SONGS ONTOLOGY REPORT

Rahul Kejriwal (CS14B023)   Ajmeera Balaji Naik (CS14B034)   Bikash Gogoi (CS14B039)

Table of contents:

Introduction:

Our Ontology models the domain of "Song". It captures information about songs, artists, albums etc. using description logic.

It stores information/knowledge about songs like name of the album it belongs to, name of the band that sang it, genre of the song, language of the song, studio in which song was recorded, if the song was composed by single person, then the name of the singer and more.

Classes and Properties:

1. Class Hierarchy:
   a. Album
   b. Band
   c. Genre
   d. Language
   e. Person
      i. Instrumentalist
         1. Accordionist
         2. Bandurist
         3. Bassist
         4. Bassoonist
         5. Bouzouki_play
         6. Cellist
         7. Clarinetist
         8. Drummer
         9. Euphoniumist
         10. Flautist
         11. Guitarist
         12. Harpist
         13. Hornist
         14. Keyboardist
         15. Organ_grinder
         16. Organist
         17. Percussionist
         18. Pianoist
         19. Saxophonist
         20. Trumpeter
         21. Violinist
         22. violist

        ii.   Lyricist

       iii.   Music_Director

       iv.   Rapper

        v.   Vocalist

  f. Playlist

  g. Song

  h. Studio
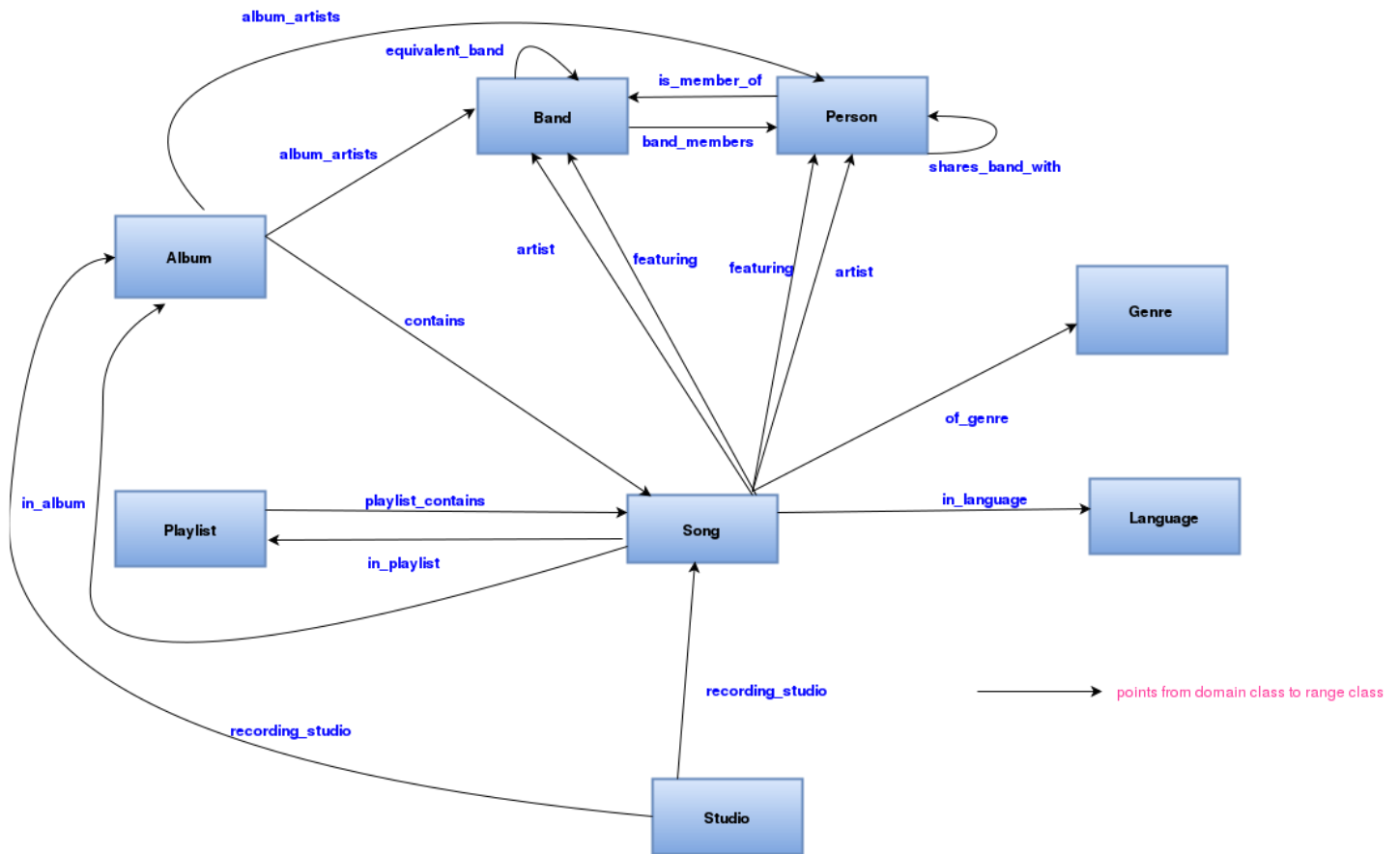
2. Object Properties:

  a. album_artists

  b. artist

  c. brand_members

  d. contains

  e. featuring

  f. in_album

  g. in_language

  h. in_playlist

  i. is_member_of

  j. of_genre

  k. playlist_contains

  l. recording_studio

  m. shares_band_with

  n. equivalent_band

3. Datatype Properties:

  a. award

  b. bitrate

  c. file_format

  d. gender

  e. is_instrumental

  f. no_of_songs

  g. origin

  h. song_length

  i. year

Object Properties Relationships:

Ontology metrics:

Ontology metrics:	□⊟▭⊠
Metrics	
Axiom	538
Logical axiom count	380
Declaration axioms count	158
Class count	35
Object property count	15
Data property count	15
Individual count	95
DL expressivity	SRIQ(D)
Class axioms	
SubClassOf	28
EquivalentClasses	0
DisjointClasses	28
GCI count	0
Hidden GCI Count	0
Object property axioms	
SubObjectPropertyOf	4
EquivalentObjectProperties	0
InverseObjectProperties	3
DisjointObjectProperties	1
FunctionalObjectProperty	4
InverseFunctionalObjectProperty	1
TransitiveObjectProperty	2
SymmetricObjectProperty	2
AsymmetricObjectProperty	12
ReflexiveObjectProperty	0
IrrefexiveObjectProperty	12
ObjectPropertyDomain	13
ObjectPropertyRange	14
SubPropertyChainOf	0
Data property axioms	
SubDataPropertyOf	13
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	7
DataPropertyDomain	12
DataPropertyRange	9
Individual axioms	
ClassAssertion	95
ObjectPropertyAssertion	53
DataPropertyAssertion	67
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0

Modelling Rationale:

1. We wanted to capture most of the information related to songs and entities somehow related to songs. We thought it is logical to make entities as classes and binary relations as object properties and data properties, i.e., the subject and object in triples are classes and properties are the predicates.
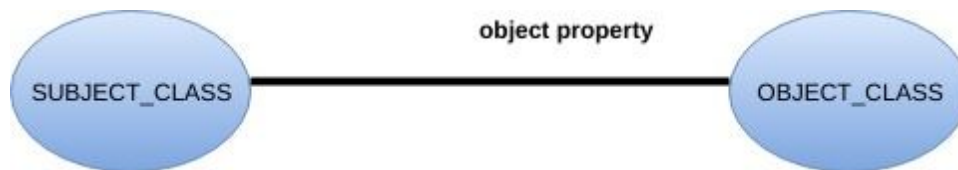
2. We have made all the human entities as one class. Since human entities who are related to song are of different kinds like singer, playback, singer, music director etc., so we have divided whole person class into subclasses of Instrumentalist, Lyricist, Music_Director, Rapper and Vocalist. Note however that none of them are disjoint since the same individual could perform multiple kinds of jobs.
3. Again Instrumentalist can be divided into subclasses of individual music instruments.
4. The main entities surrounding songs were Artists, Albums, Playlists, Genre, Language and Studio each of which we modelled as classes and added individuals of their types.
5. These entities were connected using object properties like artist, album_artist, contains, featuring, in_album etc.
6. We used data proprties to model metadata about song files that we had. These include bitrate, file_format, song_length, year.
7. We also use data properties to store awards won by songs/albums/artists.
8. We also use data properties to identify is an instrumental or not. We could have made Instrumental as a class but we chose to do it this way because we were running short of data type properties.

Possible queries:

1. Query to display songs in queried albums.
2. Query to find artist of the song and whether it was a band or an individual.
3. Query to find name of the band which composed that song, and information about that song.
4. Query to find whether the both bands are same or not, i.e., whether one of the bands was the former version of the other.
5. Query to check membership of person in a band.
6. Query to find genre of the song and find similar songs based on the genre.
7. Query to find playlists containing queried songs.
8. Query to find songs that have won certain awards.
9. Query to list all the past albums or songs recorded at queried studio.
10. Query to list all the songs in queried album.
11. Query to check whether the song belongs to queried playlist or not.

Limitations & Future Extensions:

1. Currently, the ontology doesn't support user ratings. We could possibly extend it by adding weighted edges between songs that model user ratings. The weights could be modified as per user feedback and this edge could be used for recommending songs.
2. We are using song/artist/album names directly in the IRIs. Ideally, we should use some other identifier for the IRIs and have an edge for name/title. This would allow entering songs/albums/artists etc. having the same name. We forgoed this in the interest of saving time.
3. We could model remix and inspiration songs in future versions and have transitive relations connecting originals/inspirations.

Applications:

1. It could be used in music copyrights issuing offices. It allows people to query records of the songs quickly before issuing copyright for new songs.
2. Could be extended for use in songs recommendation engines.