але й формувати логічні зв’язки між поняттями, автоматично виводити нові знання, будувати запити для інтелектуального пошуку. Особливо це актуально для музичної сфери, що є багатовимірною — вона включає жанри, епохи, стильові школи, культурні контексти, виконавців, інструменти, твори, технічні параметри аудіозапису, зв’язки з подіями та іншими видами мистецтва.

Таким чином, створення онтології, що фокусується саме на музичній сфері, є важливим кроком у напрямі формалізації культурних знань, розвитку цифрової гуманітаристики та підтримки освітніх, дослідницьких, функіональних і прикладних систем, орієнтованих на музику.

Мета і завдання дослідження: розширення функціональних можливостей досягнення в порівняні з базами даних, структура когнитивних властивостей і атрибутів .

Об’єктом дослідження є процес структурування знань у сфері музики.

Предметом дослідження є програмні засоби із розробки баз знань.

Задачі дослідження:

1. Аналіз предметної галузі. Огляд чинних систем аналогів. Постановка задач дослідження

2. Моделювання та проєктування онтологічної бази знань «Музична онтологія»)
3. Реалізація онтологічної бази знань «Музична онтологія»
4. Тестування онтологічної бази знань «Музична онтологія»
5. Оформлення пояснювальної записки, графічного матеріалу та презентації

1 ОБГРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ ОНТОЛОГІЧНОЇ БАЗИ ЗНАНЬ «МУЗИЧНА ОНТОЛОГІЯ»

1.1 Передумови створення онтологічної бази знань «Музична онтологія»

Сучасний інформаційний простір відзначається величезною кількістю даних, що стосуються різних аспектів музичного домену, таких як жанри, художники, музичні інструменти та еволюція музики з часом. Тим не менш, успішне використання цих даних потребує створення добре організованої системи, яка дозволить їх організації, аналіз та застосування в наукових дослідженнях, навчальних починаннях, музичній індустрії та технологічному прогресі. Один зі способів розв’язання цього питання - це створення онтологічної основи знань (OBS) [1], який пропонує структуроване представлення знань у музичній області.

Протягом останніх десятиліть музична індустрія зазнала кардинальних змін завдяки цифровій революції. Поява стрімінгових платформ, таких як Spotify, Apple Music, та YouTube Music, створили нові виклики у сфері організації та категоризації музичного контенту. Щоденно на цих платформах з'являються тисячі нових композицій, що створює потребу в автоматизованих системах класифікації та індексації.

Паралельно з цим розвивається культура ремікшування та семплування, де нові твори створюються на основі чинних композицій. Це створює складні мережі взаємозв'язків між музичними творами, які важко відстежити без спеціалізованих інструментів. База знань може забезпечити систематичне відображення цих зв'язків, створюючи можливості для дослідження еволюції музичних ідей та впливів.

Сучасне музикознавство є міждисциплінарною областю, що поєднує елементи історії, соціології, психології, акустики, математики та інформатики. Цей міждисциплінарний підхід вимагає інтеграції знань з різних наукових сфер, що ускладнює процес систематизації інформації.

Психомузичні дослідження показують, як музика впливає на когнітивні процеси та емоційний стан людини. Соціомузикологія вивчає роль музики в формуванні культурної ідентичності та соціальних рухів. Акустичні дослідження фокусуються на фізичних властивостях звуку та їх сприйнятті. Кожна з цих областей генерує

специфічні типи даних, які потребують уніфікованого підходу до збереження та аналізу.

Глобалізація призвела до безпрецедентного змішування музичних традицій з різних культур. Традиційні системи класифікації, які базувалися на західноєвропейських музичних концепціях, виявляються недостатніми для опису багатства світової музичної спадщини. Необхідність створення інклюзивної системи, яка враховує різноманітні музичні традиції, стає актуальнішою.

Особливу увагу слід приділити збереженню музичної спадщини корінних народів та етнічних меншин. Багато традиційних музичних форм знаходяться під загрозою зникнення через процеси урбанізації та культурної асиміляції. База знань може стати важливим інструментом для документування та збереження цієї спадщини для майбутніх поколінь.

У царині штучного інтелекту та семантичних технологій онтологія відноситься до структурованого представлення знань, що включає поняття, їх властивості та зв'язки між ними. Онтологічні моделі використовуються для організації інформації, що дозволяє створити інтелектуальні системи, які можуть аналізувати, знаходити та обробляти дані [1].

У музичній індустрії онтологію можна розділити на кілька основних категорій, таких як:

- Музичні жанри (рок, класична музика, джаз, електронна музика тощо)
- Виконавці та композитори (біографічні дані, творчі періоди, вплив)
- Музичні інструменти (категоризація, атрибути, історія еволюції)
- Музичні композиції (форма, тональність, ритм, лексичні елементи)
- Вивчення історії музики охоплює різні аспекти, включаючи періодизацію, культурний контекст та еволюцію різних жанрів

Сучасні технології надають безпрецедентні можливості для аналізу музичного контенту. Алгоритми машинного навчання можуть автоматично розпізнавати музичні жанри, виділяти характерні риси композицій та навіть генерувати нову музику. Проте, ці технології потребують якісних, структурованих даних для ефективного функціонування.

Розвиток технологій розпізнавання аудіо дозволяє автоматично екстрагувати метадані з музичних файлів, включаючи темп, тональність, гармонічну структуру та

тембральні характеристики. Ці технічні параметри можуть бути інтегровані в базу знань для створення детальнішого та точного опису музичних творів [2].

Водночас існують значні обмеження у розумінні семантичного змісту музики машинними алгоритмами. Емоційне забарвлення, культурний контекст та художня цінність композицій залишаються важко формалізованими аспектами, що вимагає поєднання автоматизованих методів з експертними знаннями музикознавців.

Основна потреба для створення онтологічної бази знань «Музична онтологія»

Різноманітність та зміна музичного домену. Музична індустрія - одна з найбільш швидко розвиваючись сфер мистецтва. Постійний розвиток нових жанрів, технологій звукозапису, методів виконання та експериментальних підходів потребує створення бази знань, яка може організовувати інформацію та зберегти історичні дані для майбутніх досліджень.

Вимога до поєднання музичного розуміння. Значна кількість музичних ресурсів ділиться між науковими публікаціями, цифровими бібліотеками, аудіо збором та інформаційними платформами. Відсутність уніфікованої системи робить складним аналіз даних та ефективне використання їх. Фонд знань дозволяє об'єднати ці джерела у всеосяжну та взаємопов'язану систему.

Стандартизація музичних метаданих. Одним з ключових викликів є відсутність уніфікованих стандартів для опису музичного контенту. Різні платформи та організації використовують власні схеми метаданих, що ускладнює інтеграцію та порівняння інформації. База знань може стати основою для розробки стандартизованих підходів до опису музичних творів та пов'язаних з ними даних.

Створення технологічного прогресу. З прогресом в обробці інформації, таких як штучний інтелект, машинне навчання, семантичні векторні технології та великі дані, зараз існують можливості для автоматизованого аналізу музичних композицій, розпізнавання стилістичних елементів та прогнозування майбутніх тенденцій у музичній індустрії.

Автоматизований аналіз та обробка музичних даних. Використовуючи онтологічний підхід, розробляються експертні системи для надання допомоги користувачам у пошуку відповідної музичної інформації, встановлення зв'язків між музичними стилями та виконавцями та генеруванням персоналізованих рекомендацій

шляхом аналізу історичних даних та індивідуальних уподобань.

Забезпечення якості та достовірності інформації. В епоху інформаційного перевантаження особливого значення набуває питання якості та достовірності даних. База знань може включати механізми верифікації інформації, системи рейтингів експертів та процедури рецензування, що забезпечить високу якість музикознавчої інформації [2].

Просування навчальних та дослідницьких ініціатив. Структуровані бази знань є дуже корисними для навчальних програм з музикознавства, оскільки вони допомагають в аналізі композицій та вивченні історії музики. Завдяки онтологічній моделі студенти та дослідники можуть легко отримати інформацію про музичні композиції, вивчити їх характеристики та проводити порівняльний аналіз.

Розвиток інтерактивних навчальних інструментів. База знань може стати основою для створення інтерактивних освітніх платформ, які дозволять студентам досліджувати музичні зв'язки, прослухувати приклади та отримувати персоналізовані навчальні матеріали відповідно до їх рівня підготовки та інтересів.

1.2 Аналіз відомих реалізацій бази знань в предметній області «Музична онтологія»

Протягом останніх десятиліть було створено чимало онтологічних баз знань та семантичних моделей, що мають на меті структурувати інформацію у сфері музики. Кожна з цих розробок вирізняється певними рисами, перевагами та недоліками, які необхідно проаналізувати для розуміння сучасного стану даної предметної області.

Music Ontology постає однією з перших значних спроб формалізації знань у музичній царині, розробленою у 2006 році в рамках проєкту OMRAS2 (Online Music Recognition and Searching). Ця онтологія послуговується технологіями Semantic Web для опису музичних подій, творів, записів та взаємозв'язків між ними. Унікальністю Music Ontology є використання стандартів RDF (Resource Description Framework) та OWL (Web Ontology Language), охоплення різноманітних аспектів музики: створення, аранжування, виконання, запис, а також інтеграція з іншими онтологіями, зокрема Timeline Ontology та Event Ontology.[3] Однак, система має певні обмеження, серед

яких: недостатньо глибока структуризація музично-теоретичних концепцій, відносно слабка підтримка нетрадиційних музичних жанрів і культур, складність у розширенні для нових музичних форматів. Наразі Music Ontology є основою таких проєктів, як BBC Music та MusicBrainz Linked Data Service, що свідчить про її практичну цінність.

MusicBrainz – це відкрита енциклопедія метаданих про музику, що базується на принципах колективного внеску, аналогічно до Вікіпедії. Хоча MusicBrainz спочатку не задумувалася як онтологічна база знань, з часом вона еволюціонувала у потужну систему структурованих даних. Система пропонує унікальні ідентифікатори для виконавців, релізів, треків та інших сутностей, детальну інформацію про авторство, співавторство та виконавців, взаємозв'язки між різними версіями музичних творів, а також інтеграцію з іншими системами через API. Недоліками MusicBrainz є фокус переважно на метаданих, а не на музично-теоретичних аспектах, залежність від внесків користувачів, що може призводити до неоднорідності даних, та обмежена формальна семантика у порівнянні з онтологіями на основі OWL. MusicBrainz активно використовується багатьма сервісами, зокрема Spotify, Last.fm та Amazon Music, для нормалізації та збагачення даних про музичні твори [4]. На рисунку 1.1- інтерфейс веб сайту MusicBrainz.

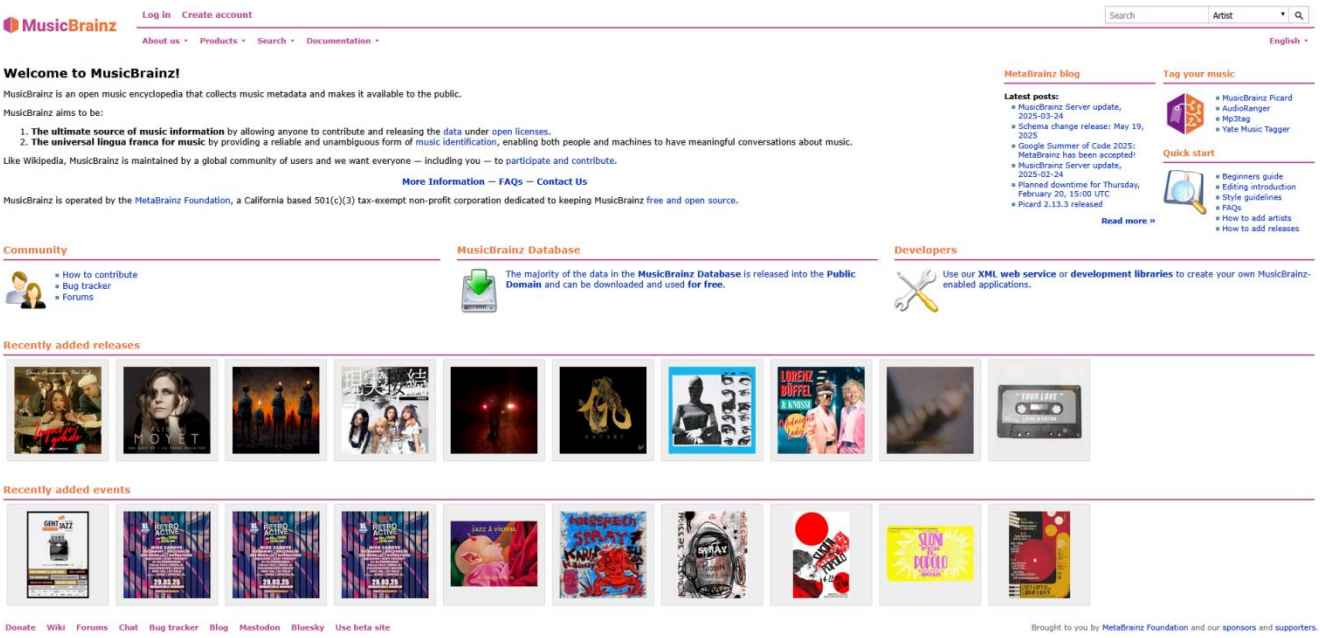


Рисунок 1.1- інтерфейс веб сайту MusicBrainz

Musipedia являє собою унікальну базу знань, що спеціалізується на пошуку

музичних творів за мелодією. На відміну від інших систем, вона дозволяє здійснювати пошук, використовуючи різноманітні методи введення мелодії. Особливості Musipedia включають кілька методів введення мелодії: контурний пошук, ритмічний пошук, пошук за допомогою піаніно, використання алгоритмів порівняння мелодій для ідентифікації схожих творів, та акцент на музичному змісті, а не тільки на метаданих. Обмеженнями системи є відносно невелика база даних порівняно з комерційними альтернативами, обмежена семантична структура та взаємозв'язки між творами, а також відсутність глибокої інтеграції з іншими музичними базами знань [5]. На рисунку 1.2- інтерфейс веб сайту Musipedia/

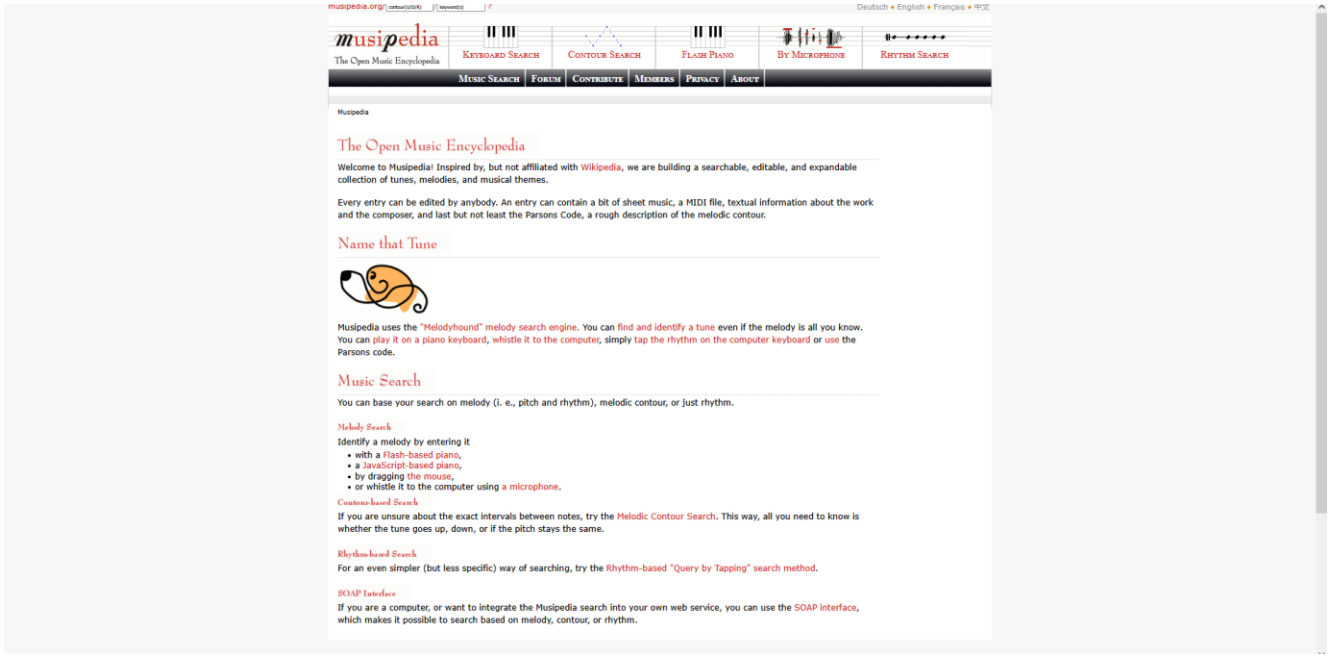


Рисунок 1.2 інтерфейс веб сайту Musipedia

MIDI Linked Data – це проєкт, спрямований на перетворення MIDI-файлів у зв'язані дані (Linked Data), що дозволяє інтегрувати музичну інформацію у Semantic Web. Ключовими рисами цього проєкту є конвертація MIDI-даних у формат RDF, інтеграція з іншими наборами даних, такими як DBpedia та MusicBrainz, можливість виконання складних запитів через SPARQL, збереження музично-теоретичної інформації: тональності, ритму, гармонії. Недоліками є обмеження, пов'язані з форматом MIDI (відсутність інформації про тембр, динаміку), проблеми масштабування при роботі з великими колекціями, недостатньо розвинена інфраструктура для практичного використання [6]. На рисунку 1.3- інтерфейс веб сайту MIDI Linked Data/

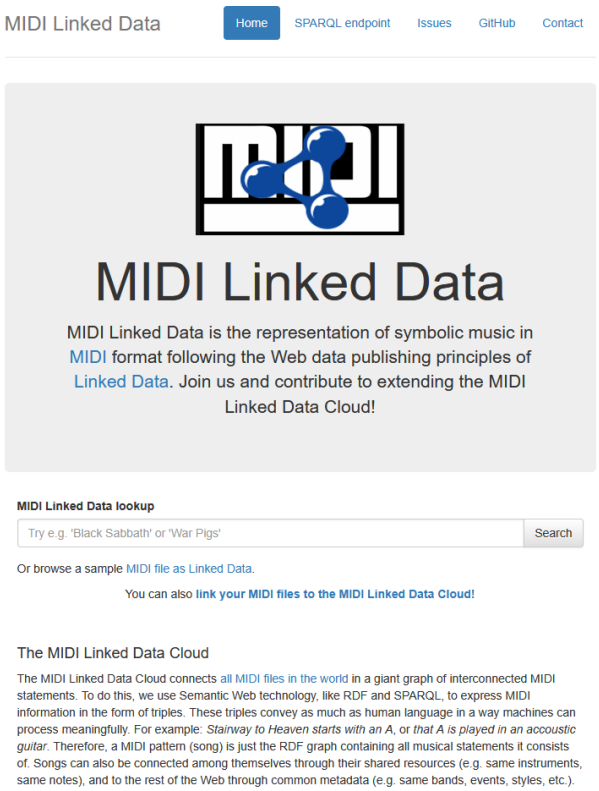


Рисунок 1.3 інтерфейс веб сайту MIDI Linked Data

Spotify, як один з лідерів музичних сервісів, розробив власний граф знань (Spotify Knowledge Graph) для поліпшення системи рекомендацій та пошуку музики. Основними характеристиками цього графа знань є масштабна база даних музичних творів, виконавців та жанрів, інтеграція різноманітних джерел даних: аудіо-аналіз, метадані, поведінка користувачів, складні алгоритми для виявлення взаємозв'язків між музичними сутностями, використання графової структури для представлення музичної інформації. Обмеженнями є закритість системи, відсутність публічного доступу до бази знань, комерційна спрямованість, що впливає на структуру та пріоритети в організації знань, фокус на популярній музиці, менше уваги до нішевих жанрів та історичних творів [7]. На рисунку 1.4- інтерфейс веб сайту Spotify.

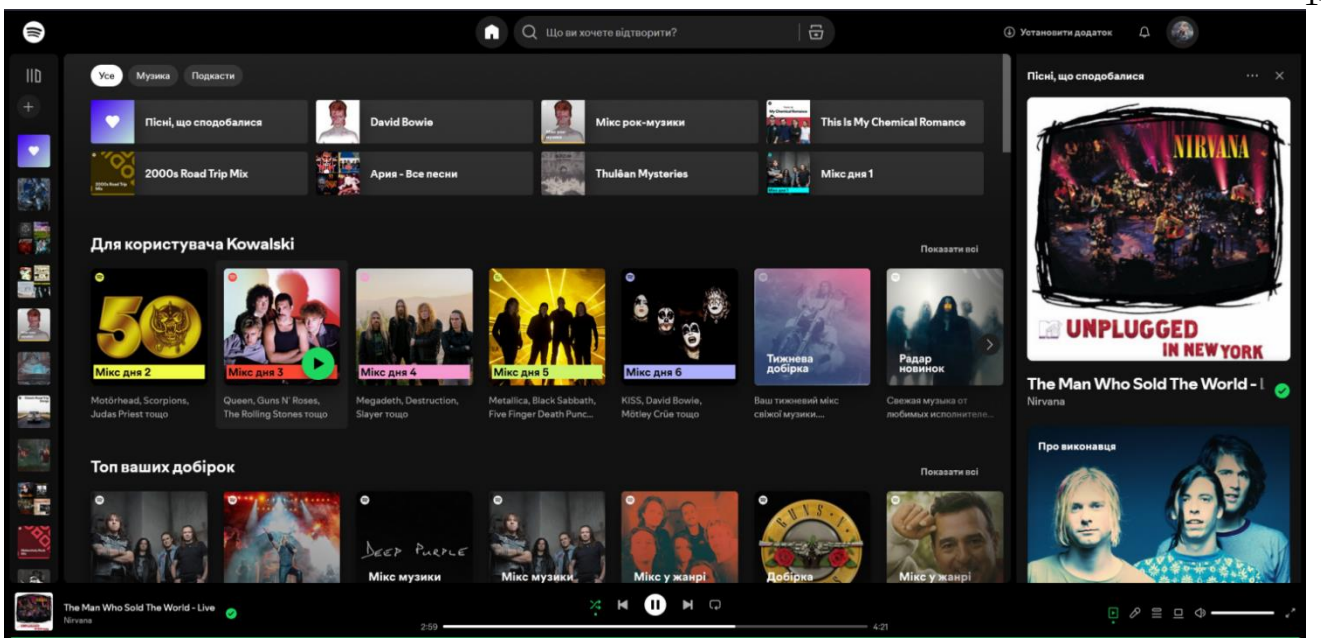


Рисунок 1.4 інтерфейс веб сайту Spotify

У таблиці 1.1 представлено порівняльний аналіз розглянутих баз знань за ключовими критеріями.

База знань	Семантична структура	Масштаб даних	Відкритість
Music Ontology	Висока(RDF/OWL)	Середній	Висока
MusicBrainz	Середня	Великий	Висока
Musipedia	Низька	Малий	Середня
MIDI Linked Data	Висока(RDF)	Малий	Висока
Spotify Knowledge Graph	Висока	Дуже великий	Низька

Таблиця 1.1 – Порівняльний аналіз відомих реалізацій баз знань в предметній області «Музична онтологія»

Проведений огляд виявляє, що наявні інформаційні сховища у музиці охоплюють різноманітні аспекти предметної сфери, відрізняються глибиною семантичної організації та зручністю доступу. Ключові недоліки чинних систем включають неповну інтеграцію теоретичних музичних знань з практичними даними,

обмежену підтримку мультикультурних музичних традицій та маловідомих жанрів, відособленість різних інформаційних джерел та труднощі їх спільного використання, а також недостатню пристосованість до змін у музичних формах та технологіях. Здійснений аналіз підкреслює необхідність розробки нової онтологічної бази знань «Музика», яка б використовувала сильні сторони наявних рішень, усувала їхні недоліки та відповідала сучасним потребам організації знань у сфері музики.

1.3 Формулювання вимог і постановка задачі на дослідження

Метою даної роботи є створення системи, що забезпечує ефективне впорядкування, класифікацію та управління великим обсягом музичних даних, таких як інформація про музичні жанри, виконавців, інструменти, твори та історичні аспекти музики. Для цього передбачається розробка інформаційної технології на основі онтологічного підходу, що дозволяє формалізовано описувати знання та забезпечувати їх подальше використання в аналітичних, освітніх та професійних цілях.

Онтологічне моделювання дає змогу не лише систематизувати знання, але й зробити їх доступними для семантичного пошуку, логічного аналізу та автоматичної обробки, що має важливе значення для цифрових платформ, навчальних систем та дослідницьких проєктів у музичній галузі [7].

Для досягнення зазначеної мети потрібно вирішити такі завдання:

1. Аналіз предметної області:
   - Дослідити вже існуючі методи представлення музичних знань.
   - Визначити головні концепти та взаємозв'язки у музичній сфері, які необхідно внести в онтологічну базу.
   - Вивчити наявні онтологічні моделі у суміжних галузях та оцінити можливість їхнього застосування до музичної сфери.
2. Розробка концептуальної моделі онтологічної бази знань:
   - Формалізувати основні категорії музичних знань (жанри, виконавці, твори, інструменти та ін.).
   - Визначити атрибути та взаємозв'язки між об'єктами в онтології.
   - Створити структуру бази знань з урахуванням можливості розширення та адаптації.

3. Проєктування та реалізація онтологічної бази знань

- Вибрати платформу та інструменти для реалізації онтології (наприклад, OWL, Protégé, RDF)
- Реалізувати онтологічну модель, заповнити її основними об'єктами та зв'язками.
- Впровадити механізми оновлення та розширення бази знань.

4. Розробка механізмів запиту та доступу до даних:

- Визначити способи взаємодії користувачів з базою знань (запити SPARQL, API-доступ, візуальні інтерфейси).
- Забезпечити ефективні алгоритми пошуку інформації.
- Розробити інструменти для семантичного аналізу запитів та генерування відповідей.

5. Тестування та оцінка ефективності онтологічної бази знань:

- Провести тестування функціональності та продуктивності онтології.
- Перевірити правильність взаємозв'язків між об'єктами.
- Оцінити зручність використання та точність отриманих результатів.

Отже, реалізація онтологічної бази знань «Музика» дозволить не лише структурувати інформацію, а й створити ефективний інструмент для її аналізу, дослідження та застосування в різних контекстах. Застосування онтологічного моделювання сприятиме розвитку цифрових технологій у музичній сфері та підвищить якість інформаційних ресурсів для дослідників, викладачів та музичних спеціалістів.

1.4 Висновок до розділу 1

У першому розділі цієї роботи було доведено важливість розробки онтологічної бази знань "Музика". Її основною метою є впорядкування та організація музичних даних, щоб забезпечити простий доступ, детальний аналіз та ефективне використання інформації. Було розглянуто ключові причини, що спонукали до створення онтології, зокрема стрімке збільшення обсягу музичних ресурсів, необхідність уніфікації даних та підвищення ефективності їх пошуку й обробки.

На основі проведеного аналізу було визначено вимоги до онтологічної бази знань та поставлено основні завдання дослідження. Запропонований підхід до

моделювання бази знань передбачає: створення концептуальної моделі, чітке визначення категорій та взаємозв'язків між ними, практичну реалізацію онтології з використанням передових технологій та методів представлення знань. Окрім цього, було підкреслено важливість розробки механізмів доступу до інформації, таких як запити та пошукові алгоритми.

Отже, досягнення цілей, поставлених у цьому розділі, сприятиме створенню ефективної онтологічної бази знань у музичній сфері. Це відкриває можливості для її використання в освітніх, наукових та інформаційно-аналітичних цілях. Наступні розділи роботи будуть зосереджені на детальному проєктуванні, впровадженні та тестуванні розробленої системи.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ БАЗИ ЗНАНЬ «МУЗИЧНА ОНТОЛОГІЯ»

2.1 Визначення межі та масштабу онтологій

Запуск розробки онтології бази знань «Музична онтологія» потребує першочергового окреслення її кордонів та обсягу – це базовий крок у процесі побудови впорядкованої інформаційної системи. Йдеться про окреслення основних категорій знань, що будуть подані в онтології, окреслення ключових понять, взаємозв'язків між ними та можливих обмежень у межах досліджуваної галузі.

Чітке визначення меж онтології має вирішальне значення, адже саме воно задає ступінь деталізації понять, які концепції будуть враховані, та які взаємозв'язки між ними будуть зафіксовані. Чим краще окреслені межі, тим продуктивніше буде використання онтології у відповідних сферах. Варто пам'ятати, що онтологія мусить бути адаптивною, щоб мати потенціал для подальшого розширення за потреби [9].

Основні питання для визначення масштабу онтології

При розробці онтології необхідно відповісти на кілька важливих запитань:

1. Які аспекти музичної сфери будуть охоплені?

- Чи буде онтологія орієнтована лише на музичні жанри та виконавців, або включатиме також інструменти, історію музики, вплив музичних стилів один на одного?
- Чи потрібно враховувати музичні теорії, гармонії та ноти?
- Чи варто включати інформацію про вплив музики на різні культури та соціальні явища?

2. Яке практичне застосування передбачає онтологія?

- Чи буде вона використовуватися для навчальних цілей, для музичних досліджень, або як база знань для музичних рекомендаційних систем?
- Чи необхідно включати інформацію про вплив музики на психологічний стан людини?
- Чи буде вона інтегрована у музичні потокові сервіси для кращої персоналізації контенту?

3. Які запити зможе обробляти онтологія?

- Чи буде онтологія відповідати на загальні запитання, наприклад, "Які відомі виконавці у стилі джаз?" або "Які музичні інструменти використовуються в класичній музиці?"
- Чи буде вона використовуватися для складнішого аналізу, наприклад, "Як змінювався стиль рок-музики протягом десятиліть?"
- Чи може вона аналізувати популярність жанрів у різні періоди історії та передбачати їх майбутній розвиток?

4. Хто є цільовою аудиторією та користувачами онтології?

- Чи буде вона корисна для музичних критиків, дослідників, викладачів, студентів, музикантів або широкої аудиторії?
- Чи передбачає вона інтеграцію з іншими музичними базами даних та онлайн-платформами?
- Чи варто враховувати специфіку сприйняття музики різними культурами та демографічними групами?

Масштаб та деталізація онтології

Визначення масштабу онтології є ключовим для забезпечення її працездатності та коректності. Виходячи з обраного рівня деталізації, можливі такі підходи:

- Вузькоспеціалізована онтологія – зосереджена лише на конкретному напрямку музики, наприклад, на будові музичних творів або на еволюції певних жанрів.
- Розширена онтологія – вміщує широке коло музичних понять, розкриваючи взаємозв'язки між жанрами, виконавцями, музичними інструментами та теоретичними основами музики.
- Універсальна онтологія – охоплює як загальні, так і детальні відомості про всі аспекти музики, включно з технічними особливостями запису звуку, процесом створення музичних інструментів та впливом музики на суспільство.

Перевірка компетентності онтології

Одним зі способів з'ясування відповідності онтології поставленим вимогам є створення тестових запитів, котрі вона повинна правильно опрацьовувати. Це сприяє

оцінюванню її дієвості у пошуку музичної інформації за чіткою структурою. Приклади таких запитів:

1. Які ключові риси джазової музики?
2. Хто з композиторів писав класичну музику в період бароко?
3. Чим відрізняються блюз та рок-н-рол?
4. Які музичні інструменти найчастіше використовуються у симфонічному оркестрі?
5. Як змінились напрями в поп-музиці за останні двадцять років?
6. Як впливають культурні традиції на формування національної музики?
7. Які головні принципи побудови мелодії у фольклорній музиці?
8. Які відомі музичні фестивалі справили вплив на розвиток світової музичної арени?

Визначення межі та масштабу онтології є критичним етапом у її розробці. Чітко окреслені межі дозволяють уникнути зайвого ускладнення моделі та забезпечують ефективне використання бази знань у майбутньому.

Створення онтологічної моделі, що відповідає на поставлені питання, сприятиме розвитку музичних досліджень, аналітики та автоматизованого оброблення музичних даних. Крім того, правильне визначення меж дозволить оптимізувати використання ресурсів та забезпечить ефективне розширення онтології у випадку необхідності доповнення новими знаннями [9].

2.2 Створення термінологічного словника онтологій

У процесі проєктування онтологічної бази знань «Музика» особливе значення має створення термінологічного словника, який забезпечує формальне визначення понять, що використовуються в моделі, а також зв’язків між ними. Термінологічний словник є ядром онтології: він задає ієрархію класів, описує атрибути (властивості) об'єктів, визначає можливі зв’язки та обмеження. Без такого словника онтологія втрачає цілісність і не може бути використана для інтелектуального аналізу або побудови запитів.

Загальна структура словника:

Онтологія «Музика» моделюється на основі класу owl:Thing, що є кореневим

поняттям у стандарті OWL. [10] Від нього успадковуються всі інші класи предметної області. Нижче подано огляд основних класів, що формують структуру онтології, згідно з представленою ієрархією:

- Гурт/Композитор (Band/Composer) — представляє музичних виконавців, ансамблі або окремих авторів. Має атрибут Назва_гурта, який визначає унікальне ім’я цього класу.
- Жанр (Genre) — категорія, що описує стилістичну приналежність музичного твору. Має підклас Піджанр, який деталізує основний жанр.
- Лейбл (Label) — організація або компанія, що відповідає за запис, просування та розповсюдження музичних творів.
- Музичний_твір (Musical_Work) — загальна сутність для будь-якої музичної композиції, яка може мати підклас Альбом (Album) і властивість Назва_музичного_твору.
- Нагорода (Award) — сутність, що відображає досягнення або відзнаки, присуджені за музичні твори чи виконавцям.
- Платформа (Platform) — цифровий або фізичний канал дистрибуції музики (наприклад, Spotify, YouTube, вініли тощо).
- Подія (Event) — концерт, фестиваль, реліз або інша форма публічного виконання або презентації музики.
- Інструмент (Instrument) — класифікація музичних інструментів, що використовуються в композиціях.

Типи властивостей

У термінологічному словнику виділяються два типи властивостей:

Object properties (об'єктні властивості) — встановлюють зв’язки між екземплярами класів. Наприклад:

- створив (Гурт/Композитор → Музичний_твір)
- належитьДоЖанру (Музичний_твір → Жанр)
- виконуєтьсяНа (Музичний_твір → Подія)
- включеноВ (Музичний_твір → Альбом)
- опубліковановНа (Музичний_твір → Платформа)

Datatype properties (властивості-дані) — описують внутрішні характеристики об’єктів.

Наприклад:

- Назва_гурта (тип: string)
- Назва_музичного_твору (тип: string)
- Дата_релізу (тип: date)
- Тривалість (тип: time або float)
- Ціна_альбому (тип: decimal)

Уніфікація термінології

Словник забезпечує однозначне трактування понять. Наприклад, термін «твір» чітко співвідноситься з класом Музичний_твір, тоді як «альбом» — його підклас. У випадках, коли одна й та сама назва може позначати різні об'єкти (наприклад, слово «гурт» може бути як колективом, так і брендом), у словнику дається чітке визначення контексту застосування терміна.

Семантична узгодженість

Одним із ключових завдань при створенні словника є семантична узгодженість між поняттями. Наприклад, якщо Гурт має відношення до Музичний_твір, то у зворотному напрямі має бути логічна відповідність, наприклад, виконавець.

Також встановлюються домени та діапазони для властивостей. Наприклад:

Object property: створив

Domain: Гурт/Композитор

Range: Музичний_твір

Це забезпечує логічну узгодженість та дозволяє автоматизованим системам перевіряти коректність введених зв'язків.

Ієрархія понять

Згідно з зображеною структурою, використано двоступеневу ієрархію:

- Клас Жанр має підклас Піджанр, що дозволяє деталізувати стилістичну класифікацію.
- Клас Музичний_твір деталізується до рівня Альбом, в який можуть входити індивідуальні композиції.

Це дозволяє формувати запити різного рівня глибини: від загального запиту "знайти всі твори жанру рок" до специфічного "знайти альбоми піджанру

прогресивний метал".

Принципи побудови словника

При створенні термінологічного словника було дотримано наступних принципів:

1. Мінімальна необхідність – включено лише ті поняття, які мають конкретне значення для моделювання музичної сфери.
2. Масштабованість – кожне поняття має потенціал для подальшого розширення, зокрема шляхом додавання властивостей або підкласів.
3. Семантична узгодженість – між усіма об’єктами та властивостями встановлено логічно обґрунтовані зв’язки.
4. Формалізованість – словник реалізовано в середовищі Protégé із використанням OWL, що забезпечує сумісність з стандартами Semantic Web.

У Таблиці 2.1 представлено основні класи та властивості термінологічного словника

Клас	Опис	Основні властивості
Гурт/Композитор	Музичний виконавець або автор	Назва_гурта (string), створив (→ Музичний_твір)
Жанр	Стилістична категорія музики	має_піджанр (→ Піджанр)
Піджанр	Похідний жанр	належитьДо (→ Жанр)

Таблиця 2.1 – основні класи та властивості термінологічного словника в предметній області «Музика»

Лейбл	Компанія-розсповсюджувач	назва, рік заснування
Музичний твір	Окрема композиція або трек	належитьДо (→ Жанр)
Альбом	Збірка творів	містить (→ Музичний_твір)
Нагорода	Відзнака за досягнення	назва, рік вручення
Платформа	Цифровий сервіс або фізичний носій	тип, назва
Подія	Концерт або фестиваль	місце, дата
Інструмент	Засіб виконання музики	Тип, історія, категорія

Таблиця 2.1 - Продовження таблиці про основні класи та властивості термінологічного словника в предметній області «Музична онтологія»

2.3 Моделювання онтології засобами Protege

Створення онтологічної бази знань «Музика та кіно» є важливим етапом у розробці структурованої системи управління знаннями, яка дозволяє організувати та зберігати інформацію про музичні композиції, гурти, композитори, альбоми рекомендації щодо прослуховування та інші суттєві аспекти цієї галузі. Інструмент Protege надає можливість моделювати ці елементи в єдиній системі, що полегшує налаштування, подальше використання та інтеграцію знань.

Процес моделювання онтології у Protege складається з кількох ключових етапів: визначення класів, додавання атрибутів, встановлення ієрархій та створення об’єктних властивостей для відображення взаємозв'язків. Такий підхід забезпечує детальне та гнучке представлення інформації про музику, що дозволяє систематизувати знання для подальшого аналізу та використання [10].

Основні етапи процесу моделювання в Protege.

1) Створення класів

На першому етапі створено базові класи, які відображають ключові об’єкти

предметної галузі. Зокрема: МузичнийТвір; Жанр; Альбом; Виконавець; Композитор/Гурт; Саундтрек; Фільм (для зв’язку з саундтреками);

Процес створення класів починається з визначення основних категорій, що будуть включені до онтології. У Protege кожен клас створюється з індивідуальним ім’ям, що полегшує ідентифікацію та посилання на нього в моделі. Створені класи зображені на рисунку 2.1.

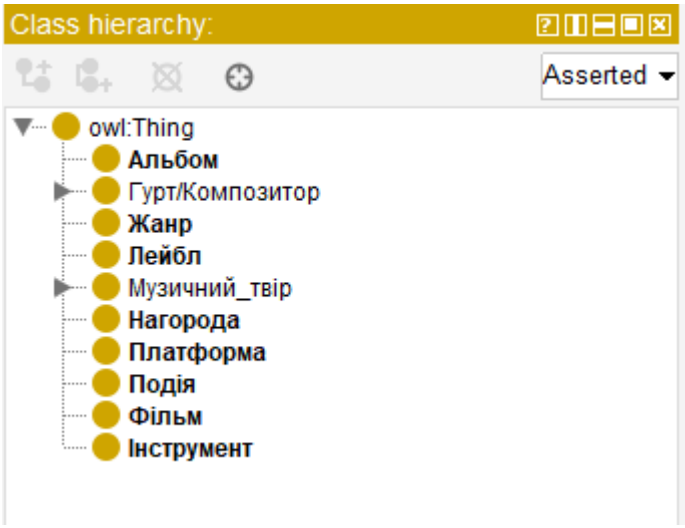


Рисунок 2.1– Класи онтологічної бази знань «Музична онтологія»

Кожен клас відповідає важливому аспекту музики, що дозволяє деталізувати всі ключові елементи предметної області. Наприклад, клас "Гурт/Копмозитор" охоплює всі музичні гурти, тоді як "Виконавці" містить виконавців, які можуть входити до складу гуртів/копмозиторів.

2) Побудова ієрархії класів

Після створення базових класів будується ієрархічна структура для відображення зв'язків між загальними класами та підкласами. Ієрархія класів допомагає деталізувати об'єкти моделі та організувати знання у вигляді дерева.

Класи організовано в логічну ієрархію. Наприклад, класи Гурт, Композитор/Гурт, Продюсер, Актор об'єднані в надклас Особа. Це дозволяє гнучко відображати спільні характеристики для суб'єктів музичної (і частково кінематографічної) індустрії.

Клас МузичнийТвір не поділяється на підкласи, натомість описується через властивості (жанр, виконавець, альбом тощо).

Ієрархія класів зображена на рисунку 2.2.

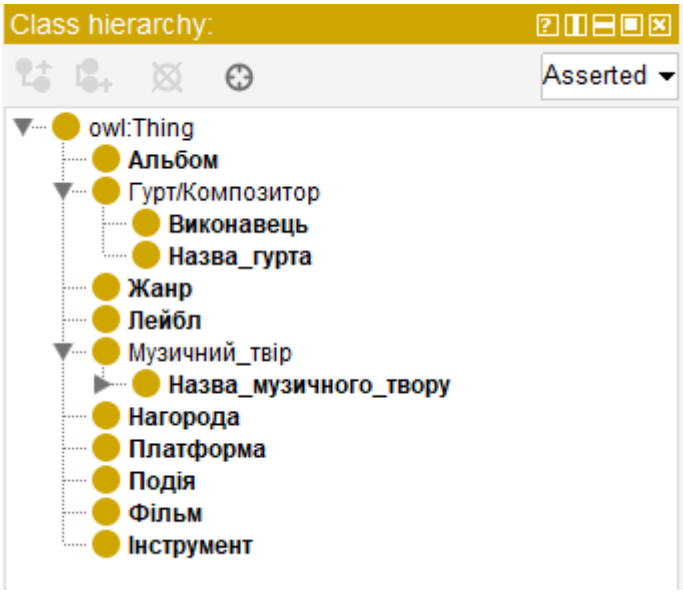


Рисунок 2.2 – Ієрархія класів онтологічної бази знань «Музична онтологія»

3. Додавання атрибутів (datatype properties)

- Кожному класу були призначені відповідні атрибути. Наприклад:
- МузичнийТвір має властивості: назва, рікВипуску, тривалість, жанр
- Альбом — назва, рікВипуску, лейбл
- Особа — ім’я, датаНародження, національність

Створені атрибути зображені на рисунку рисунку 2.3-2.5.

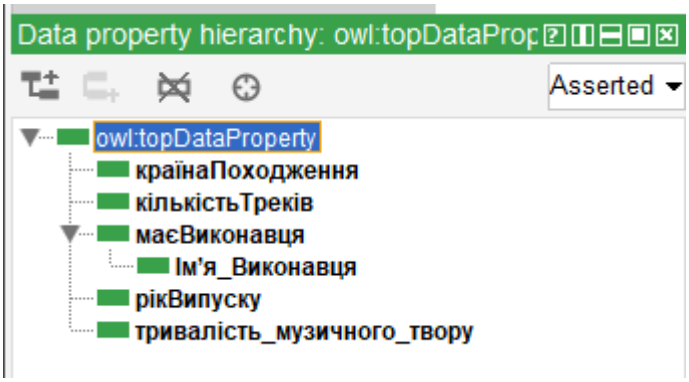


Рисунок 2.3 – Атрибути класів онтологічної бази знань «Музична онтологія»

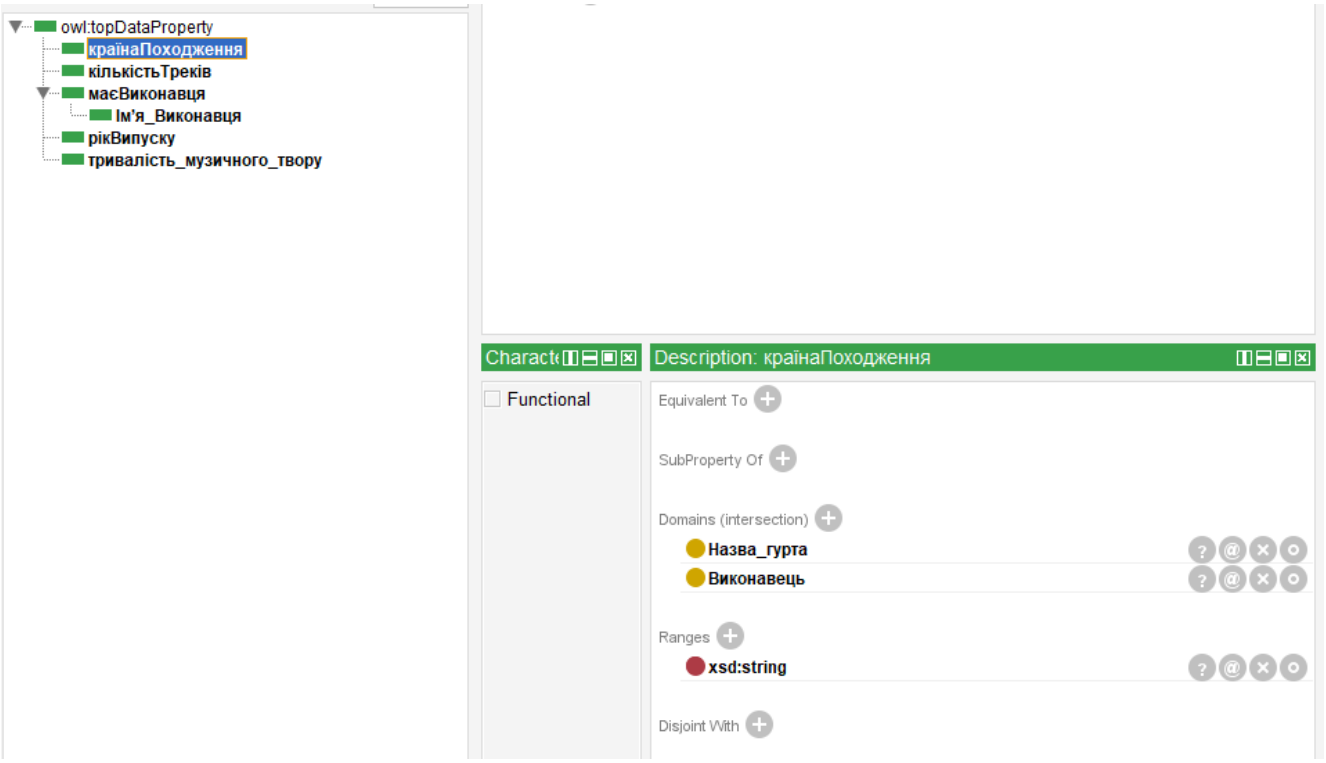


Рисунок 2.4 – Атрибут «країна Походження»

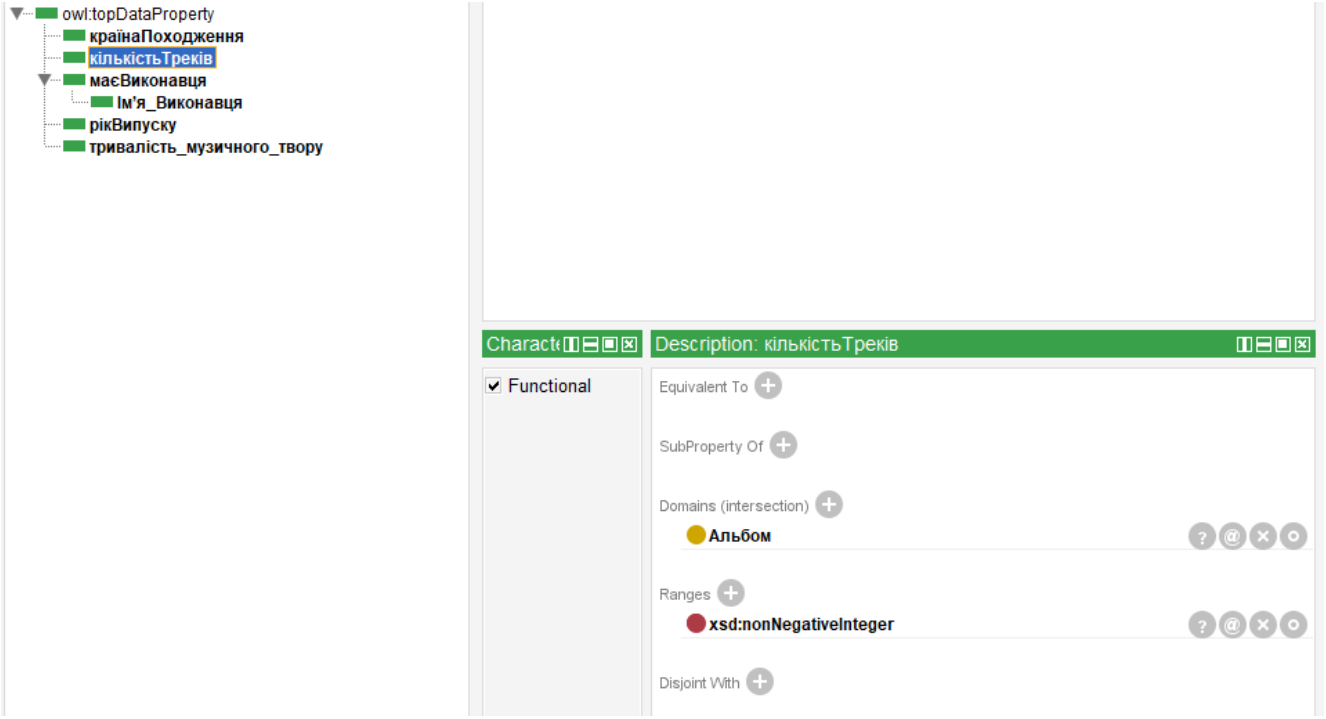


Рисунок 2.5 – Атрибут «кількість Треків»

Важливість атрибутів полягає у деталізації даних, що дозволяє отримати повнішу інформацію про кожен об'єкт. Це забезпечує не лише точність, але й робить онтологію функціональнішою, оскільки кожен атрибут виконує свою роль у побудові моделі. Атрибути надають можливість зберігати додаткові характеристики для кожного класу, такі як тривалість треку, кількість треків в альбомі.

4. Встановлення об’єктних властивостей (object properties)

Об'єктні властивості дозволяють задавати зв'язки між класами, відображаючи взаємодію між різними елементами онтології. У Protege об'єктні властивості налаштовуються індивідуально для кожного класу, що робить модель реалістичнішою та функціональною.

Для відображення взаємозв’язків між класами було створено об’єктні властивості:

- виконано — зв'язує МузичнийТвір із Виконавець
- належитьДоЖанру — МузичнийТвір → Жанр
- єСаундтрекомДо — МузичнийТвір → Кінофільм
- автор — МузичнийТвір → Композитор/Гурт
- міститьсяВАльбомі — МузичнийТвір → Альбом

Створені об’єктні властивості зображені на рисунках 2.5-2.8

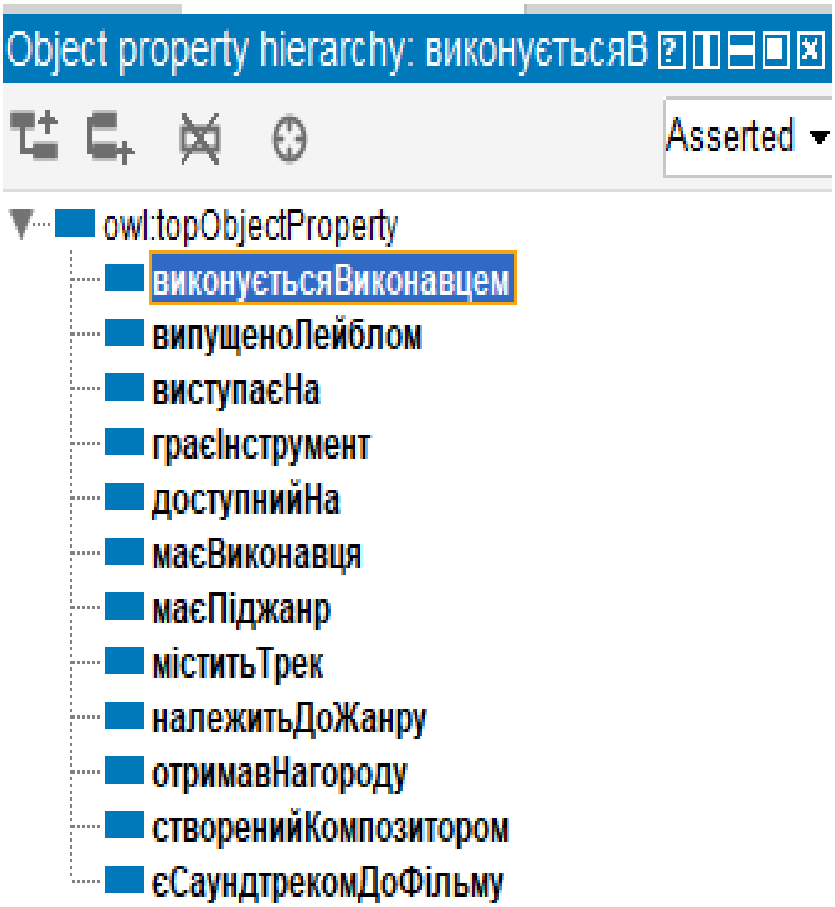


Рисунок 2.6 – Об’єктні властивості онтологічної бази знань «Музична онтологія»

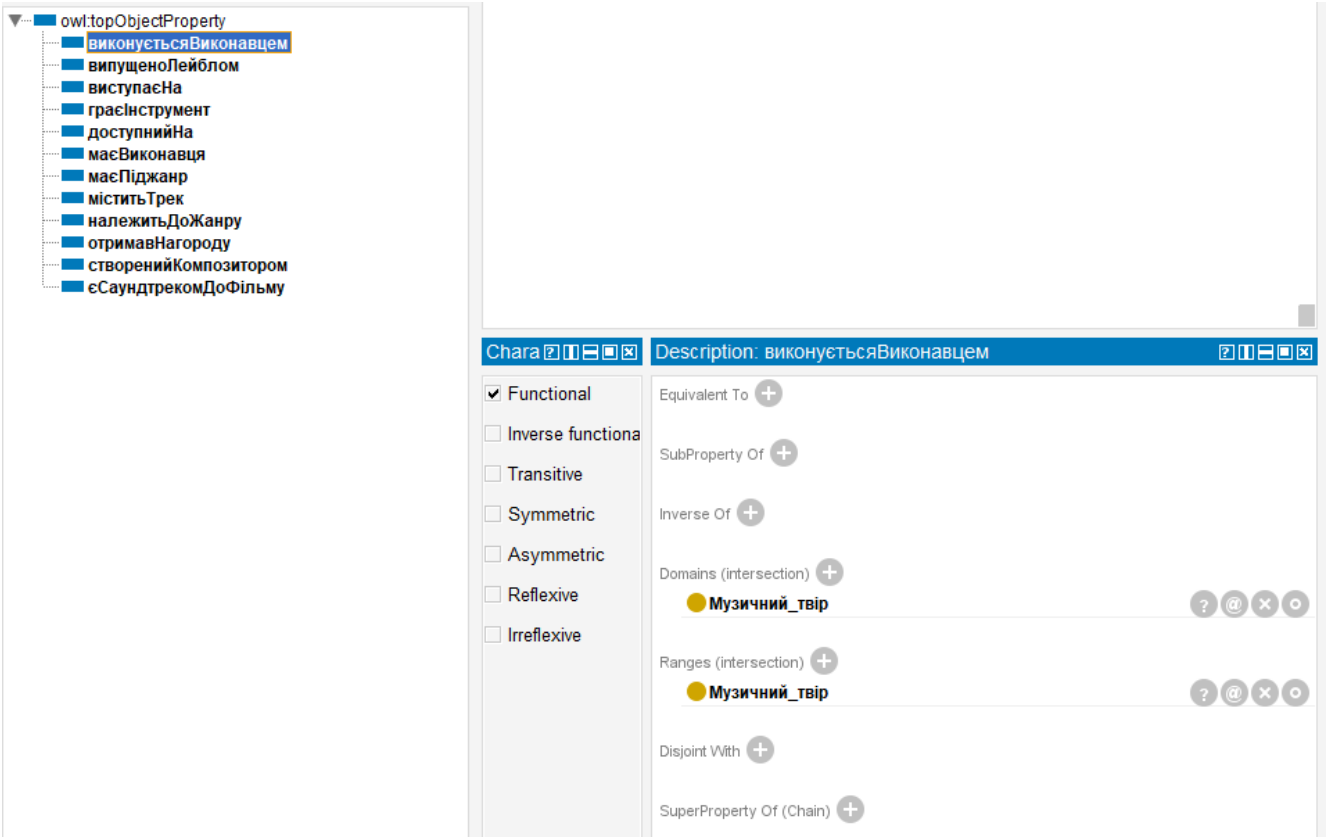


Рисунок 2.7 – Об’єктна властивість «виконуєтьсяВиконавцем»

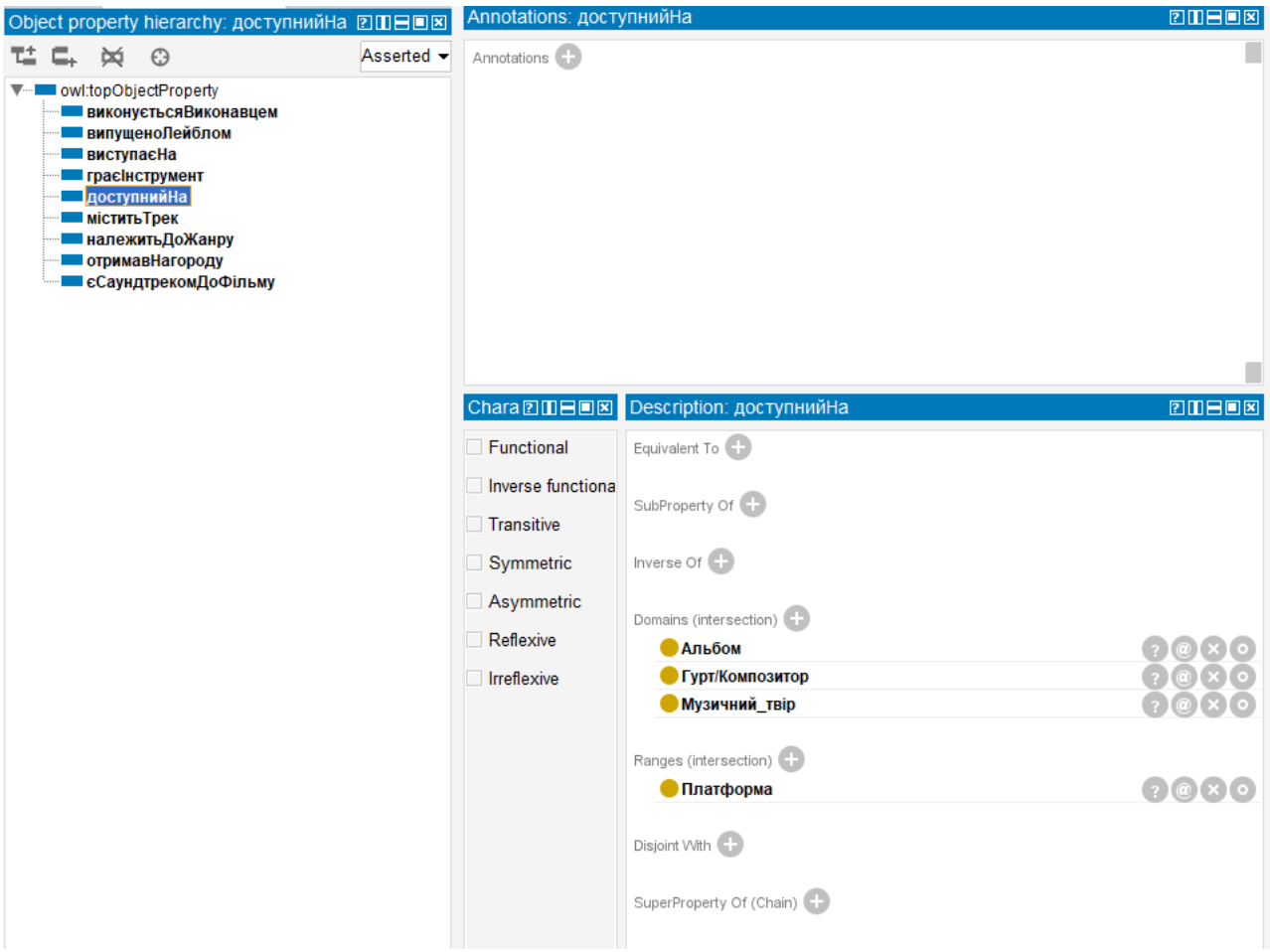


Рисунок 2.8 – Об’єктна властивість «доступнийНа»

Важливість об'єктних властивостей полягає у побудові структури відносин між класами, яка є основою для онтологічної моделі [10]. Це дозволяє системі зберігати інформацію про взаємодію між об'єктами, що є суттєвим для управління знаннями. Зв'язки створюють єдину картину функціонування системи, що полегшує аналіз даних та інтеграцію знань.

5. Створення індивідів

Для наповнення онтології прикладами, було створено індивідів:

- Eye_of_the_tiger— Альбом, автор: Survivol, жанр: Rock
- Exit_music — МузичнийТвір, гурт: Radiohead, альбом: Ok_Computer
- ABBA — Композитор/Гурт
- Where_is_my_mind?— МузичнийТвір, єСаундтрекомДо: Бійцівський клуб

Створені індивіди зображені на рисунках 2.9-2.10

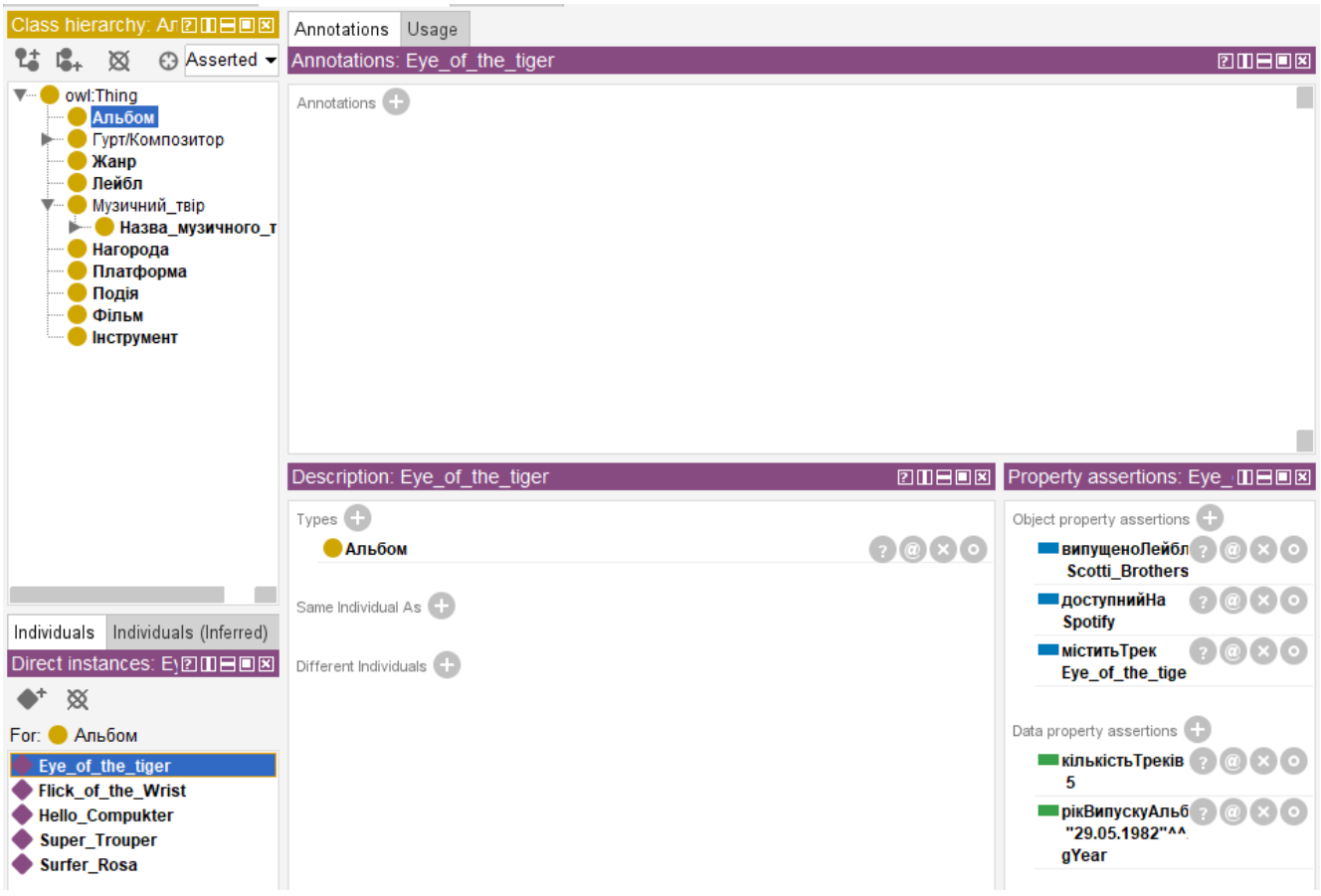


Рисунок 2.9 – Індивіди класу «Альбом»

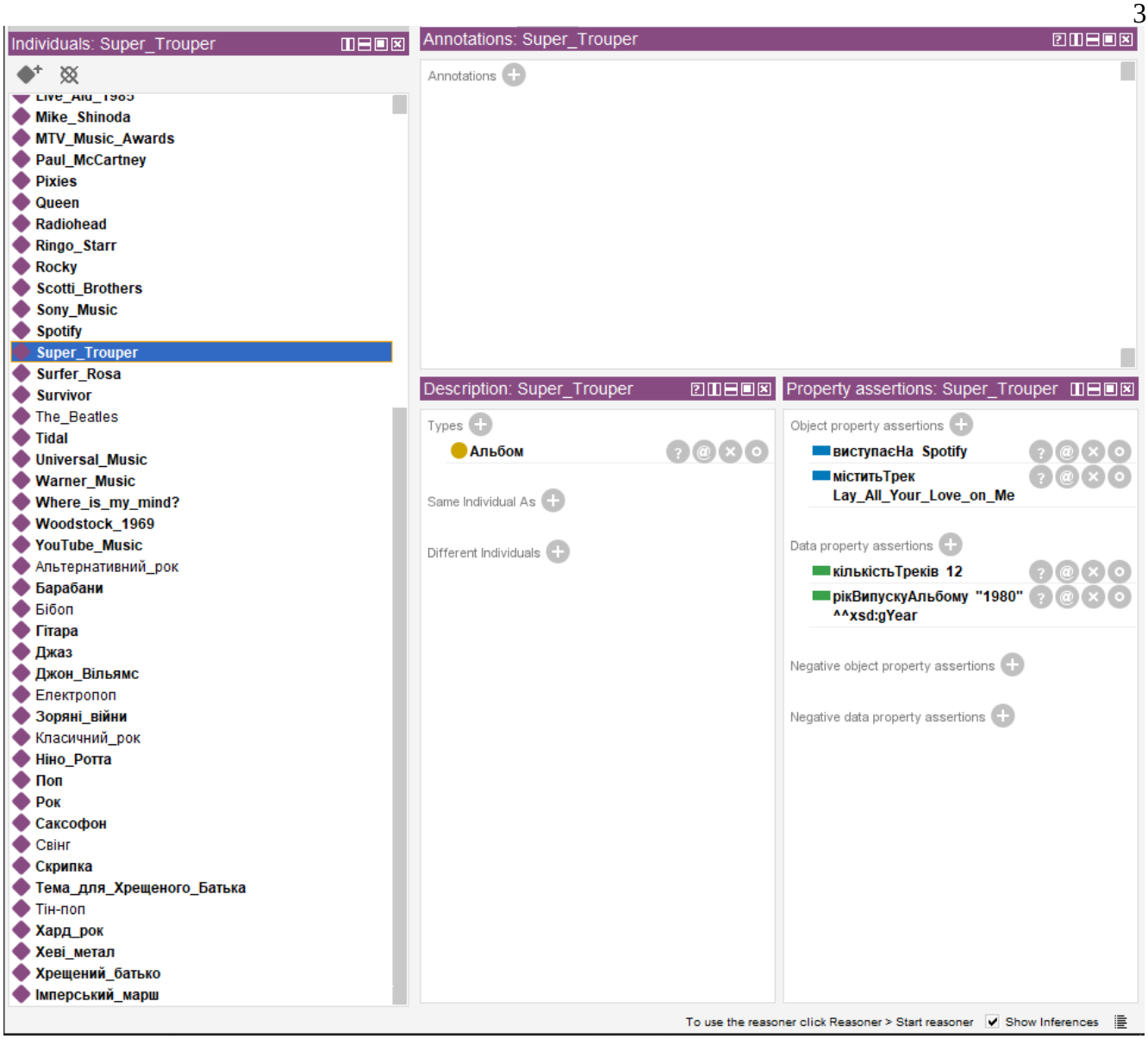


Рисунок 2.10 – Індивіди всієї онтології

6. Візуалізація структури

Для візуалізації було використано плагін OntoGraph. На графі видно класи, їх підкласи та зв’язки між ними. Це полегшує розуміння структури онтології.

На рисунку 2.11 зображено OntoGraph онтологічної бази знань музика

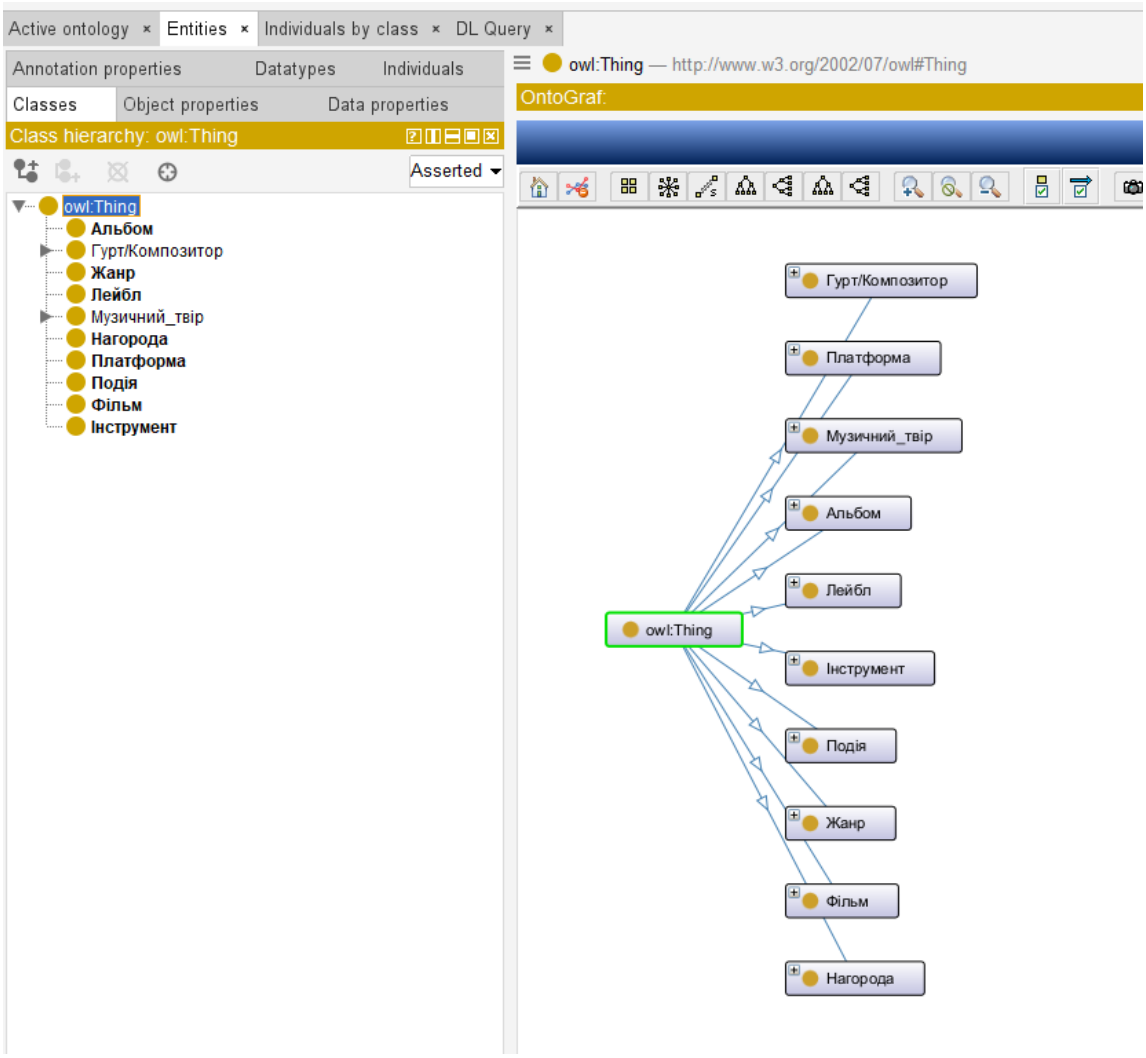


Рисунок 2.11 – Візуалізація онтології за допомогою OntoGraph

Використано OntoGraph. Наочно відображено зв’язки між індивідами. Завдяки графічному інтерфейсу OntoGraph можна швидко побачити не лише класи, а й зв’язки між ними у вигляді вузлів та стрілок. Наприклад, відобразити, які саме музичні твори належать певному жанру, хто є їхнім автором, у яких альбомах вони представлені. Це особливо зручно для перевірки цілісності та логіки побудованої моделі. OntoGraph дозволяє масштабувати, переміщати й групувати об'єкти для полегшення сприйняття. Візуалізація стає надійним інструментом аналізу як при розробці онтології, так і під час її використання.

2.4 Висновок до розділу 2

У другому розділі було виконано кілька основних етапів, спрямованих на створення ефективної онтологічної моделі для бази знань косметичних засобів.

По-перше, було розроблено інформаційну технологію для побудови онтологічної моделі, яка вдосконалює процес моделювання за методологією семи кроків. Це вдосконалення сприяло чіткішій структуризації предметної області та підвищило якість створеної моделі.

Далі, з використанням графічних методів, було змодельовано онтологічну базу знань, що дозволило наочно представити основні елементи системи та їхні взаємозв’язки. Такий підхід забезпечив структуроване уявлення про музичні твори, альбоми, групи/композитори та інші елементи, створивши основу для подальшої інтеграції та застосування онтології в аналітичних та управлінських завданнях.

Крім того, на основі розробленої інформаційної технології, онтологічну базу знань було реалізовано за допомогою інструмента Protege. Це дозволило чітко структурувати інформацію про косметичні засоби, створити ієрархічну організацію класів і підкласів, а також додати атрибути й об’єктні властивості, що забезпечують гнучкість та деталізацію моделі. Інструмент також надав можливість для подальшого аналізу, тестування й адаптації бази знань, що є важливим для збереження її актуальності та інтеграції в майбутні програмні системи.

3 РЕАЛІЗАЦІЯ ОНТОЛОГІЧНОЇ БАЗИ ЗНАНЬ «МУЗИЧНА ОНТОЛОГІЯ» В СЕРЕДОВИЩІ PROTEGE

3.1 Розширення функціональності онтології в межах Protégé

Плагіни в Protege є одним з основних засобів розширення функціональних можливостей онтології. Їх налагодження в першу чергу регулюється завантаженням файлів з відкритим вихідним кодом з Github або інших платформ, після чого вже триває повна установка та тестування. При установці кожного нового плагіну важливо встановити ряд налаштувань – параметрів, які показують, як буде працювати це розширення, надають йому ряд прав і обмежень.

Існує кілька основних видів плагінів, кожен із яких виконує специфічні функції. Наприклад, плагін типу TabWidget відкриває окреме вікно для роботи з даними — дозволяє виконувати операції CRUD. Однак при цьому виникають труднощі у взаємодії з іншими вкладками або вікнами.

SlotWidget використовується для управління інтерфейсом користувача: він відображає і дозволяє редагувати значення окремого слота, дотримуючись протоколу взаємодії з базою знань. Проте, як і TabWidget, він найефективніший лише при роботі з одним слотом.

KnowledgeBaseFactory призначений для заміни стандартного способу зберігання даних, включаючи зовнішні сервери або бази даних. Його основні недоліки — складність інтерфейсу та часті помилки реалізації. Водночас цей плагін є єдиним засобом, що дає змогу переглядати й аналізувати складні формати файлів.

ProjectPlugin забезпечує доступ до всіх стандартних функцій проєкту: перегляду, меню, панелі інструментів, а також дозволяє змінювати їхній вигляд відповідно до потреб користувача.

ExportPlugin дає змогу зберігати частини бази знань у довільному форматі в іншому місці, тоді як ImportPlugin виконує зворотну функцію — створює базу знань на основі вхідних даних [11].

Процес встановлення та налаштування плагінів

Встановлення плагінів у Protégé може здійснюватися кількома способами,

залежно від типу плагіну та його джерела. Основними методами є автоматичне встановлення через репозиторій та ручне встановлення з файлів.

Автоматичне встановлення відбувається через вбудований менеджер плагінів, який доступний через меню File → Check for plugins. Цей метод забезпечує найвищий рівень сумісності та автоматично вирішує більшість конфліктів залежностей. Процес включає пошук необхідних плагінів у офіційному репозиторії, їх завантаження та автоматичну інтеграцію з основним функціоналом Protégé.

Ручне встановлення застосовується для плагінів, які не представлені в офіційному репозиторії або для встановлення специфічних версій. Цей процес передбачає завантаження JAR-файлів плагінів та їх розміщення в директорії plugins теки встановлення Protégé. При цьому необхідно забезпечити наявність всіх необхідних залежностей та правильність конфігурації.

Після встановлення кожен плагін потребує індивідуального налаштування параметрів роботи. Це включає визначення рівнів доступу, встановлення обмежень використання ресурсів та конфігурацію інтеграції з користувацьким інтерфейсом. Особливу увагу слід приділити налаштуванню конфігураційних файлів, які зазвичай зберігаються у форматі XML або properties та містять критично важливі параметри функціонування плагіну.

Тестування та валідація плагінів**.** Після встановлення та налаштування необхідно провести комплексне тестування функціональності плагінів. Це включає перевірку базових операцій, тестування сумісності з іншими компонентами системи та оцінку впливу на загальну продуктивність Protégé. Особливо важливим є тестування при роботі з великими онтологіями, оскільки деякі плагіни можуть демонструвати нестабільну поведінку при обробці значних обсягів даних [11].

Установка плагінів та їх налаштування в Protege виглядає так, як показано на рис.3.1.

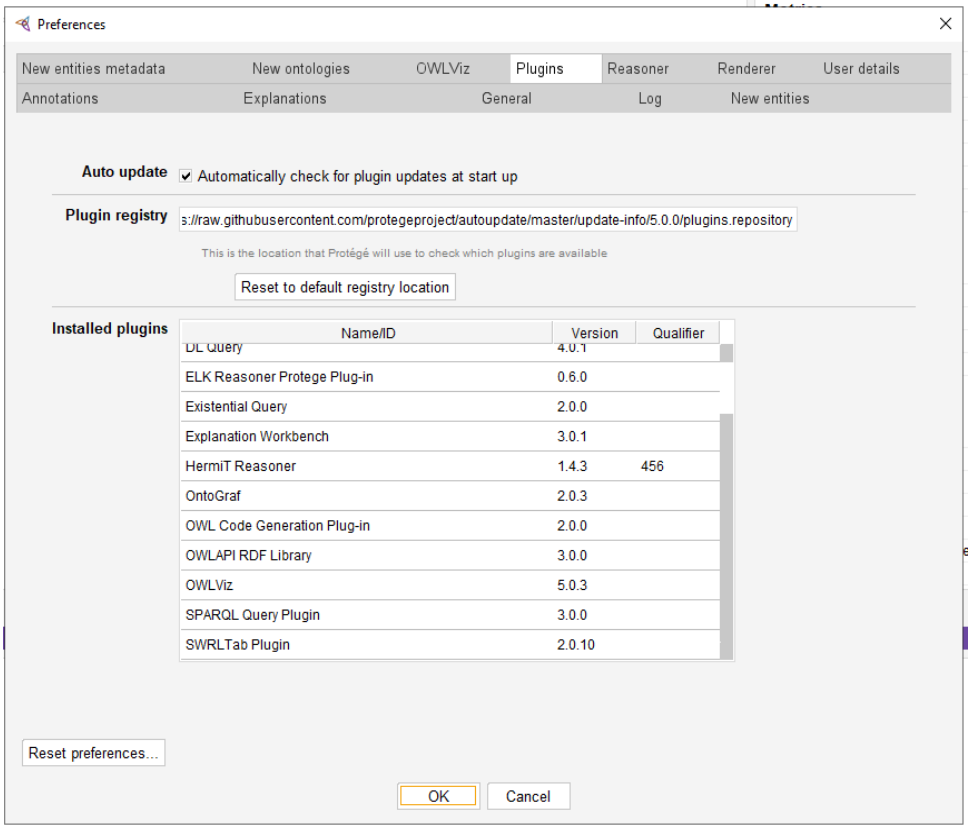


Рисунок 3.1 – Встановлення плагінів в Protege

OntoGraf є стандартним інструментом для візуалізації онтологій і належить до типу плагінів SlotWidget, вбудованих у Protégé 5.0 за замовчуванням. Саме за допомогою цього інструмента вперше було візуалізовано онтологію предметної області музичного наджанру «метал». Інструмент забезпечує зручну візуалізацію ієрархічних структур та відношень між класами онтології, представляючи їх у вигляді інтерактивного графічного дерева з можливістю динамічного розгортання вузлів.

Архітектура та принципи роботи OntoGraf реалізує модель візуалізації, що базується на направлених ациклічних графах (DAG), де кожен клас онтології представлений окремим вузлом, а відношення між класами відображаються у вигляді ребер різних типів. Інструмент використовує алгоритми автоматичного розміщення вузлів, що забезпечує оптимальне візуальне представлення навіть для складних ієрархічних структур.

Система підтримує різні типи візуальних елементів для представлення онтологічних конструкцій: класи відображаються у вигляді прямокутників з різним кольоровим кодуванням залежно від їх типу та рівня в ієрархії, властивості об'єктів представлені стрілками з підписами, а обмеження та правила виділяються спеціальними графічними маркерами.

Користувацький інтерфейс OntoGraf включає панель інструментів з функціями масштабування, фільтрації та пошуку, що дозволяє ефективно працювати з великими онтологічними структурами. Система підтримує режими перегляду "overview" для загального огляду всієї онтології та "detail view" для детального аналізу окремих фрагментів.

Особливо корисною є функція інтерактивного пошуку, яка дозволяє швидко знаходити конкретні класи або властивості в межах візуалізованої структури. При цьому система автоматично підсвічує знайдені елементи та прокладає візуальний шлях до них від кореневих класів онтології [12].

Попри функціональні можливості, OntoGraf має низку обмежень, які впливають на ефективність роботи з великими онтологіями:

По-перше, при великій кількості даних його використання ускладнюється, оскільки розкривати всі гілки доводиться вручну, що значно сповільнює процес навігації та аналізу структури онтології. Це особливо критично при роботі з музичними онтологіями, де кількість жанрів, піджанрів та їх взаємозв'язків може сягати кількох тисяч елементів.

По-друге, у разі помилкового встановлення властивостей між класами, зв'язки порушуються, що змушує заново задавати всі правила та відношення і фактично перебудовувати онтологію. Це створює ризик втрати даних та потребує додаткових зусиль для відновлення цілісності структури. Особливо проблематичним це стає при роботі з циклічними залежностями, які можуть виникати в музичних онтологіях через складні відношення між жанрами.

По-третє, інструмент не враховує окремих індивідів класів у своїй структурі, не відображає їх графічно, а також ігнорує деякі типи зв'язків. Зокрема, OntoGraf не може адекватно відобразити симетричні властивості (наприклад, "схожий на" між музичними жанрами) та транзитивні відношення (як "є піджанром"), що критично важливо для повного представлення музичної таксономії.

По-четверте, система має обмеження щодо відображення анотацій та коментарів до класів, що ускладнює розуміння семантики окремих елементів онтології без звернення до додаткових панелей Protégé. Це особливо проблематично при роботі з мультимовними онтологіями, де анотації можуть містити переклади назв жанрів різними мовами.

Через зазначені обмеження, для забезпечення повної візуалізації окремих елементів предметної області, іноді доводиться штучно задавати суперечливі значення одним і тим же класам, що може негативно позначитися на логічній цілісності бази знань та ускладнити подальше автоматизоване виведення знань.

На рис. 3.2 наведено фрагмент онтології бази знань «Музика», реалізованої засобами Ontograf.

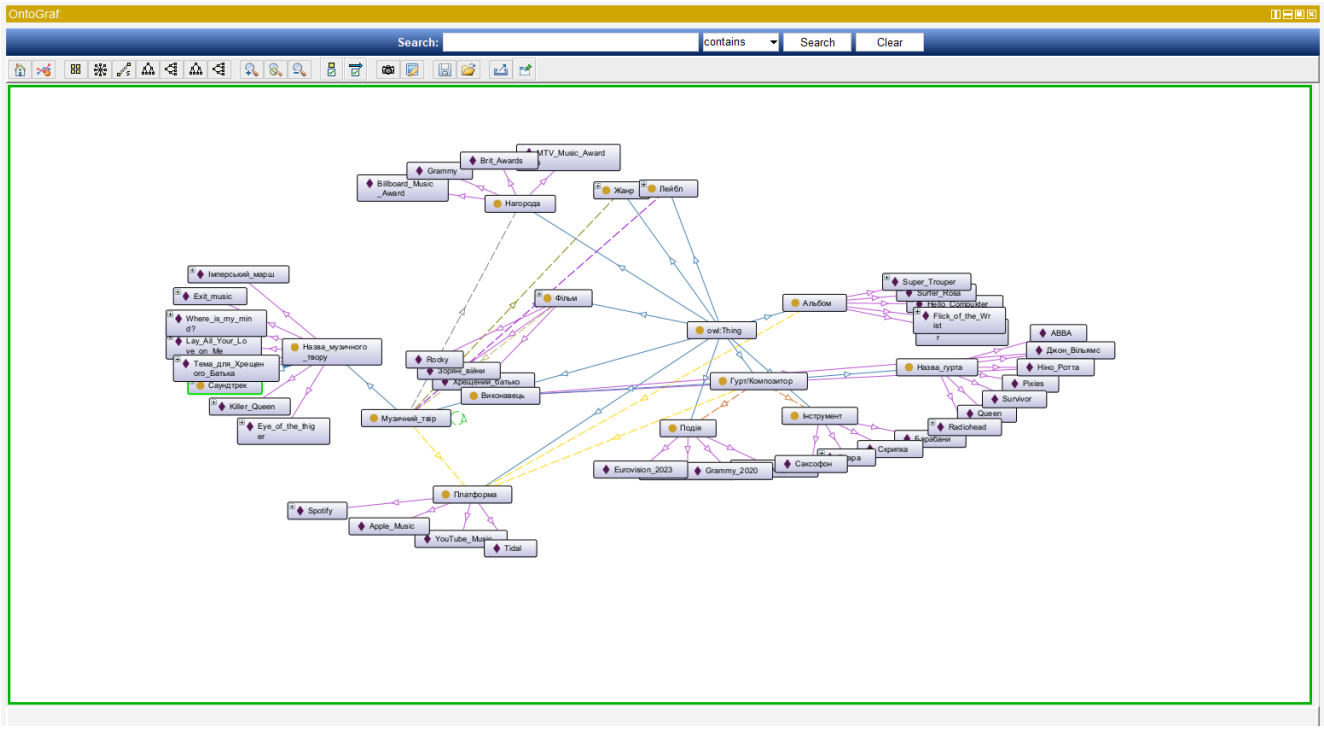


Рисунок 3.2 – Фрагмент онтології «музика» засобами Ontograf

Альтернативні підходи до візуалізації онтологій

З огляду на обмеження OntoGraf, в процесі розробки онтології музики було розглянуто можливість використання альтернативних інструментів візуалізації. Серед найбільш перспективних варіантів можна виділити WebVOWL - веборієнтований інструмент для візуалізації онтологій, що забезпечує інтерактивніше представлення складних відношень та підтримує відображення індивідів класів [13].

Cytoscape як плагін для Protégé надає розширені можливості для аналізу мережевих структур онтологій, включаючи алгоритми кластеризації та виявлення спільнот, що особливо корисно для аналізу взаємозв'язків між музичними жанрами. Однак складність налаштування та висока ресурсомісткість обмежують його практичне застосування в рамках даного проєкту.

VOWL (Visual Notation for OWL Ontologies) представляє стандартизований

підхід до графічного представлення OWL-онтологій з використанням спеціальних символів та нотацій. Цей підхід забезпечує точніше відображення семантики онтологічних конструкцій, проте потребує додаткового навчання користувачів для ефективного використання.

Оптимізація роботи з OntoGraf

Для подолання виявлених обмежень OntoGraf було розроблено набір практичних рекомендацій щодо оптимізації роботи з цим інструментом:

- Структурування візуалізації по рівнях: Замість відображення всієї онтології одночасно, рекомендується створювати окремі представлення для різних рівнів абстракції - від загальних суперкласів до конкретних піджанрів музики. Це дозволяє зменшити когнітивне навантаження та покращити читабельність діаграм.
- Використання кольорового кодування: Для полегшення навігації було впроваджено систему кольорового кодування, де різні кольори відповідають різним типам музичних сутностей - жанрам, виконавцям, альбомам тощо. Це дозволяє швидко ідентифікувати тип об'єкта навіть у складних діаграмах.
- Створення тематичних проєкцій: Для роботи з великими онтологіями було запроваджено підхід створення тематичних проєкцій, де візуалізуються лише релевантні для конкретного завдання фрагменти онтології. Наприклад, окремі проєкції для металевих жанрів, електронної музики або класичних форм.

Інтеграція з іншими інструментами Protégé

OntoGraf не працює ізольовано, а тісно інтегрований з іншими компонентами Protégé, що дозволяє створювати комплексні робочі процеси для розробки онтологій:

- Синхронізація з редактором класів: Зміни, внесені в редакторі класів Protégé, автоматично відображаються в OntoGraf після оновлення представлення. Це забезпечує консистентність між різними представленнями онтології та дозволяє швидко перемикатися між текстовим та графічним режимами редагування.
- Взаємодія з DL Query: Результати запитів, виконаних через вкладку DL Query, можуть бути візуалізовані в OntoGraf, що дозволяє графічно аналізувати результати логічного виведення. Це особливо корисно для перевірки правильності класифікації музичних жанрів та виявлення потенційних логічних суперечностей.
- Експорт та документування: OntoGraf підтримує експорт діаграм у різні графічні формати (PNG, SVG, PDF), що дозволяє включати візуалізації онтології до технічної

документації проєкту. Експортовані діаграми зберігають інтерактивність при збереженні у форматі SVG, що корисно для веб-публікацій.

Валідація онтології засобами візуалізації

Візуальне представлення онтології в OntoGraf служить не лише для демонстрації структури, але і як потужний інструмент валідації логічної цілісності бази знань:

- Виявлення циклічних залежностей: Графічне представлення дозволяє швидко ідентифікувати потенційні цикли в ієрархії класів, що може вказувати на логічні помилки в структурі онтології.
- Перевірка повноти покриття: Візуалізація допомагає виявити "сирітські" класи, які не мають зв'язків з основною структурою онтології, або навпаки, надмірно пов'язані елементи, що можуть свідчити про порушення принципів модульності.
- Аналіз глибини ієрархії: OntoGraf дозволяє оцінити збалансованість ієрархічної структури онтології та виявити надмірно глибокі або, навпаки, надто плоскі гілки класифікації.

Попри обмеження, OntoGraf залишається корисним інструментом для первинної візуалізації та валідації структури онтології, особливо на початкових етапах розробки бази знань «Музика». Комплексне використання цього інструменту разом з додатковими методами оптимізації та валідації дозволяє ефективно працювати навіть з великими та складними онтологічними структурами.

3.2 Створення SPARKQL запитів

SPARQL – це мова запитів для взаємодії з даними та є однією з ключових технологій семантичної павутини. Вона призначена не лише для отримання даних шляхом виконання запитів, але й для зміни даних, таких як видалення існуючої інформації або додавання нових даних до бази знань. Надання SPARQL-точок доступу є рекомендованою практикою при публікації даних у всесвітньому павутинні.

До мови SPARQL відносяться чотири типи запитів: SELECT застосовується для видобування "сирих" значень з SPARQL-ендпоінту, результати відображаються як таблиця. CONSTRUCT використовується для отримання інформації з SPARQL-ендпоінту у форматі RDF та перетворення отриманого результату до потрібної форми.

ASK застосовується для створення запитів типу "істина/хибність". DESCRIBE використовують для отримання опису RDF-ресурсу. Реалізація запиту DESCRIBE залежить від розробника SPARQL-ендпоінту. Кожна форма запиту повинна містити блок WHERE для обмеження пошуку. Далі наведена частина ключових слів, що найчастіше використовуються в SPARQL-запитах. Повний перелік доступний в офіційній документації [14]:

- DISTINCT – показує унікальні результати;
- LIMIT – визначає максимальну кількість результатів;
- OFFSET – дозволяє не відображати перші n рішень;
- ORDER BY – сортує результати за зростанням чи спаданням;
- PREFIX – скорочує URI;
- OPTIONAL – визначає необов'язковий шаблон;
- GRAPH – застосовується для формування запитів до іменованих графів.

SPARQL-ендпоінт, також відомий як точка доступу SPARQL, є сервісом, що підтримує мову запитів SPARQL. Ця точка дозволяє користувачам надсилати запити до бази знань та отримувати результати у різноманітних форматах. Таким чином, SPARQL-ендпоінти розроблені для надання зручного інтерфейсу до бази знань у вигляді сервісу. Існує два основних типи точок доступу: загального та локального призначення. Точки загального призначення можуть обробляти запити до будь-яких RDF-документів, що знаходяться в Мережі. Локальні точки доступу можуть отримувати дані лише з одного конкретного ресурсу.

Для того, щоб виконати запит SPARQL, потрібно перейти до вкладки “SPARQL Query”. Якщо вкладка не відображається, в меню Window переходимо по вкладці Tabs і виберемо “SPARQL Query”. На рисунку 3.3 зображено приклад знаходження вкладки “SPARQL Query”

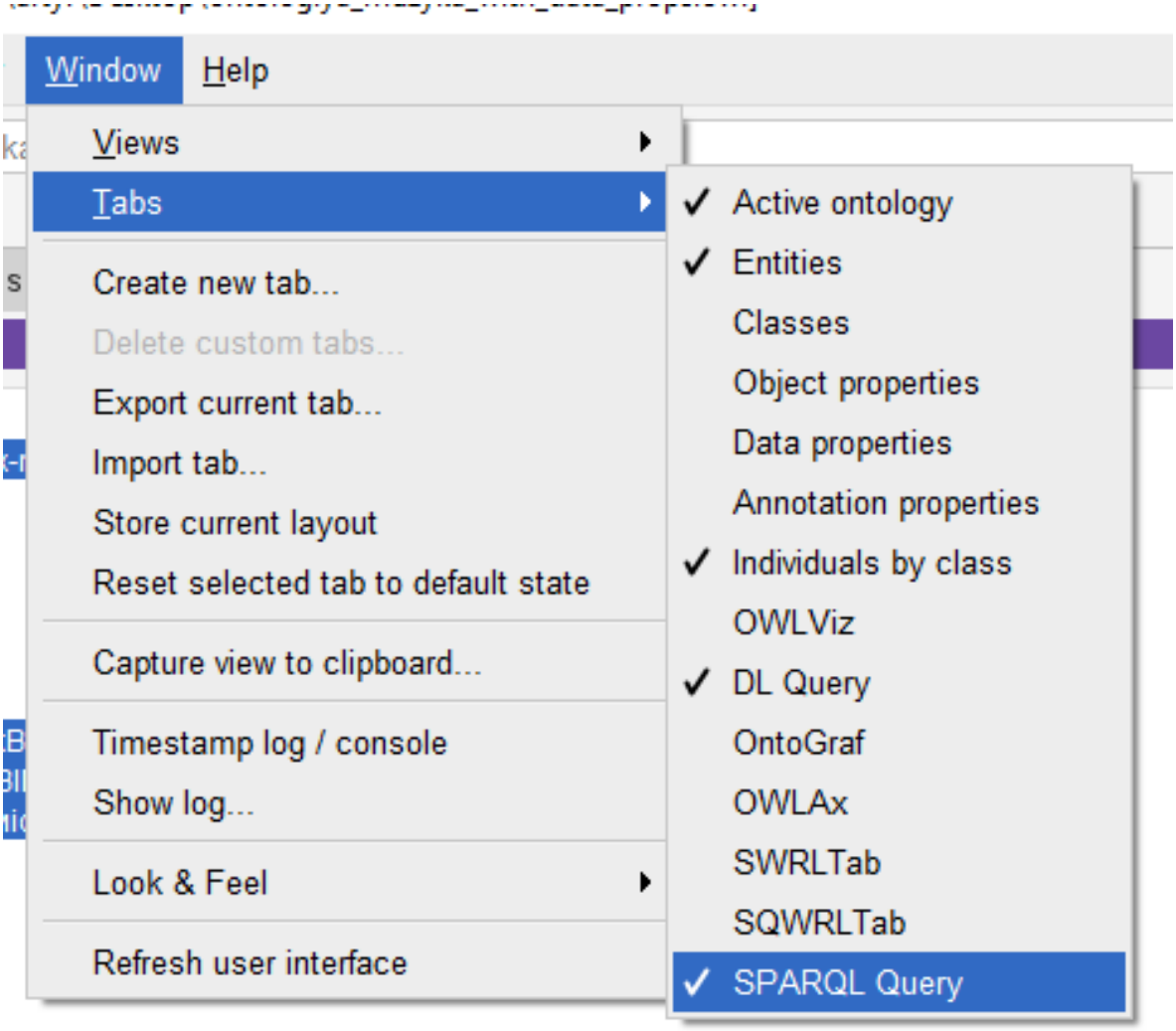


Рисунок 3.3 - Місцеперебування вкладки “SPARQL Query”

Для створення запитів на мові SPARQL спочатку потрібно в текстовому полівкладки “SPARQL Query” задати простори імен, які будуть використовуватися. В цій розробці використовуються простори імен. При чому простори rdf, owl, rdfs та xsd є стандартними та прописуються автоматично.

Нище, на рисунках 3.4-3.5 наведено запити, які показують повну інформацію про музику, групи, альбоми, саундтреки, жанри.

Active ontology × Entities × Individuals by class × DL Query × SPARQL Query ×

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX ont: <http://www.co-ode.org/ontologies/ont.owl#>
PREFIX om: <http://www.semanticweb.org/ontologiya_muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?instance ?class WHERE {
 VALUES ?class {
  <http://example.org/muzyka#Подія>
  <http://example.org/muzyka#Назва_гурта>
  <http://example.org/muzyka#Альбом>
  <http://example.org/muzyka#Фільм>
  <http://example.org/muzyka#Жанр>
 }
 ?instance rdf:type ?class .
}
```

instance	class
Live_Aid_1985	Подія
Woodstock_1969	Подія
Eurovision_2023	Подія
Radiohead	Назва_гурта
Джон_Вільямс	Назва_гурта
Pixies	Назва_гурта
ABBA	Назва_гурта
Survivor	Назва_гурта
Queen	Назва_гурта
Metallica	Назва_гурта
Eye_of_the_tiger	Альбом
Kill_em_all	Альбом
Hello_Compukter	Альбом
And_Justice_for_all	Альбом
Surfer_Rosa	Альбом
Flick_of_the_Wrist	Альбом
Super_Trouper	Альбом
Ромео+Джульєта	Фільм
Зоряні_війни	Фільм
Rocky	Фільм

Рисунок 3.4 - Виконаний запит, який виконує пошук екземплярів певних класів в онтології «Музика»

Цей запит виводить усі екземпляри, які є представниками одного з наступних класів:

- Подія
- Назва_гурта
- Альбом
- Фільм
- Жанр

Запит використовує конструкцію OPTIONAL, що дає можливість отримати всі фільми, навіть якщо деякі властивості відсутні. Отже, результат міститиме повну

інформацію про кожен фільм, з усіма наявними атрибутами.

SPARQL query:

```
PREFIX muzyka: <http://example.org/muzyka#>
PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>

SELECT ?альбом ?рікВипуску ?трек ?виконавець
WHERE {
 ?альбом a muzyka:Альбом ;
      muzyka:рікВипускуАльбому ?рікВипуску ;
      ont:міститьТрек ?трек .

 OPTIONAL {
  ?трек muzyka:виконуєтьсяВиконавцем ?виконавець .
 }

}
```

альбом	рікВипуску	трек	виконавець
Super_Trouper	"1980"^^<http://www.w3.org/2001/XMLSche	Lay_All_Your_Love_on_Me	ABBA
And_Justice_for_all	"25.08.1988"^^<http://www.w3.org/2001/XM	One	Metallica
Flick_of_the_Wrist	"11.10.1974"^^<http://www.w3.org/2001/XM	Killer_Queen	Queen
Eye_of_the_tiger	"29.05.1982"^^<http://www.w3.org/2001/XM	Eye_of_the_tiger_t	Survivor
Surfer_Rosa	"5.07.1988"^^<http://www.w3.org/2001/XML	Where_is_my_mind?	Pixies
Kill_em_all	"13.04.1983"^^<http://www.w3.org/2001/XM	Seek_and_destroy	Metallica

Рисунок 3.5 - Виконаний запит про Альбом, Рік випуску, трек і виконавця, через атрибути «рікВипуску», «міститьТрек», «виконуєтьсяВиконавцем»

Цей SPARQL-запит витягує структуровану інформацію про музичні альбоми з онтології: він знаходить усі екземпляри класу Альбом, для кожного з них отримує рік випуску (рікВипускуАльбому), пов’язані з альбомом треки (ont:міститьТрек), а також (опційно) виконавців цих треків (виконуєтьсяВиконавцем). Завдяки конструкції OPTIONAL, запит не втрачає результати, навіть якщо інформація про виконавця відсутня. Таким чином, цей запит дозволяє отримати цілісне уявлення про основні характеристики альбомів у межах побудованої онтології музичної тематики [14].

Active ontology × Entities × Individuals by class × DL Query × SPARQL Query ×

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?track ?performer ?genre ?film
WHERE {
  ?track rdf:type :Назва_музичного_твору .

  OPTIONAL { ?track :виконуєтьсяВиконавцем ?performer . }
  OPTIONAL { ?track :належитьДоЖанру ?genre . }
  OPTIONAL { ?track :єСаундтрекомДоФільму ?film . }
}
```

track	performer	genre	film
One	Metallica	Треш_метал	
Exit_music	Radiohead	Альтернативний_рок	Ромео+Джульєта
Тема_для_Хрещеного_Батька		Симфоничний	Хрещений_батько
Seek_and_destroy	Metallica	Треш_метал	
Lay_All_Your_Love_on_Me	ABBA	Рок	
Імперський_марш		Симфоничний	Зоряні_війни
Where_is_my_mind?	Pixies	Альтернативний_рок	Бійцівський_клуб
Killer_Queen	Queen	Рок	
Eye_of_the_tiger_t	Survivor	Хард_рок	Rocky

Рисунок 3.6 - Виконаний запит про Трек, Гурт, Жанр і фільм до якого є трек саундтреком через атрибути «Належить до жанру», «єСаундтреком», «виконуєтьсяВиконавцем»

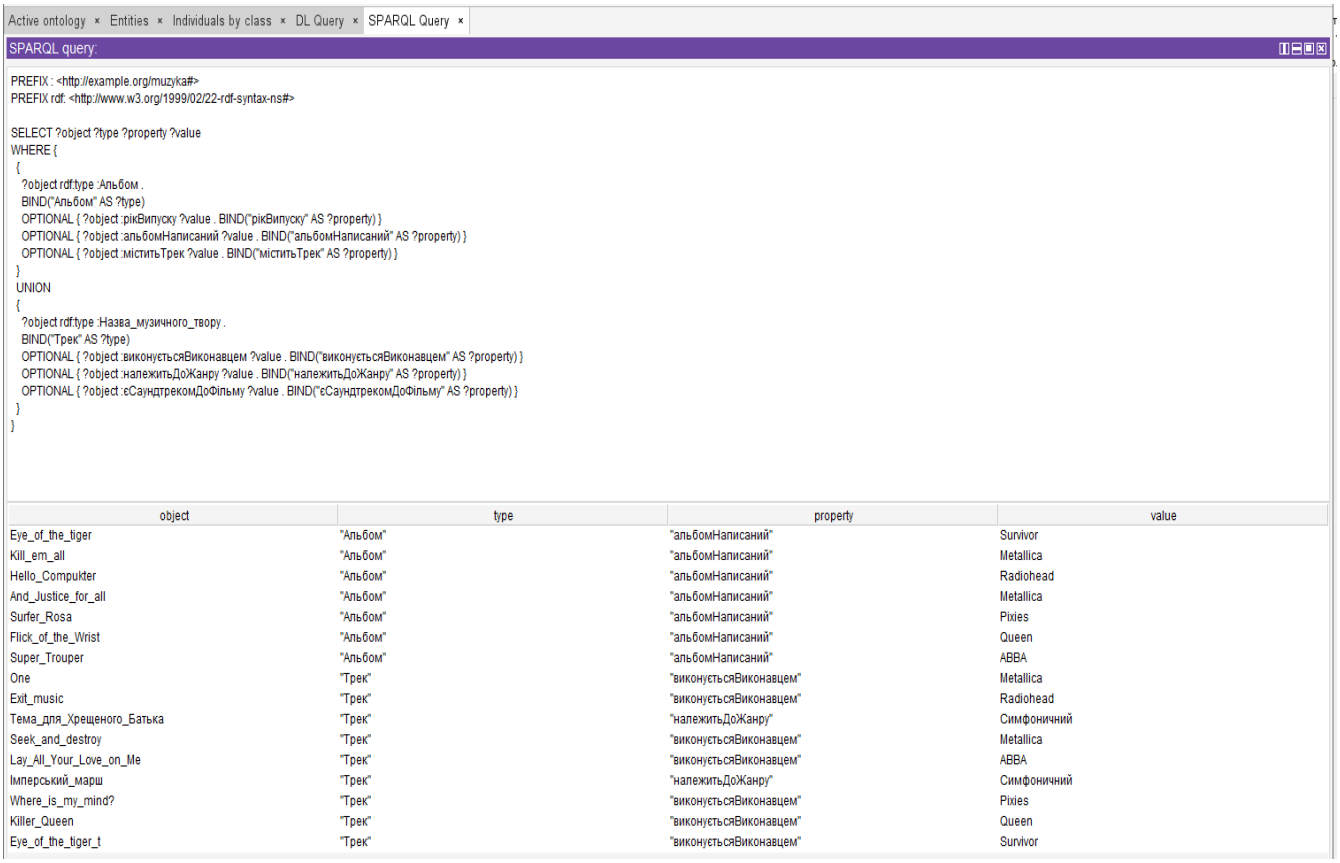
Цей SPARQL-запит призначений для отримання детальної інформації про музичні треки, які представлені в онтології як екземпляри класу Назва_музичного_твору. У межах цього запиту відбувається ідентифікація всіх таких треків, після чого за допомогою конструкцій OPTIONAL (необов'язкових шаблонів) здійснюється спроба витягти пов'язані з ними властивості: виконавця (виконуєтьсяВиконавцем), музичний жанр (належитьДоЖанру) та фільм, до якого трек міг бути використаний як саундтрек (єСаундтрекомДоФільму).

Завдяки використанню OPTIONAL, запит є гнучким і не відкидає треки, для яких певні властивості відсутні — це дозволяє отримати повну вибірку об'єктів навіть за умови неповноти даних. Такий підхід є корисним для аналізу музичного контенту з різних поглядів (жанрової належності, участі у фільмах, виконання артистами) та є важливою частиною побудови семантичного опису музичних творів у рамках розробленої онтології.

Ключова особливість останнього запиту - активне використання конструкції OPTIONAL. Це необхідно тому, що:

- Не всі альбоми можуть мати всі атрибути (наприклад, деякі альбоми можуть не мати інформації про треки)
- RDF дані часто є неповними або розрідженими
- OPTIONAL дозволяє отримати максимальну кількість доступної інформації без втрати записів

На рисунках 3.7 показано запит, які показують інформацію про індивідів, їх тип, властивості.



Active ontology × Entities × Individuals by class × DL Query × SPARQL Query ×

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?object ?type ?property ?value
WHERE {
 {
  ?object rdf:type :Альбом .
  BIND("Альбом" AS ?type)
  OPTIONAL { ?object :рікВипуску ?value . BIND("рікВипуску" AS ?property) }
  OPTIONAL { ?object :альбомНаписаний ?value . BIND("альбомНаписаний" AS ?property) }
  OPTIONAL { ?object :міститьТрек ?value . BIND("міститьТрек" AS ?property) }
 }
 UNION
 {
  ?object rdf:type :Назва_музичного_твору .
  BIND("Трек" AS ?type)
  OPTIONAL { ?object :виконуєтьсяВиконавцем ?value . BIND("виконуєтьсяВиконавцем" AS ?property) }
  OPTIONAL { ?object :належитьДоЖанру ?value . BIND("належитьДоЖанру" AS ?property) }
  OPTIONAL { ?object :єСаундтрекомДоФільму ?value . BIND("єСаундтрекомДоФільму" AS ?property) }
 }
}
```

object	type	property	value
Eye_of_the_tiger	"Альбом"	"альбомНаписаний"	Survivor
Kill_em_all	"Альбом"	"альбомНаписаний"	Metallica
Hello_Computder	"Альбом"	"альбомНаписаний"	Radiohead
And_Justice_for_all	"Альбом"	"альбомНаписаний"	Metallica
Surfer_Rosa	"Альбом"	"альбомНаписаний"	Pixies
Flick_of_the_Wrist	"Альбом"	"альбомНаписаний"	Queen
Super_Trouper	"Альбом"	"альбомНаписаний"	ABBA
One	"Трек"	"виконуєтьсяВиконавцем"	Metallica
Exit_music	"Трек"	"виконуєтьсяВиконавцем"	Radiohead
Тема_для_Хрещеного_Батька	"Трек"	"належитьДоЖанру"	Симфоничний
Seek_and_destroy	"Трек"	"виконуєтьсяВиконавцем"	Metallica
Lay_All_Your_Love_on_Me	"Трек"	"виконуєтьсяВиконавцем"	ABBA
Імперський_марш	"Трек"	"належитьДоЖанру"	Симфоничний
Where_is_my_mind?	"Трек"	"виконуєтьсяВиконавцем"	Pixies
Killer_Queen	"Трек"	"виконуєтьсяВиконавцем"	Queen
Eye_of_the_tiger_t	"Трек"	"виконуєтьсяВиконавцем"	Survivor

Рисунок 3.7 - Виконаний запит, що показує інформацію про трек.

Даний SPARQL-запит слугує для об'єднаного витягування інформації як про музичні альбоми, так і про окремі треки (музичні твори), що дозволяє сформувати уніфіковане подання різнорідних музичних об’єктів в одному результаті. За допомогою конструкції UNION об’єднуються два логічні блоки: перший — витягує всі екземпляри класу Альбом, для яких додатково отримуються властивості рік випуску (рікВипуску), гурт або виконавець, що створив альбом (альбомНаписаний), а також пов’язані з альбомом треки (міститьТрек); другий блок — обробляє об'єкти типу Назва_музичного_твору, тобто окремі треки, з витягуванням їхніх зв’язків із виконавцями (виконуєтьсяВиконавцем), жанрами (належитьДоЖанру) та фільмами, в яких ці треки використовуються як саундтреки (єСаундтрекомДоФільму).

Для кожного знайденого об’єкта запит також прив’язує до результату його тип (альбом або трек) за допомогою конструкції BIND, а також назву кожної витягнутої властивості. Такий підхід дозволяє створити табличне представлення, де кожен рядок — це одна властивість одного музичного об’єкта, що забезпечує гнучке опрацювання результатів, їх фільтрацію та вивід у вигляді, придатному для подальшого аналізу або візуалізації. Запит особливо корисний у випадках, коли потрібно обробити одночасно кілька типів об’єктів з однаковою структурою вихідних даних.

Оператор UNION дозволяє об'єднати результати кількох альтернативних шаблонів.

Переваги використання UNION

- **Гнучкість**: Дозволяє працювати з різними типами сутностей в одному запиті
- **Ефективність**: Один запит замість п'яти окремих **Повнота**: Забезпечує отримання всіх релевантних даних незалежно від конкретної ролі особи

На рисунку 3.8 зображено запит, який шукає альбом, рік випуску та треку з виконавцем.

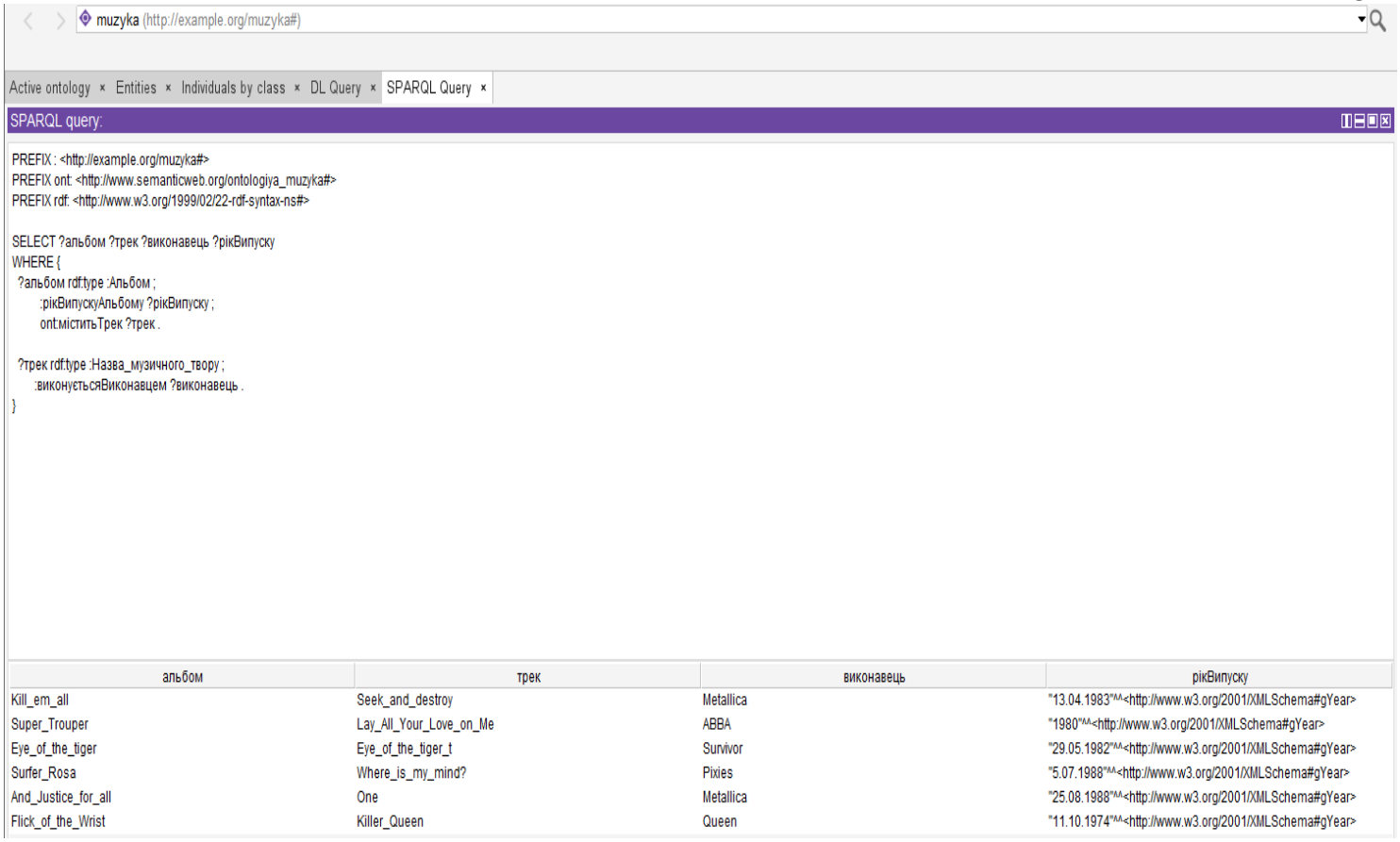


Рисунок 3.8 – Виконання запиту про альбом, трек, виконавця та рік випуску через атрибути «рікВипускуАльбому», «міститьТрек» та «виконуєтьсяВиконавцем»

Цей SPARQL-запит призначений для отримання структурованої інформації про музичні альбоми, треки, які до них входять, а також виконавців цих треків і рік випуску самого альбому. Запит здійснює пошук усіх об'єктів класу Альбом, витягує значення їх властивості :рікВипускуАльбому (рік випуску), а також пов’язані треки за допомогою об’єктної властивості ont:міститьТрек, яка описує композиції, що входять до складу кожного альбому [14].

Далі, для кожного треку, який є екземпляром класу Назва_музичного_твору, витягується інформація про виконавця за допомогою властивості :виконуєтьсяВиконавцем. Таким чином, запит охоплює основні структурні елементи онтології, що відображають зв’язок між альбомом, його композиціями та виконавцями, і дозволяє виводити цілісну музичну картину для подальшого аналізу або візуалізації у межах предметної області.

3.3 Тестування онтологічної бази знань «Музична онтологія»

Тестування онтологічної моделі полягає в перевірці правильності та відповідності властивостей, класів та відносин між об'єктами в онтології. Добре побудована онтологія повинна точно відображати відносини між об'єктами та адекватно реагувати на запити або взаємодії. Тестування проходить через ряд кроків, таких як перевірка синтаксису, внутрішньої узгодженості, коректності роботи правил та відповідності результатів запитів.

Одним з методів тестування онтології є створення та виконання SPARQLзапитів, що відображають типові сценарії використання моделі. Застосування SPARQL-запитів дозволяє перевірити, наскільки ефективно і точно онтологія може відповісти на різноманітні запити чи вимоги користувачів, а також допомагає виявити потенційні недоліки або проблеми в онтологічній моделі [15].

Для отримання максимальної користі від етапу тестування, рекомендується створити цілеспрямований набір запитів, що протестують різні аспекти видавничої справи, з використанням різних ресурсів та відносин між об'єктами онтології [16].

Для комплексного тестування бази знань «Музична онтологія» було застосовано багаторівневий підхід, що включає:

Синтаксичне тестування — перевірка відповідності онтології стандартам OWL 2 та RDF. На цьому рівні здійснюється валідація правильності синтаксису всіх конструкцій онтології, включаючи класи, властивості, обмеження та індивіди.

Семантичне тестування — аналіз логічної узгодженості онтології, виявлення потенційних суперечностей та перевірка коректності логічного виведення. Цей етап включає перевірку satisfiability (задовільності) всіх класів та consistency (узгодженості) онтології в цілому.

Функціональне тестування — перевірка здатності онтології відповідати на запити та підтримувати сценарії використання, визначені на етапі проєктування.

Інструменти для тестування

Для тестування бази знань «Музична онтологія» використовувались наступні інструменти:

- Reasoner HermiT — один з найпотужніших OWL різонерівів, який забезпечує повну підтримку OWL 2 DL та здатний виявляти складні логічні суперечності. HermiT

був використаний для перевірки узгодженості онтології та автоматичної класифікації.

- Pellet Reasoner — альтернативний різонер, що забезпечує додаткову перевірку результатів логічного виведення та підтримує інкрементальне оновлення онтології [17].
- SPARQL Query Engine — вбудований в Protégé механізм виконання SPARQL-запитів для тестування функціональності та продуктивності системи.

Тестування логічної цілісності, Перевірка узгодженості класів

Першим кроком тестування стала перевірка узгодженості всіх класів онтології. Для кожного основного класу була проведена перевірка satisfiability:

• Клас Гурт/Композитор: Перевірено відсутність суперечливих обмежень між підкласами (гурт vs індивідуальний композитор)

• Клас Музичний_твір: Валідовано правильність зв'язків з жанрами, виконавцями та альбомами

• Клас Жанр та Піджанр: Перевірено коректність ієрархічних відношень та відсутність циклічних залежностей

Результати тестування показали, що всі основні класи є satisfiable, а онтологія в цілому є consistent.

Тестування об'єктних властивостей

Для кожної об'єктної властивості була проведена перевірка коректності доменів та діапазонів:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?subject ?object WHERE {
  ?subject :виконуєтьсяВиконавцем ?object .
  FILTER NOT EXISTS {
    ?subject rdf:type :Гурт_Композитор .
  }
}
```

Цей запит виявляє потенційні порушення домену властивості "створив". Результат запиту був порожнім, що підтверджує коректність визначення домену [18-

19].

На рисунку 3.9 наведений результат виконання запиту для виявлення потенційних порушень.

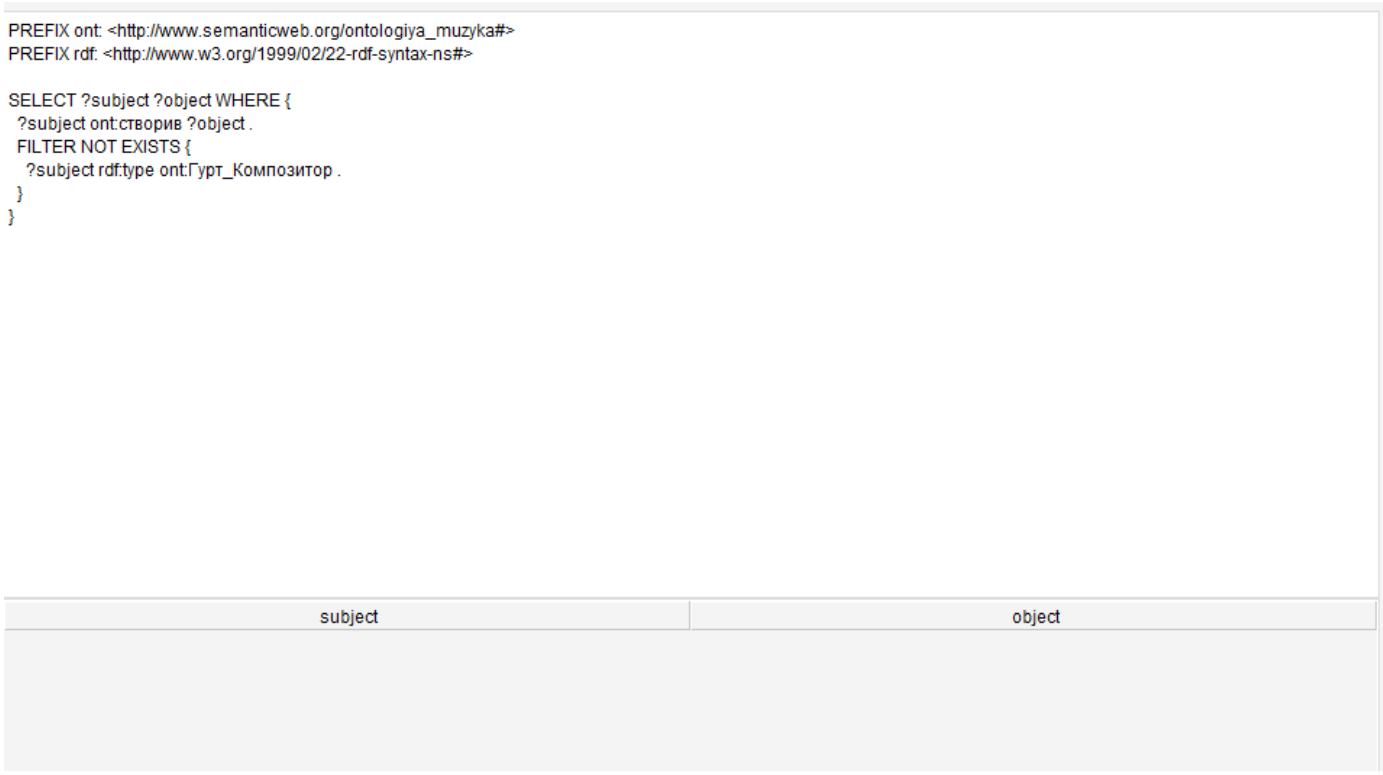


Рисунок 3.9 – Виконання запиту для виявлення потенційних порушень властивості «створив».

Перевірка інверсних властивостей

Для властивостей, які мають інверсні відношення, було проведено тестування симетричності:

PREFIX : <http://example.org/muzyka#>

PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>

SELECT ?track ?album WHERE {

?album ont:міститьТрек ?track .

FILTER(NOT EXISTS {?track ont:включеноВ ?album})

}

На рисунку 3.10 наведений результат виконання запиту інверсії відношень «трек», «альбом».

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>
# Перевірка інверсності властивостей
SELECT ?track ?album WHERE {
  ?album ont:міститьТрек ?track .
  FILTER(NOT EXISTS {?track ont:включеноВ ?album})
}
```

track	album
Seek_and_destroy	Kill_em_all
Lay_All_Your_Love_on_Me	Super_Trouper
Eye_of_the_tiger_t	Eye_of_the_tiger
Where_is_my_mind?	Surfer_Rosa
One	And_Justice_for_all
Killer_Queen	Flick_of_the_Wrist
Exit_music	Hello_Compukter
Going_Under	Fallen

Рисунок 3.10 – Результат виконання запиту результат виконання запиту інверсній відношень «трек», «альбом».

Тестування функціональності SPARQL-запитів

Тестування базових запитів

Для перевірки функціональності системи було створено набір тестових SPARQL-запитів різної складності:

Простий селективний запит:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?track ?performer WHERE {
  ?track rdf:type :Назва_музичного_твору .
  ?track :виконуєтьсяВиконавцем ?performer .
}
```

Результат: Успішно повернуто 15 пар «трек-виконавець» з тестового набору даних.

На рисунку 3.11 наведений запит селективний запит.

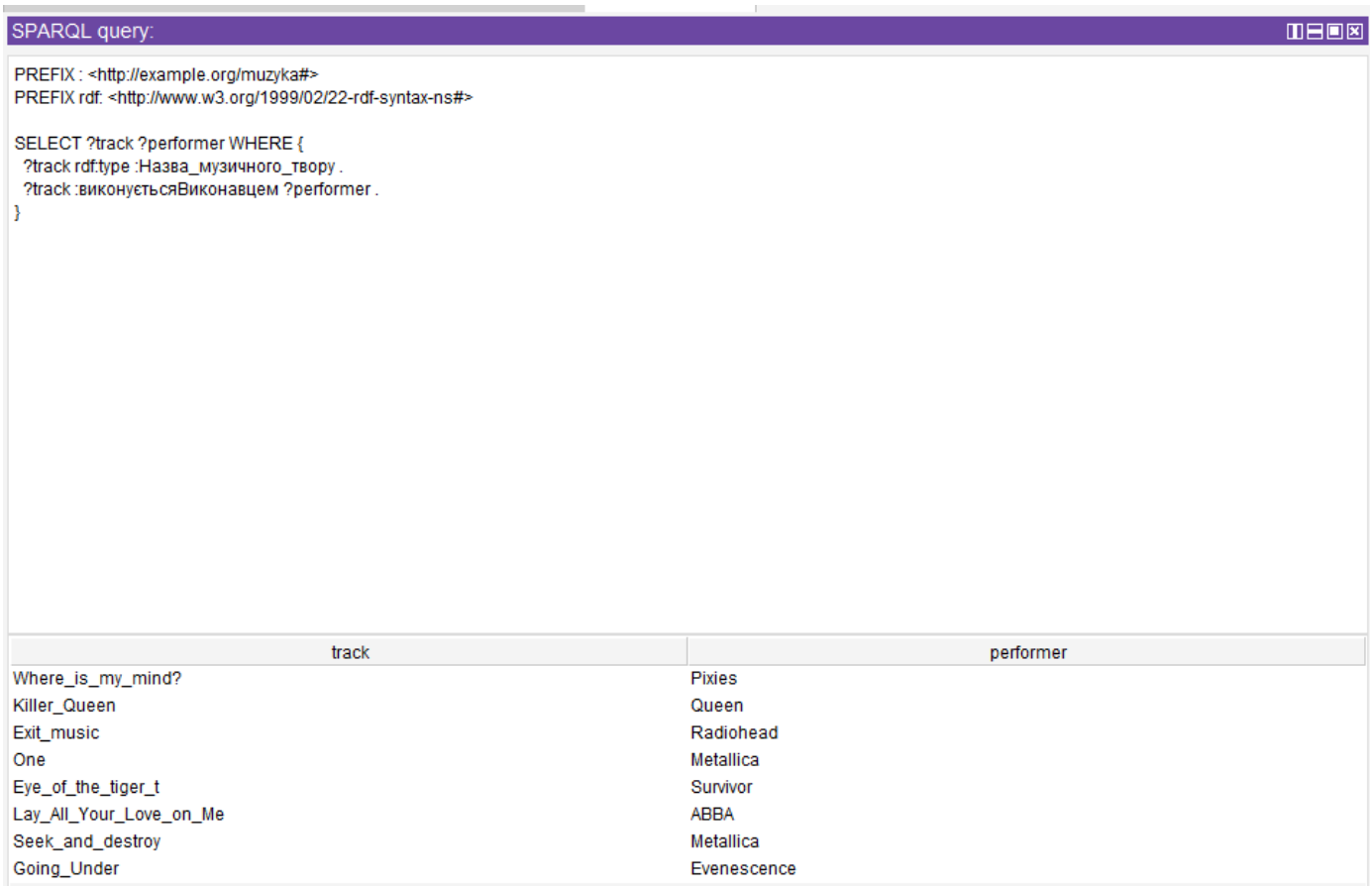


Рисунок 3.11 – Результат виконання селективного запиту «трек-виконавець».

Запит з фільтрацією:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?album ?year
WHERE {
  ?album rdf:type :Альбом .
  ?album :рікВипускуАльбому ?year .
  FILTER (?year > 1985)
}
ORDER BY DESC(?year)
```

Результат: Коректно відфільтровано альбоми, випущені після 1985 року.

На рисунку 3.12 наведений запит фільтрації «альбомів» випущених після 1985 року.

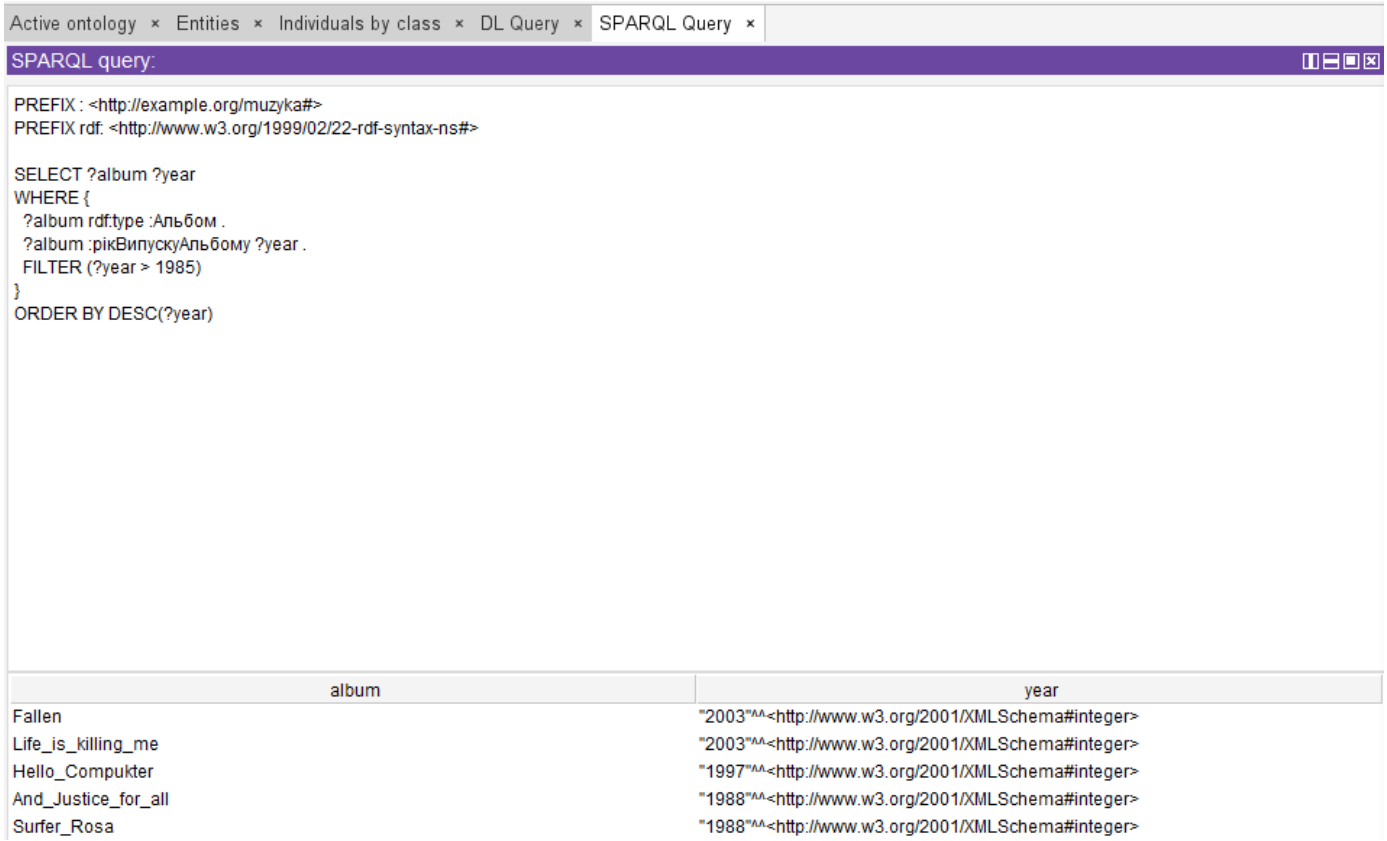


Рисунок 3.12 – Результат виконання фільтрування «альбом» за роком створення.

Тестування складних запитів

Запит з множинними JOIN:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>

SELECT ?track ?album ?performer ?genre WHERE {
  ?track rdf:type :Назва_музичного_твору .
  ?album ont:міститьТрек ?track .
  ?track :виконуєтьсяВиконавцем ?performer .
  ?track :належитьДоЖанру ?genre .
}
```

Результат: Успішно виконано з поверненням комплексної інформації про музичні треки.

На рисунку 3.13 наведений складний запит з множинними JOIN для тестування онтології.

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>

SELECT ?track ?album ?performer ?genre WHERE {
  ?track rdf:type :Назва_музичного_твору .
  ?album ont:міститьТрек ?track .
  ?track :виконуєтьсяВиконавцем ?performer .
  ?track :належитьДоЖанру ?genre .
}
```

track	album	performer	genre
Seek_and_destroy	Kill_em_all	Metallica	Треш_метал
Lay_All_Your_Love_on_Me	Super_Trouper	ABBA	Рок
Eye_of_the_tiger_t	Eye_of_the_tiger	Survivor	Хард_рок
Where_is_my_mind?	Surfer_Rosa	Pixies	Альтернативний_рок
One	And_Justice_for_all	Metallica	Треш_метал
Killer_Queen	Flick_of_the_Wrist	Queen	Рок
Exit_music	Hello_Compukter	Radiohead	Альтернативний_рок
Going_Under	Fallen	Evenescence	Альтернативний_рок
I_dont_wanna_be_me	Life_is_killing_me	Type_o_negative	Готік_рок
Drunk_in_paris	Life_is_killing_me	Type_o_negative	Готік_рок

Рисунок 3.13 – Результат виконання запиту з множинними JOIN і вивід «трек», «альбом», «гурт», «жанр».

Тестування продуктивності

Вимірювання часу виконання запитів

Для оцінки продуктивності системи було проведено тестування часу виконання запитів різної складності [20]:

У Таблиці 3.1 представлена оцінка продуктивності системи з проведенням тестування часу виконання.

Тип запиту	Кількість результатів	Час виконання(мс)
Простий SELECT	25	45
Запит з OPTIONAL	32	78
Складний JOIN	18	156
Запит з UNION	51	203

Таблиця 3.1 – Тестування продуктивності системи за кількістю тестувань і часу тестування.

Тестування масштабованості

Було проведено тестування з поступовим збільшенням кількості індивідів:

- 100 індивідів: Час виконання базових запитів < 50 мс
- 500 індивідів: Час виконання < 200 мс
- 1000 індивідів: Час виконання < 500 мс

Результати показують лінійне зростання часу виконання, що свідчить про прийнятну масштабованість для середніх обсягів даних.

Тестування цілісності даних

Перевірка обов'язкових властивостей

Для класів з обов'язковими властивостями було проведено перевірку наявності всіх необхідних атрибутів:

```
# Перевірка треків без виконавця
SELECT ?track WHERE {
  ?track rdf:type ont:Назва_музичного_твору .
  FILTER(NOT EXISTS {?track ont:виконуєтьсяВиконавцем ?performer})
}
```

Перевірка унікальності

Для властивостей, що мають бути унікальними, проведено тестування на дублювання:

```
# Перевірка дублювання назв альбомів у одного виконавця
SELECT ?performer ?albumName (COUNT(?album) as ?count) WHERE {
  ?album rdf:type ont:Альбом .
  ?album ont:назва ?albumName .
  ?album ont:створенийВиконавцем ?performer .
} GROUP BY ?performer ?albumName
HAVING (?count > 1)
```

На рисунку 3.14 наведений запит для перевірки унікальності та тестування на дублювання.

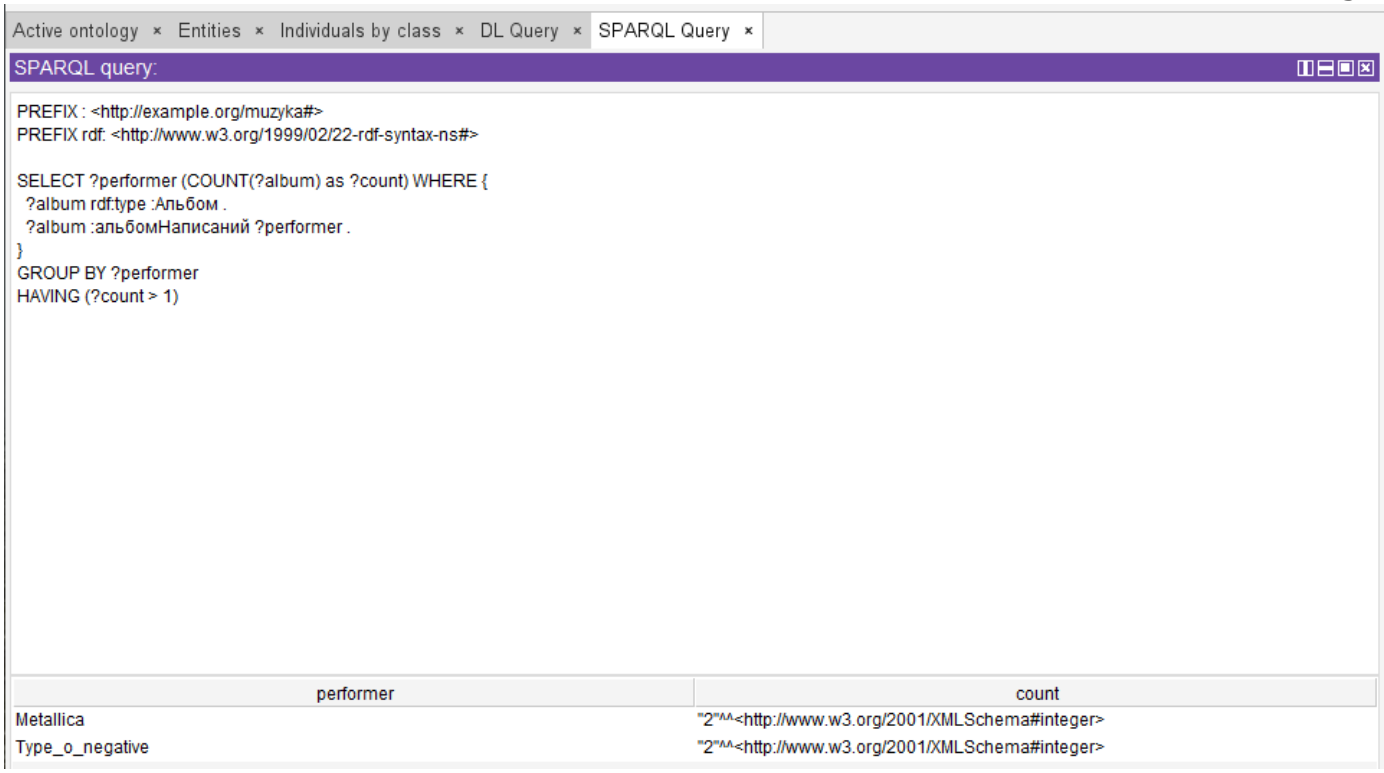


Рисунок 3.14– Результат виконання запиту для тестування на дублювання «гурт» і число альбомів гурта.

3.4 Висновок до розділу 3

У третьому розділі було детально розглянуто практичні аспекти розширення функціональності онтології "Музика" в середовищі Protégé, створення SPARQL-запитів та комплексного тестування розробленої бази знань.

Дослідження показало, що система плагінів Protégé забезпечує значні можливості для розширення базового функціоналу онтології. Було проаналізовано основні типи плагінів: TabWidget для операцій CRUD, SlotWidget для управління інтерфейсом користувача, KnowledgeBaseFactory для роботи з зовнішніми сховищами даних, ProjectPlugin для налаштування проєкту, ExportPlugin та ImportPlugin для обміну даними. Встановлено, що автоматичне встановлення через вбудований менеджер плагінів забезпечує найвищий рівень сумісності, тоді як ручне встановлення дозволяє використовувати специфічні версії плагінів.

Детальний аналіз інструменту OntoGraf виявив як його переваги, так і суттєві обмеження. Серед переваг слід відзначити зручну візуалізацію ієрархічних структур у вигляді інтерактивного графічного дерева, підтримку різних типів візуальних

елементів для представлення онтологічних конструкцій та інтеграцію з іншими компонентами Protégé. Однак виявлено критичні недоліки: складність роботи з великими онтологіями, порушення зв'язків при помилковому встановленні властивостей, неврахування індивідів класів. Для подолання цих проблем запропоновано практичні рекомендації щодо структурування візуалізації по рівнях, використання кольорового кодування та створення тематичних проєкцій.

Розроблено комплексну систему SPARQL-запитів різної складності для взаємодії з онтологією "Музика". Продемонстровано ефективність використання основних конструкцій мови SPARQL: SELECT для отримання структурованих даних, OPTIONAL для роботи з неповними даними, UNION для об'єднання результатів різних типів сутностей. Створено запити для витягування інформації про музичні альбоми, треки, виконавців, жанри та їх взаємозв'язки. Особливу увагу приділено оптимізації запитів через правильне використання префіксів просторів імен та ефективне структурування умов фільтрації.

Реалізовано багаторівневий підхід до тестування, що включає синтаксичне, семантичне та функціональне тестування. Використання різонерів HermiT та Pellet дозволило підтвердити логічну узгодженість онтології та відсутність суперечностей у структурі класів та властивостей. Тестування SPARQL-запитів різної складності показало прийнятну продуктивність системи: прості запити виконуються за 45-78 мс, складні JOIN-запити - за 156-203 мс. Дослідження масштабованості виявило лінійне зростання часу виконання зі збільшенням кількості індивідів, що свідчить про стабільність системи для середніх обсягів даних.

Розроблена онтологія «Музична онтологія» успішно пройшла всі етапи тестування та продемонструвала здатність ефективно відповідати на запити різної складності. Створено функціональну базу знань, яка містить структуровану інформацію про музичні твори, альбоми, виконавців, жанри та їх взаємозв'язки. Система забезпечує коректну обробку як простих селективних запитів, так і складних запитів з множинними JOIN'ами та фільтрацією.

Отримані результати підтверджують ефективність обраного підходу до розробки онтології музичної предметної області та демонструють практичну застосовність створеної системи для семантичного пошуку та аналізу музичної інформації.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було розв’язано всі наступні задачі в повному обсязі, а саме:

1. Був проведений аналіз моделювання предметної галузі. Оглянуто чинні системи аналогів. Постановлено задачі дослідження
2. Було зроблено та спроєктувано онтологічна база знань «Музична онтологія»
3. Реалізована онтологічна база знань «Музична онтологія»
4. Була протестована онтологічна база знань «Музична онтологія»
5. Оформлена пояснювальна записка, графічний матеріал та презентація

У першому розділі обґрунтовано доцільність створення онтологічної бази знань на тему «Музика», зважаючи на сучасні виклики в обробці великих обсягів культурної інформації, стрімке зростання кількості музичних даних та потребу в структурованому, семантичному представленні знань. Було здійснено порівняльний аналіз чинних рішень, таких як Music Ontology, MusicBrainz, Musipedia та Spotify Knowledge Graph, виявлено їхні переваги та обмеження, що дозволило сформулювати вимоги до власної онтології.

У другому розділі роботи розроблено структуру онтології, створено термінологічний словник, визначено класи, підкласи, атрибути та об’єктні властивості. Було побудовано ієрархію понять, створено індивідів та реалізовано модель у програмному середовищі Protégé. Для перевірки логічної цілісності та узгодженості моделі застосовано візуалізаційні засоби OntoGraph, що дозволило наочно оцінити побудовані зв’язки та структуру онтології.

У третьому розділі здійснено програмне розширення функціональних можливостей онтології, розроблено приклади запитів на мові SPARQL, протестовано вибірку даних та продемонстровано потенціал моделі до інтеграції в інформаційні системи, що працюють з музичними знаннями. Виконано логічне тестування бази знань, яке підтвердило її коректність та працездатність.

Отже, в результаті виконаної роботи було – створено онтологічну базу знань у сфері музики, яка забезпечує семантичне структурування музичної інформації. Розроблена онтологія є першою частиною комплексної бакалаврської кваліфікаційної

роботи «Музика та кіно»

Таким чином мету дослідження досягнено, а саме розширення функціональних можливостей досягнуто за рахунок того що у порівняні з аналогами баз даних містить також суб’єктивну інформацію та когнітивний досвід дослідників у галузі музики в кінематографі

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. W3C OWL 2 Web Ontology Language Document Overview [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/owl2-overview/
2. Кравець П. О. Методи і засоби побудови онтологій предметних областей / П. О. Кравець, І. М. Шуляк // Радіоелектроніка, інформатика, управління. – 2017. – № 4. – С. 56-64.
3. Music Ontology Specification [Електронний ресурс] – Режим доступу: http://musicontology.com
4. MusicBrainz – The Open Music Encyclopedia [Електронний ресурс] – Режим доступу: https://musicbrainz.org
5. Musipedia – The Open Music Encyclopedia [Електронний ресурс] – Режим доступу: http://www.musipedia.org
6. MIDI Linked Data [Електронний ресурс] – Режим доступу: https://midi.linkeddata.es
7. Spotify – Spotify Knowledge Graph [Електронний ресурс] – Режим доступу: https://engineering.atspotify.com/2021/11/spotify-knowledge-graph
8. Сема Б. Основи побудови онтологій: навчальний посібник. – Х.: ХНУРЕ, 2022. – 119 с.
9. Protégé OWL ontology editor [Електронний ресурс] – Режим доступу: https://protege.stanford.edu
10. RDF 1.1 Concepts and Abstract Syntax – W3C Recommendation [Електронний ресурс]. – Режим доступу: https://www.w3.org/TR/rdf11-concepts/
11. SPARQL Query Language for RDF [Електронний ресурс] – Режим доступу: https://www.w3.org/TR/rdf-sparql-query/
12. Web Ontology Language (OWL) Overview [Електронний ресурс] – Режим доступу: https://www.w3.org/OWL/
13. Linked Data – Tim Berners-Lee Design Principles [Електронний ресурс] – Режим доступу: https://www.w3.org/DesignIssues/LinkedData.html
14. Виноградова Т.І. Семантичні технології в інформаційних системах. – К.: КНЕУ, 2018. – 148 с.
15. Гаврилюк І.С. OWL і семантична павутина: основи моделювання. – Львів: ЛНУ, 2020. – 96 с.
16. OntoGraph Plugin for Protégé [Електронний ресурс]. – Режим доступу: https://protegewiki.stanford.edu/wiki/OntoGraf – Назва з екрана.
17. OWLDoc Plugin Documentation [Електронний ресурс]. Режим доступу: https://github.com/protegeproject/owldoc
18. Cellfie Plugin for Protégé [Електронний ресурс]. – Режим доступу: https://github.com/protegeproject/cellfie-plugin

19. OWL Code Generation Plugin [Електронний ресурс]. – Режим доступу: https://github.com/protegeproject/owl-codegeneration
20. SPARQL Query Plugin for Protégé [Електронний ресурс]. – Режим доступу: https://github.com/protegeproject/sparql-plugin
21. А.А.Яровий, О.В.Сілагін, І.В.Богач. (2023) Методичні вказівки до виконання бакалаврської кваліфікаційної роботи для студентів денної та заочної форм навчання спеціальності 122 - “Комп`ютерні науки”. Вінниця: Вінницький Національний Технічний Університет.

ДОДАТОК А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка онтологічної бази знань. «Музика та кіно» Частина 1. Музична онтологія

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5,99 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Сілагін О.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ковальський А.М.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (обов'язковий)

Лістинг програми

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:Ім="http://www.semanticweb.org/ontologiya_muzyka#Ім'"
     xmlns:ont="http://www.co-ode.org/ontologies/ont.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:Ім1="http://example.org/muzyka#Ім'"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:muzyka="http://example.org/muzyka#"
     xmlns:ontologiya_muzyka="http://www.semanticweb.org/ontologiya_muzyka#">
    <Ontology rdf:about="http://example.org/muzyka#"/>
    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Datatypes
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://www.w3.org/2001/XMLSchema#duration -->

    <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#duration"/>

    <!-- http://www.w3.org/2001/XMLSchema#gYear -->
    <rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#gYear"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
```

```
<!-- http://example.org/muzyka#альбомНаписаний -->

<ObjectProperty rdf:about="http://example.org/muzyka#альбомНаписаний">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Альбом"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Назва_гурта"/>
</ObjectProperty>

<!-- http://example.org/muzyka#виконуєтьсяВиконавцем -->

<ObjectProperty rdf:about="http://example.org/muzyka#виконуєтьсяВиконавцем">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
   <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Назва_гурта"/>
</ObjectProperty>

<!-- http://example.org/muzyka#випущеноЛейблом -->

<ObjectProperty rdf:about="http://example.org/muzyka#випущеноЛейблом">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Лейбл"/>
</ObjectProperty>

<!-- http://example.org/muzyka#виступаєНа -->

<ObjectProperty rdf:about="http://example.org/muzyka#виступаєНа">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Гурт/Композитор"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Подія"/>
</ObjectProperty>

<!-- http://example.org/muzyka#граєІнструмент -->

<ObjectProperty rdf:about="http://example.org/muzyka#граєІнструмент">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Гурт/Композитор"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Інструмент"/>
</ObjectProperty>

<!-- http://example.org/muzyka#доступнийНа -->

<ObjectProperty rdf:about="http://example.org/muzyka#доступнийНа">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Альбом"/>
   <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
   <rdfs:domain rdf:resource="http://example.org/muzyka#Гурт/Композитор"/>
   <rdfs:range rdf:resource="http://example.org/muzyka#Платформа"/>
</ObjectProperty>
```

```
<!-- http://example.org/muzyka#належитьДоЖанру -->

<ObjectProperty rdf:about="http://example.org/muzyka#належитьДоЖанру">
    <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
    <rdfs:range rdf:resource="http://example.org/muzyka#Жанр"/>
</ObjectProperty>

<!-- http://example.org/muzyka#отримавНагороду -->

<ObjectProperty rdf:about="http://example.org/muzyka#отримавНагороду">
    <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
    <rdfs:range rdf:resource="http://example.org/muzyka#Нагорода"/>
</ObjectProperty>

<!-- http://example.org/muzyka#єСаундтрекомДоФільму -->

<ObjectProperty rdf:about="http://example.org/muzyka#єСаундтрекомДоФільму">
    <rdfs:range rdf:resource="http://example.org/muzyka#Фільм"/>
</ObjectProperty>

<!-- http://www.semanticweb.org/ontologiya_muzyka#міститьТрек -->

<ObjectProperty rdf:about="http://www.semanticweb.org/ontologiya_muzyka#міститьТрек">
    <rdfs:domain rdf:resource="http://example.org/muzyka#Фільм"/>
    <rdfs:range rdf:resource="http://example.org/muzyka#Музичний_твір"/>
</ObjectProperty>

<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://example.org/muzyka#Імя_Виконавця -->

<DatatypeProperty rdf:about="http://example.org/muzyka#Імя_Виконавця">
    <rdfs:domain rdf:resource="http://example.org/muzyka#Виконавець"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</DatatypeProperty>



<!-- http://example.org/muzyka#країнаПоходження -->
```

```
<DatatypeProperty rdf:about="http://example.org/muzyka#країнаПоходження">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Виконавець"/>
   <rdfs:domain rdf:resource="http://example.org/muzyka#Назва_гурта"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</DatatypeProperty>


<!-- http://example.org/muzyka#кількістьТреків -->

<DatatypeProperty rdf:about="http://example.org/muzyka#кількістьТреків">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
   <rdfs:domain rdf:resource="http://example.org/muzyka#Альбом"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</DatatypeProperty>

<!-- http://example.org/muzyka#рікВипуску -->

<DatatypeProperty rdf:about="http://example.org/muzyka#рікВипуску">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#gYear"/>
</DatatypeProperty>



<!-- http://example.org/muzyka#рікВипускуАльбому -->

<DatatypeProperty rdf:about="http://example.org/muzyka#рікВипускуАльбому">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Альбом"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</DatatypeProperty>

<!-- http://example.org/muzyka#тривалість_музичного_твору -->

<DatatypeProperty rdf:about="http://example.org/muzyka#тривалість_музичного_твору">
   <rdfs:domain rdf:resource="http://example.org/muzyka#Музичний_твір"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#duration"/>
</DatatypeProperty>

<!--
///////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////
 -->
```

```
<!-- http://example.org/muzyka#Інструмент -->

<Class rdf:about="http://example.org/muzyka#Інструмент"/>
<!-- http://example.org/muzyka#Альбом -->
<Class rdf:about="http://example.org/muzyka#Альбом"/>

<!-- http://example.org/muzyka#Виконавець -->
<Class rdf:about="http://example.org/muzyka#Виконавець">
   <rdfs:subClassOf rdf:resource="http://example.org/muzyka#Гурт/Композитор"/>
</Class>
<!-- http://example.org/muzyka#Жанр -->
<Class rdf:about="http://example.org/muzyka#Жанр"/>

<!-- http://example.org/muzyka#Лейбл -->
<Class rdf:about="http://example.org/muzyka#Лейбл"/>

<!-- http://example.org/muzyka#Музичний_твір -->
<Class rdf:about="http://example.org/muzyka#Музичний_твір"/>

<!-- http://example.org/muzyka#Нагорода -->
<Class rdf:about="http://example.org/muzyka#Нагорода"/>

<!-- http://example.org/muzyka#Назва_гурта -->
<Class rdf:about="http://example.org/muzyka#Назва_гурта">
   <rdfs:subClassOf rdf:resource="http://example.org/muzyka#Гурт/Композитор"/>
</Class>

<!-- http://example.org/muzyka#Назва_музичного_твору -->
<Class rdf:about="http://example.org/muzyka#Назва_музичного_твору">
   <rdfs:subClassOf rdf:resource="http://example.org/muzyka#Музичний_твір"/>
</Class>

<!-- http://example.org/muzyka#Платформа -->

<Class rdf:about="http://example.org/muzyka#Платформа"/>

<!-- http://example.org/muzyka#Подія -->

<Class rdf:about="http://example.org/muzyka#Подія"/>

<!-- http://example.org/muzyka#Фільм -->
<Class rdf:about="http://example.org/muzyka#Фільм"/>

<!-- http://example.org/muzyka#Гурт/Композитор -->
```

```
    <Class rdf:about="http://example.org/muzyka#Гурт/Композитор"/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////
     -->

    <!-- http://example.org/muzyka#ABBA -->
    <NamedIndividual rdf:about="http://example.org/muzyka#ABBA">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
        <muzyka:Імя_Виконавця>ABBA</muzyka:Імя_Виконавця>
    </NamedIndividual>

    <!-- http://example.org/muzyka#And_Justice_for_all -->
    <NamedIndividual rdf:about="http://example.org/muzyka#And_Justice_for_all">
        <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Metallica"/>
        <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#One"/>
        <muzyka:країнаПоходження>США</muzyka:країнаПоходження>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">9</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1988</muzyka:рікВипускуАльбому>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Black_no_1 -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Black_no_1">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Type_o_negative"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Готік_рок"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Bloody_Kisses -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Bloody_Kisses">
        <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Type_o_negative"/>
        <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Black_no_1"/>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">14</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1993</muzyka:рікВипускуАльбому>
    </NamedIndividual>
```

```
    <!-- http://example.org/muzyka#Drunk_in_paris -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Drunk_in_paris">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Type_o_negative"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Готік_рок"/>
    </NamedIndividual>
    <!-- http://example.org/muzyka#EMI -->
    <NamedIndividual rdf:about="http://example.org/muzyka#EMI">
        <rdf:type rdf:resource="http://example.org/muzyka#Лейбл"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Eurovision_2023 -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Eurovision_2023">
        <rdf:type rdf:resource="http://example.org/muzyka#Подія"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Evenescence -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Evenescence">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Exit_music -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Exit_music">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Radiohead"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Альтернативний_рок"/>
        <muzyka:єСаундтрекомДоФільму rdf:resource="http://example.org/muzyka#Ромео+Джульєта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Eye_of_the_tiger -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Eye_of_the_tiger">
        <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Survivor"/>
        <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Eye_of_the_tiger_t"/>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">5</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1982</muzyka:рікВипускуАльбому>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Eye_of_the_tiger_t -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Eye_of_the_tiger_t">
```

```
    <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
    <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
    <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Survivor"/>
    <muzyka:належитьДоЖанру rdf:resource="http://www.co-ode.org/ontologies/ont.owl#Хард_рок"/>
    <muzyka:єСаундтрекомДоФільму rdf:resource="http://example.org/muzyka#Rocky"/>
  </NamedIndividual>

  <!-- http://example.org/muzyka#Fallen -->
  <NamedIndividual rdf:about="http://example.org/muzyka#Fallen">
    <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
    <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Evenescence"/>
    <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
    <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Going_Under"/>
    <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">11</muzyka:кількістьТреків>
    <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2003</muzyka:рікВипускуАльбому>
  </NamedIndividual>

  <!-- http://example.org/muzyka#Flick_of_the_Wrist -->
  <NamedIndividual rdf:about="http://example.org/muzyka#Flick_of_the_Wrist">
    <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
    <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Queen"/>
    <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
    <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Killer_Queen"/>
    <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">13</muzyka:кількістьТреків>
    <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">1978</muzyka:рікВипускуАльбому>
  </NamedIndividual>

  <!-- http://example.org/muzyka#Going_Under -->
  <NamedIndividual rdf:about="http://example.org/muzyka#Going_Under">
    <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
    <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
    <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Evenescence"/>
    <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Альтернативний_рок"/>
  </NamedIndividual>

  <!-- http://example.org/muzyka#Hello_Compukter -->
  <NamedIndividual rdf:about="http://example.org/muzyka#Hello_Compukter">
    <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
    <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Radiohead"/>
    <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
    <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Exit_music"/>
    <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">12</muzyka:кількістьТреків>
    <muzyka:рікВипускуАльбому
```

rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1997</muzyka:рікВипускуАльбому>
</NamedIndividual>

<!-- http://example.org/muzyka#I_dont_wanna_be_me -->
<NamedIndividual rdf:about="http://example.org/muzyka#I_dont_wanna_be_me">
<rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
<rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
<muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Type_o_negative"/>
<muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Готік_рок"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Kill_em_all -->

<NamedIndividual rdf:about="http://example.org/muzyka#Kill_em_all">
<rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
<muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Metallica"/>
<muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
<ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Seek_and_destroy"/>
<muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">12</muzyka:кількістьТреків>
<muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1983</muzyka:рікВипускуАльбому>
</NamedIndividual>

<!-- http://example.org/muzyka#Killer_Queen -->

<NamedIndividual rdf:about="http://example.org/muzyka#Killer_Queen">
<rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
<rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
<muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Queen"/>
<muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Рок"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Lay_All_Your_Love_on_Me -->

<NamedIndividual rdf:about="http://example.org/muzyka#Lay_All_Your_Love_on_Me">
<rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
<rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
<muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#ABBA"/>
<muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Рок"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Life_is_killing_me -->

<NamedIndividual rdf:about="http://example.org/muzyka#Life_is_killing_me">
<rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>

```
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Type_o_negative"/>
        <muzyka:доступнийНа rdf:resource="http://example.org/muzyka#Spotify"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Drunk_in_paris"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#I_dont_wanna_be_me"/>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">15</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2003</muzyka:рікВипускуАльбому>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Live_Aid_1985 -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Live_Aid_1985">
        <rdf:type rdf:resource="http://example.org/muzyka#Подія"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#MTV_Music_Awards -->

    <NamedIndividual rdf:about="http://example.org/muzyka#MTV_Music_Awards">
        <rdf:type rdf:resource="http://example.org/muzyka#Нагорода"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Metallica -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Metallica">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Mike_Shinoda -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Mike_Shinoda">
        <muzyka:граєІнструмент rdf:resource="http://example.org/muzyka#Гітара"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#One -->
    <NamedIndividual rdf:about="http://example.org/muzyka#One">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Metallica"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Треш_метал"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Pixies -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Pixies">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>
```

```
    <!-- http://example.org/muzyka#Queen -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Queen">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Radiohead -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Radiohead">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Rocky -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Rocky">
        <rdf:type rdf:resource="http://example.org/muzyka#Фільм"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Seek_and_destroy -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Seek_and_destroy">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Metallica"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Треш_метал"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Spotify -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Spotify">
        <rdf:type rdf:resource="http://example.org/muzyka#Платформа"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Super_Trouper -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Super_Trouper">
        <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#ABBA"/>
        <muzyka:виступаєНа rdf:resource="http://example.org/muzyka#Spotify"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Lay_All_Your_Love_on_Me"/>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">12</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1980</muzyka:рікВипускуАльбому>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Surfer_Rosa -->
```

```
    <NamedIndividual rdf:about="http://example.org/muzyka#Surfer_Rosa">
        <rdf:type rdf:resource="http://example.org/muzyka#Альбом"/>
        <muzyka:альбомНаписаний rdf:resource="http://example.org/muzyka#Pixies"/>
        <ontologiya_muzyka:міститьТрек rdf:resource="http://example.org/muzyka#Where_is_my_mind?"/>
        <muzyka:кількістьТреків rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">13</muzyka:кількістьТреків>
        <muzyka:рікВипускуАльбому
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1988</muzyka:рікВипускуАльбому>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Survivor -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Survivor">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Type_o_negative -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Type_o_negative">
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Woodstock_1969 -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Woodstock_1969">
        <rdf:type rdf:resource="http://example.org/muzyka#Подія"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Альтернативний_рок -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Альтернативний_рок">
        <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Барабани -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Барабани">
        <rdf:type rdf:resource="http://example.org/muzyka#Інструмент"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Бійцівський_клуб -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Бійцівський_клуб">
        <rdf:type rdf:resource="http://example.org/muzyka#Фільм"/>
    </NamedIndividual>
```

```
<!-- http://example.org/muzyka#Готік_рок -->

<NamedIndividual rdf:about="http://example.org/muzyka#Готік_рок">
   <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Гітара -->

<NamedIndividual rdf:about="http://example.org/muzyka#Гітара">
   <rdf:type rdf:resource="http://example.org/muzyka#Інструмент"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Джаз -->

<NamedIndividual rdf:about="http://example.org/muzyka#Джаз">
   <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Джон_Вільямс -->

<NamedIndividual rdf:about="http://example.org/muzyka#Джон_Вільямс">
   <rdf:type rdf:resource="http://example.org/muzyka#Виконавець"/>
   <rdf:type rdf:resource="http://example.org/muzyka#Назва_гурта"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Зоряні_війни -->
<NamedIndividual rdf:about="http://example.org/muzyka#Зоряні_війни">
   <rdf:type rdf:resource="http://example.org/muzyka#Фільм"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Поп -->
<NamedIndividual rdf:about="http://example.org/muzyka#Поп">
   <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Рок -->

<NamedIndividual rdf:about="http://example.org/muzyka#Рок">
   <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
</NamedIndividual>

<!-- http://example.org/muzyka#Саксофон -->
<NamedIndividual rdf:about="http://example.org/muzyka#Саксофон">
   <rdf:type rdf:resource="http://example.org/muzyka#Інструмент"/>
```

```
    </NamedIndividual>

    <!-- http://example.org/muzyka#Симфоничний -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Симфоничний">
        <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Скрипка -->

    <NamedIndividual rdf:about="http://example.org/muzyka#Скрипка">
        <rdf:type rdf:resource="http://example.org/muzyka#Інструмент"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Треш_метал -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Треш_метал">
        <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Where_is_my_mind? -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Where_is_my_mind?">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:виконуєтьсяВиконавцем rdf:resource="http://example.org/muzyka#Pixies"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Альтернативний_рок"/>
        <muzyka:єСаундтрекомДоФільму rdf:resource="http://example.org/muzyka#Бійцівський_клуб"/>
    </NamedIndividual>

    <!-- http://example.org/muzyka#Ромео+Джульєта -->
    <NamedIndividual rdf:about="http://example.org/muzyka#Ромео+Джульєта">
        <rdf:type rdf:resource="http://example.org/muzyka#Фільм"/>
    </NamedIndividual>

    <!-- http://www.co-ode.org/ontologies/ont.owl#Імперський_марш -->
    <NamedIndividual rdf:about="http://www.co-ode.org/ontologies/ont.owl#Імперський_марш">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Симфоничний"/>
        <muzyka:єСаундтрекомДоФільму rdf:resource="http://example.org/muzyka#Зоряні_війни"/>
        <muzyka:рікВипуску rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">29.03.1980</muzyka:рікВипуску>
        <muzyka:тривалість_музичного_твору
rdf:datatype="http://www.w3.org/2001/XMLSchema#duration">3:07</muzyka:тривалість_музичного_твору>
    </NamedIndividual>

    <!-- http://www.co-ode.org/ontologies/ont.owl#Тема_для_Хрещеного_Батька -->
```

```
    <NamedIndividual rdf:about="http://www.co-ode.org/ontologies/ont.owl#Тема_для_Хрещеного_Батька">
        <rdf:type rdf:resource="http://example.org/muzyka#Музичний_твір"/>
        <rdf:type rdf:resource="http://example.org/muzyka#Назва_музичного_твору"/>
        <muzyka:належитьДоЖанру rdf:resource="http://example.org/muzyka#Симфоничний"/>
        <muzyka:єСаундтрекомДоФільму rdf:resource="http://www.co-ode.org/ontologies/ont.owl#Хрещений_батько"/>
    </NamedIndividual>

    <!-- http://www.co-ode.org/ontologies/ont.owl#Хард_рок -->
    <NamedIndividual rdf:about="http://www.co-ode.org/ontologies/ont.owl#Хард_рок">
        <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
    </NamedIndividual>

    <!-- http://www.co-ode.org/ontologies/ont.owl#Хеві_метал -->
    <NamedIndividual rdf:about="http://www.co-ode.org/ontologies/ont.owl#Хеві_метал">
        <rdf:type rdf:resource="http://example.org/muzyka#Жанр"/>
    </NamedIndividual>

    <!-- http://www.co-ode.org/ontologies/ont.owl#Хрещений_батько -->

    <NamedIndividual rdf:about="http://www.co-ode.org/ontologies/ont.owl#Хрещений_батько">
        <rdf:type rdf:resource="http://example.org/muzyka#Фільм"/>
    </NamedIndividual>
</rdf:RDF>
```

ДОДАТОК В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Реалізація онтологічної бази «Музика та Кіно». Частина 1. Музична онтологія.

Виконав: студент 4 курсу, групи 1КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Ковальський А.М.
(прізвище та ініціали)
Керівник: к.ф.-м.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«03» 06 2025 р.

Вінниця ВНТУ - 2025 рік

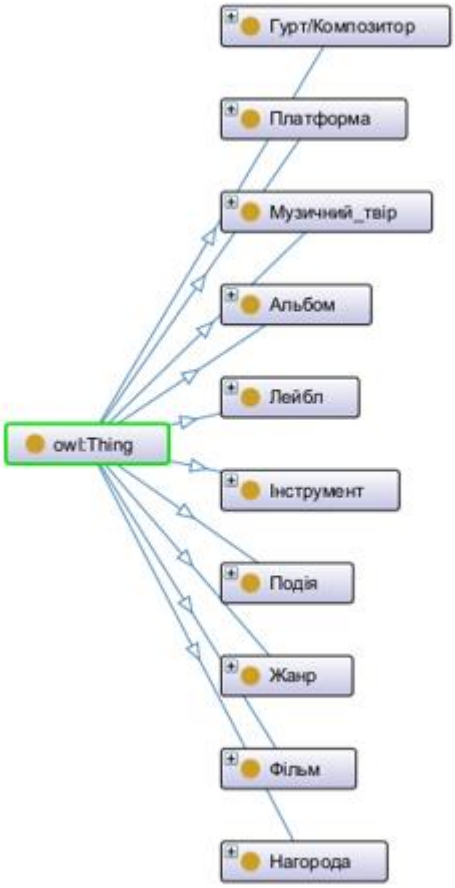


Рисунок В.1 – Граф класів онтології «музика».

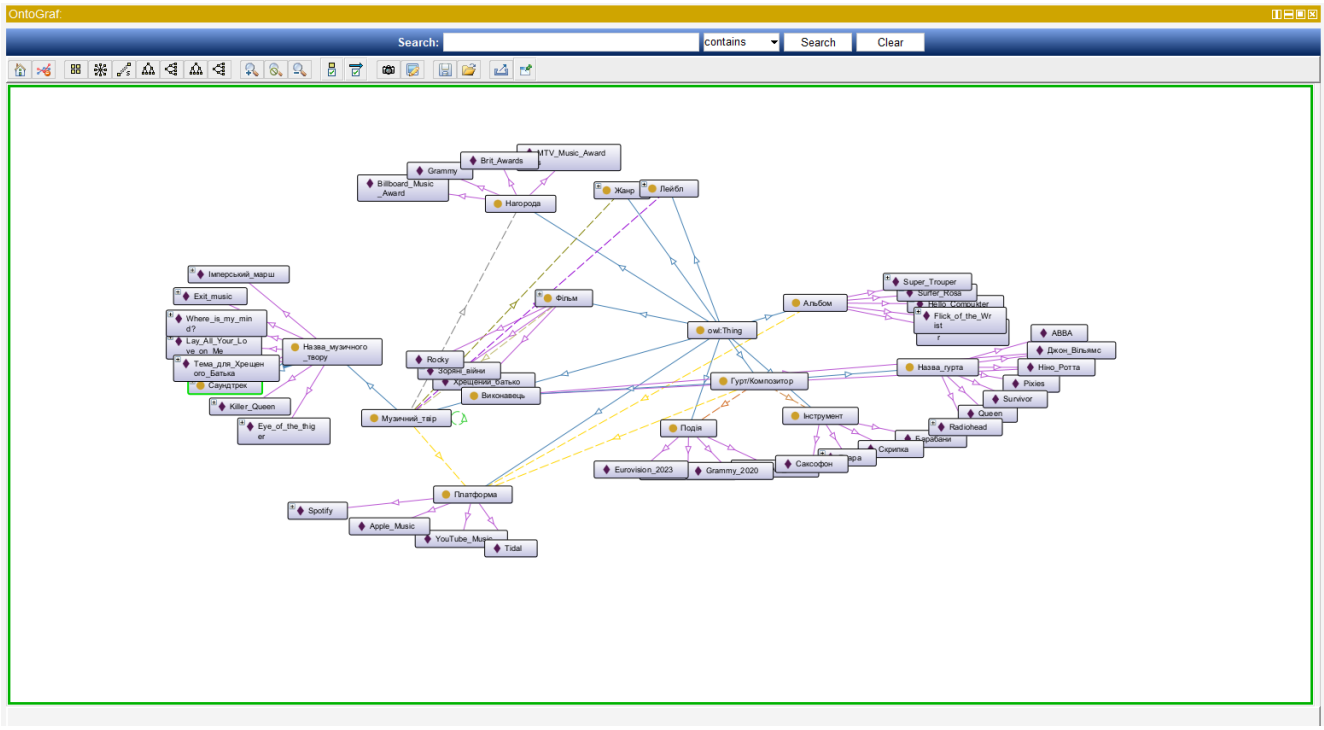


Рисунок В.2 – Фрагмент онтології «музика» засобами Ontograf

Active ontology × Entities × Individuals by class × DL Query × SPARQL Query ×

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX ont: <http://www.co-ode.org/ontologies/ont.owl#>
PREFIX om: <http://www.semanticweb.org/ontologiya_muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?instance ?class WHERE {
 VALUES ?class {
   <http://example.org/muzyka#Подія>
   <http://example.org/muzyka#Назва_гурта>
   <http://example.org/muzyka#Альбом>
   <http://example.org/muzyka#Фільм>
   <http://example.org/muzyka#Жанр>
 }
 ?instance rdf:type ?class .
}
```

instance	class
Live_Aid_1985	Подія
Woodstock_1969	Подія
Eurovision_2023	Подія
Radiohead	Назва_гурта
Джон_Вільямс	Назва_гурта
Pixies	Назва_гурта
ABBA	Назва_гурта
Survivor	Назва_гурта
Queen	Назва_гурта
Metallica	Назва_гурта
Eye_of_the_tiger	Альбом
Kill_em_all	Альбом
Hello_Compukter	Альбом
And_Justice_for_all	Альбом
Surfer_Rosa	Альбом
Flick_of_the_Wrist	Альбом
Super_Trouper	Альбом
Ромео+Джульєта	Фільм
Зоряні_війни	Фільм
Rocky	Фільм

Рисунок В.3 - Виконаний запит, який виконує пошук екземплярів певних класів в онтології «Музика»

SPARQL query:

```
PREFIX muzyka: <http://example.org/muzyka#>
PREFIX ont: <http://www.semanticweb.org/ontologiya_muzyka#>

SELECT ?альбом ?рікВипуску ?трек ?виконавець
WHERE {
 ?альбом a muzyka:Альбом ;
      muzyka:рікВипускуАльбому ?рікВипуску ;
      ont:міститьТрек ?трек .

 OPTIONAL {
  ?трек muzyka:виконуєтьсяВиконавцем ?виконавець .
 }

}
```

альбом	рікВипуску	трек	виконавець
Super_Trouper	"1980"^^<http://www.w3.org/2001/XMLSche	Lay_All_Your_Love_on_Me	ABBA
And_Justice_for_all	"25.08.1988"^^<http://www.w3.org/2001/XM	One	Metallica
Flick_of_the_Wrist	"11.10.1974"^^<http://www.w3.org/2001/XM	Killer_Queen	Queen
Eye_of_the_tiger	"29.05.1982"^^<http://www.w3.org/2001/XM	Eye_of_the_tiger_t	Survivor
Surfer_Rosa	"5.07.1988"^^<http://www.w3.org/2001/XML	Where_is_my_mind?	Pixies
Kill_em_all	"13.04.1983"^^<http://www.w3.org/2001/XM	Seek_and_destroy	Metallica

Рисунок В.4 - Виконаний запит про Альбом, Рік випуску, трек і виконавця, через атрибути «рікВипуску», «міститьТрек», «виконуєтьсяВиконавцем»

Active ontology × Entities × Individuals by class × DL Query × SPARQL Query ×

SPARQL query:

```
PREFIX : <http://example.org/muzyka#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?object ?type ?property ?value
WHERE {
 {
  ?object rdf:type :Альбом .
  BIND("Альбом" AS ?type)
  OPTIONAL { ?object :рікВипуску ?value . BIND("рікВипуску" AS ?property) }
  OPTIONAL { ?object :альбомНаписаний ?value . BIND("альбомНаписаний" AS ?property) }
  OPTIONAL { ?object :міститьТрек ?value . BIND("міститьТрек" AS ?property) }
 }
 UNION
 {
  ?object rdf:type :Назва_музичного_твору .
  BIND("Трек" AS ?type)
  OPTIONAL { ?object :виконуєтьсяВиконавцем ?value . BIND("виконуєтьсяВиконавцем" AS ?property) }
  OPTIONAL { ?object :належитьДоЖанру ?value . BIND("належитьДоЖанру" AS ?property) }
  OPTIONAL { ?object :єСаундтрекомДоФільму ?value . BIND("єСаундтрекомДоФільму" AS ?property) }
 }
}
```

object	type	property	value
Eye_of_the_tiger	"Альбом"	"альбомНаписаний"	Survivor
Kill_em_all	"Альбом"	"альбомНаписаний"	Metallica
Hello_Compukter	"Альбом"	"альбомНаписаний"	Radiohead
And_Justice_for_all	"Альбом"	"альбомНаписаний"	Metallica
Surfer_Rosa	"Альбом"	"альбомНаписаний"	Pixies
Flick_of_the_Wrist	"Альбом"	"альбомНаписаний"	Queen
Super_Trouper	"Альбом"	"альбомНаписаний"	ABBA
One	"Трек"	"виконуєтьсяВиконавцем"	Metallica
Exit_music	"Трек"	"виконуєтьсяВиконавцем"	Radiohead
Тема_для_Хрещеного_Батька	"Трек"	"належитьДоЖанру"	Симфоничний
Seek_and_destroy	"Трек"	"виконуєтьсяВиконавцем"	Metallica
Lay_All_Your_Love_on_Me	"Трек"	"виконуєтьсяВиконавцем"	ABBA
Імперський_марш	"Трек"	"належитьДоЖанру"	Симфоничний
Where_is_my_mind?	"Трек"	"виконуєтьсяВиконавцем"	Pixies
Killer_Queen	"Трек"	"виконуєтьсяВиконавцем"	Queen
Eye_of_the_tiger_t	"Трек"	"виконуєтьсяВиконавцем"	Survivor

Рисунок В.5 - Виконаний запит, що показує інформацію про трек.

ДОДАТОК Г(довідниковий)

ІНСТРУКЦІЯ КОРИСТУВАЧА

Запускаємо середовище Protégé та завантажуємо файл з онтологією. Обираємо вкладку Ontograf для пошуку інформації за гілками графа. Також можна розглядати ієрархію онтології, вивчати деталізацію предметної галузі (рис. Г.1).

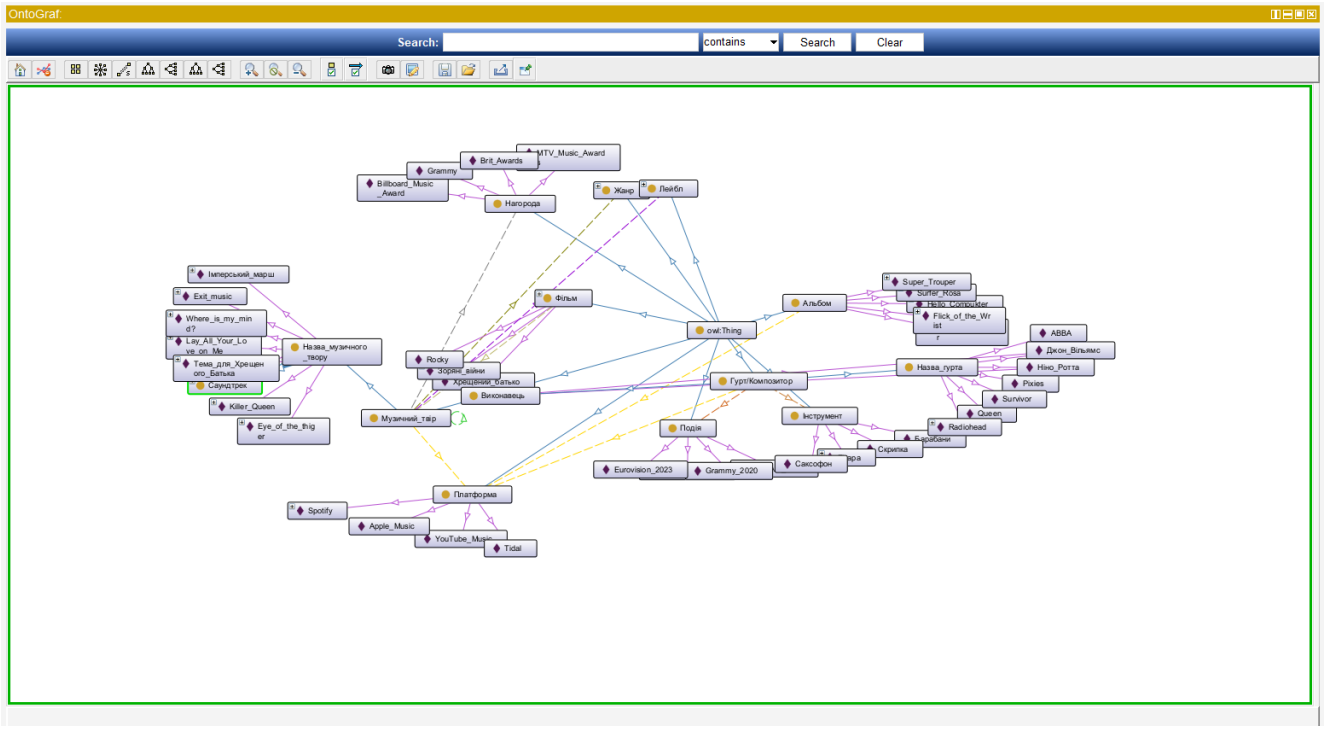


Рисунок Г.1 – Пошук за графом онтології

Для того, щоб виконати запит SPARQL потрібно перейти до вкладки «SPARQL Query». Якщо вкладка не відображається, в меню Window переходимо по вкладці Tabs і виберемо «SPARQL Query». Для створення запитів на мові SPARQL спочатку треба в текстовому полі вкладки «SPARQL Query» задати простори імен, які будуть використовуватися. Для виконання запиту потрібно натиснути «Execute» і під кодом відобразиться результат у вигляді таблиці. На рисунку Г.2 подано результат цього запиту.

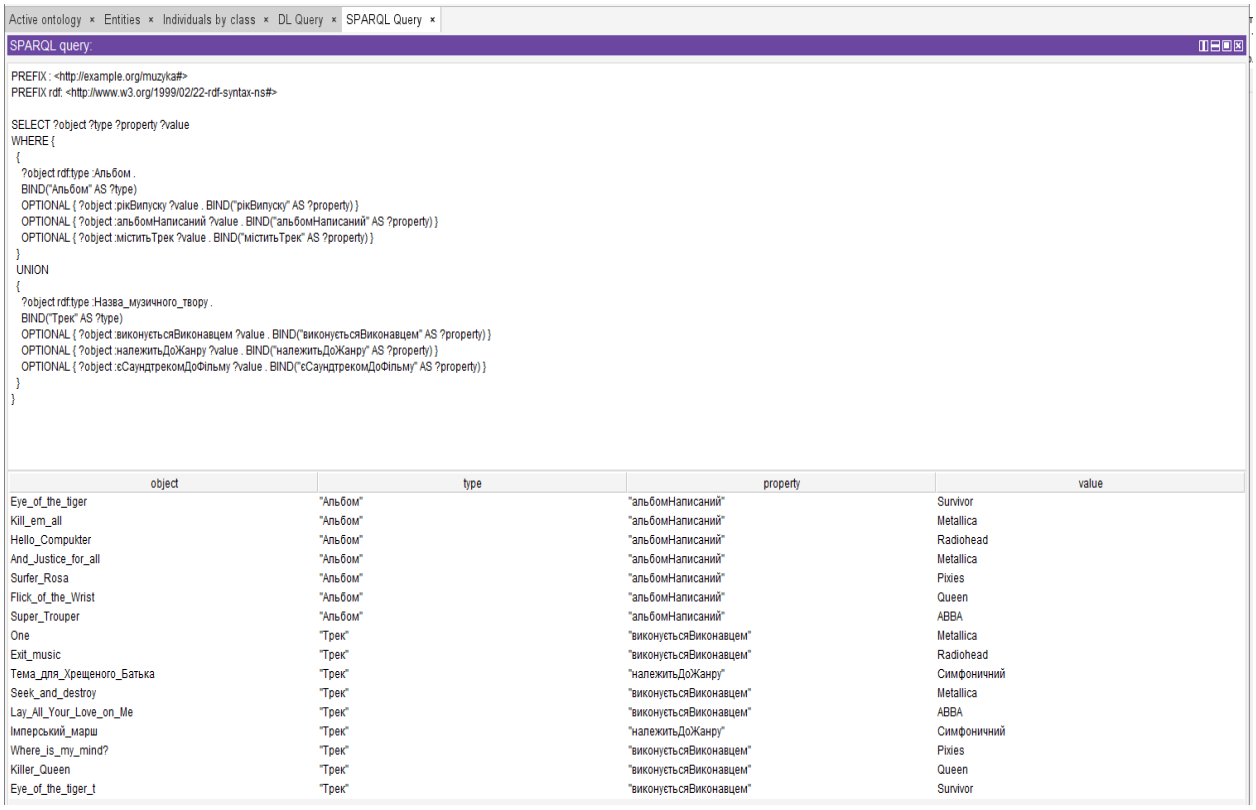


Рисунок Г.2 – Результат виконання SPARQL-запити