UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Graph Databases A.Y. 2024-2025

Task 3 – RDF dataset

Master Degree in Computer Engineering

Deadline: 20 December, 2025

Group Acronym	MELODY	
Github	`https://github.com/ludovicodimartino/MELODY`	
Last Name	**First Name**	**Badge Number**
Di Martino	Ludovico	2104292
Galli	Filippo	2120826
Rigobello	Manuel	2103374

1 Mapping Open Data to RDF

During the mapping phase, we encountered several challenges, particularly in linking data contained across different CSV files. The files involved in the process are the following:

- `albums.csv`: from this file we extracted the name of the album, the total number of tracks and the artist ID, in order to link the album to the corresponding *artist*.
- `artists.csv`: from this file we extracted the artist name, the number of followers, the main genre, the artist type and, based on the value of the artist popularity, we set a value chosen from "High", "Medium" or "Low".
- `releases.csv`: it contains the associations between artists and albums.
- `songs.csv`: from this file we extracted the song name, the artists featuring each song and the song type (solo or collaborative).
- `tracks.csv`: we used this file to link the song that appears in *BillboardHot100* to the corresponding album.
- `song_chart.csv`: from this file we extracted the position of a song in a Billboard. And therefore, we used it to populate both the class *BillboardHot100* and the class *Membership*.
- `acoustic_feature.csv`: from this file we extracted the features of songs of interest such as danceability and BPM.
- `classified_genres.csv`: this file defines, for each main genre, all its possible sub-genres.
- `GrammyCategoriesUppercase.csv`: this file contains the list of possible categories for a Grammy in camelCase format.
- `the_grammy_awards_mapped_uppercase.csv`: this file was used to map songs, artists, and albums to the GRAMMY categories they were nominated for or won.

The complete RDF Dataset (`melody-complete-rdf-dataset.ttl`) can be found in the folder `https://github.com/ludovicodimartino/MELODY/tree/main/PopulateRDFdb` of our repository along with all the python notebooks used for populating it starting from the csv files.

To clearly explain the mapping of the open data found in the RDF dataset, we have provided the ontology design below.

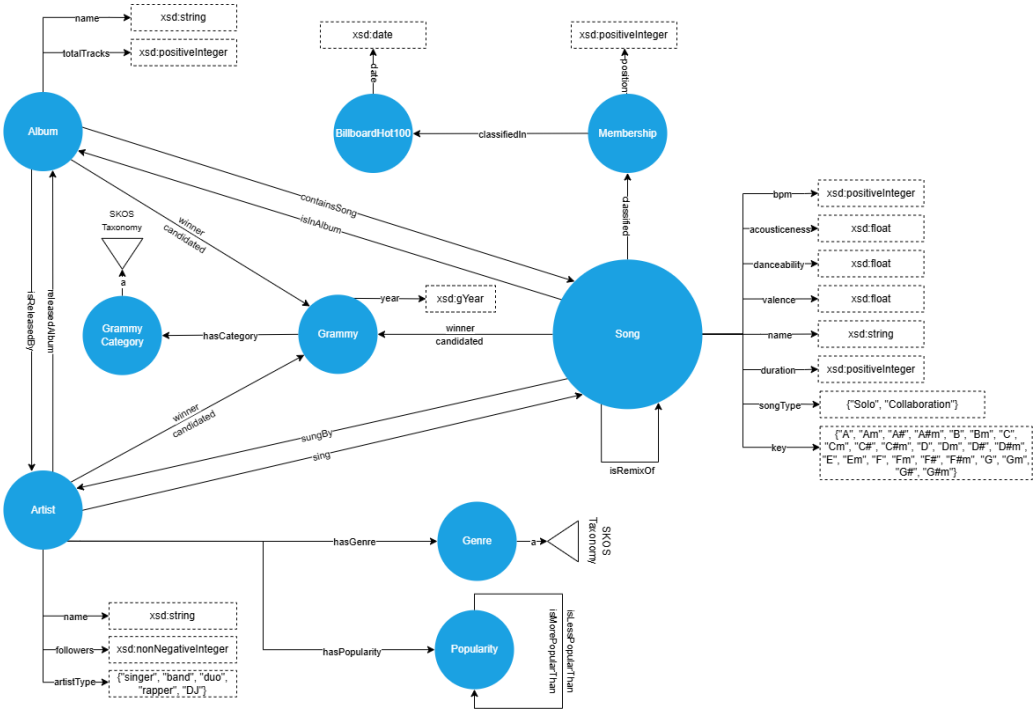

Figure 1: MELODY Ontology design diagram

In the following sections, we present the main challenges we faced during the mapping phase.

1.1 Songs

Regarding songs, the main challenges were identifying remixes based on song titles and associating songs from the `songs.csv` file with nominees in the `the_grammy_awards_mapped_uppercase.csv` dataset. For both tasks, we used the Python library *FuzzyWuzzy*, which simplifies string matching.

To identify remixes, we analyzed the data and observed that all remixes include the word "remix" in their titles. Based on this observation, we filtered all songs with "remix" in their title and, for each of these, created a list of potential original songs featuring one of the artists present in the remix. Next, we applied the `extractOne` method from *FuzzyWuzzy* to find the best match between the remix song and the list of potential original songs. The `extractOne` method returns a score indicating the quality of the match. We considered matches valid only if their score exceeded 90, a threshold determined by trial and error. Each valid match is then inserted into the RDF dataset.

For the association between songs and GRAMMY nominees, we first performed a cleaning step to standardize song titles in the csv dataset, removing some words that could interfere with matching (e.g. *solo*, *version*, *theme*). Then, we focused on Grammy entries from the dataset where the category did not include the words "artist" or "album" and included a worker. For these entries, we again used the `extractOne` method to search for matches in the `songs.csv` file. As before, we established a threshold score through trial and error, above which an association was deemed valid.

1.2 Albums

Regarding albums, we mainly focused on finding only albums containing songs that appeared on the *Billboard-Hot100*, excluding any albums that did not meet this criterion. This was done through `tracks.csv`, which allowed us to link each song from `song_chart.csv` with the corresponding album. A similar approach was used to associate each artist ID with the corresponding artist.

For the association between albums and GRAMMY nominees, we began by standardizing album titles in the dataset, removing symbols, commas, and periods. Then, we focused on GRAMMY entries where the category includes the term "artist" and specifies a worker. One of the main challenges we addressed was homonymy, where different albums share the same title. To solve this problem, we generated a list of all workers for each album. In cases where there was a match between the title of the album in `Albums.csv` and the title in `GrammyCategoryUppercase.csv`, we compared the album's list of workers with the GRAMMY winner or candidate. If at least one match was found, we established the association.

1.3 Artists

The file `artist.csv` was already well formatted for our scope. The bit of work in this part was dedicated to the selection of the individual to associate with the artist that describes the amount of popularity the artist has.

The main work for artists was to handle the problem of correctly matching the name of the artists with the winner or candidate of a GRAMMY. The first step was to filter only the Grammy that are assigned directly to an artist, and not for a particular song or album. For this scope, we decide to follow these rules based on the category of the Grammy and to the fields *nominee* and *workers*:

- The Grammy category contains the keyword "Artist" and, if the field *workers* is empty, then we use the field *nominee* to retrieve the name of the artist, otherwise we use the data stored as *workers*.
- The Grammy category contains the keyword "Producer" and, as for the previous case, we consider the field *nominee* in case of the empty *workers*, otherwise we use *workers*.
- The Grammy category contains the keyword "Performance" and *workers* is empty. In this situation *nominee* will contain the name of the artist.

The second and final step was to establish the association between the artists and one or more Grammy he won. The Grammy dataset contains the name of the artists instead of their id, so also in this case we used the *FuzzyWuzzy* Python library. We started with a normalization of the artist name in both Grammy and artists dataset to eliminate some missing matches due to a difference in the formatting of the strings to refer to an artist. Then we chose to create the association only if the match score given by the *extractOne* function and applied to the two strings was 100, so that we avoid possible incorrect match due to equal sub-string (for instance, the band "Red" and "Red Hot Chili Peppers").

1.4 Grammy

While analyzing the `the_grammy_awards.csv` dataset, we identified significant discrepancies between the category names in the dataset and the official Grammy categories from DBpedia. The main issues included:

- Non-standardized formatting
- Presence of special characters
- Variations in naming conventions

To address these discrepancies, we used ChatGPT to create an accurate mapping between the categories listed in the CSV file and the official categories from DBpedia. This process allowed us to standardize and normalize the category names, ensuring consistency for the subsequent processing. Following the creation of the mapping file, we replaced the incorrectly formatted category names in `the_grammy_awards_raw.csv` with the official names.

Then, we grouped the Grammy categories into 13 macro-categories to establish a hierarchical structure designed to cover all known musical genres worldwide. This hierarchical structure was then integrated into our ontology using SKOS.

2 SHACL Validation

We implemented a few SHACL shapes to validate our data, which helped identify inconsistencies in the dataset. For example, using the SongShape, we discovered that some songs in the initial CSV files were not mapped to a Billboard Hot 100 entry. Threfore, we removed those songs from the dataset.

All the SHACL shapes we used can be found in the SHACL folder of our repository.

3 Statistics

In the following table are described the main statistics of our resulting RDF dataset.

MELODY RDF dataset statistics	
Total statements (no inferencing)	1211131
Total songs	20402
Total albums	6329
Total artists	11518
Total remixes	4
Total Grammy categories	194
Total artist genres	1601
Grammy-Songs Associations	1312
Grammy-Album Associations	293
Grammy-Artist Associations	117

4 Ontology Changes

This section outlines the adjustments made to our ontology since the previous task submission.

4.1 Class Renaming

The Billboard Hot 200 pertains exclusively to albums; however, we mistakenly named the class for songs as *SongHot200*. It should instead be named *SongHot100*. Similarly, the *Billboard* class has been renamed to *BillboardHot100* to reflect its proper scope.
Finally, we realized that naming the class *SongHot100* was unnecessary, as it aligns with the overall scope of our project. Therefore, we decided to name it simply *Song*.

4.2 Added Object Properties

In the ontology submitted for the previous task, we omitted the connection between the *Artist* and *Album* classes. To address this, we introduced two new object properties: *releasedAlbum*, which links an album to its artist/s, and *isReleasedBy*, which links artist/s to their album.

4.3 Added Data Properties

We realized that we had forgotten to include the *position* data property, which represents a song's rank in the Billboard Hot 100. Therefore, we added it.

4.4 Changed range of the followers object property

After reviewing the data, we noticed that an artist can have zero followers. As a result, the data property *followers* needs to accommodate the value 0. To address this, we changed its range from *xsd:positiveInteger* to *xsd:nonNegativeInteger*.

4.5 Solved SKOS artist genres conflicts

We noticed that the SKOS taxonomy of artist genres submitted in the previous task contained some genres that appeared as both main genres and sub-genres (e.g., *Blues* as a main genre and *Blues* as a sub-genre of *Blues*). This meant that sub-genres with the same name as a main genre were treated as both a SKOS Concept Schema and a SKOS Concept. To address this, we renamed the main genres by adding the suffix "Genres".