

# VKG Hands-on

## Decidable Logics for Knowledge and Data

Source Code: [tobsel7/music-event-ontology](#)

The Git repository has all the code and data used to build the database and ontology.

## Domain

I decided to use the domain of music to build a virtual knowledge connecting two hypothetical data sources: a music performance database and a song database. This is a realistic setup, because there are many services dealing either with songs and artists or venues and concerts but not both.

## Data Source

The data source is personal notes on concert attendance from 2016. To build the whole database content, some additional information is generated in a Python script.

## Database Schema

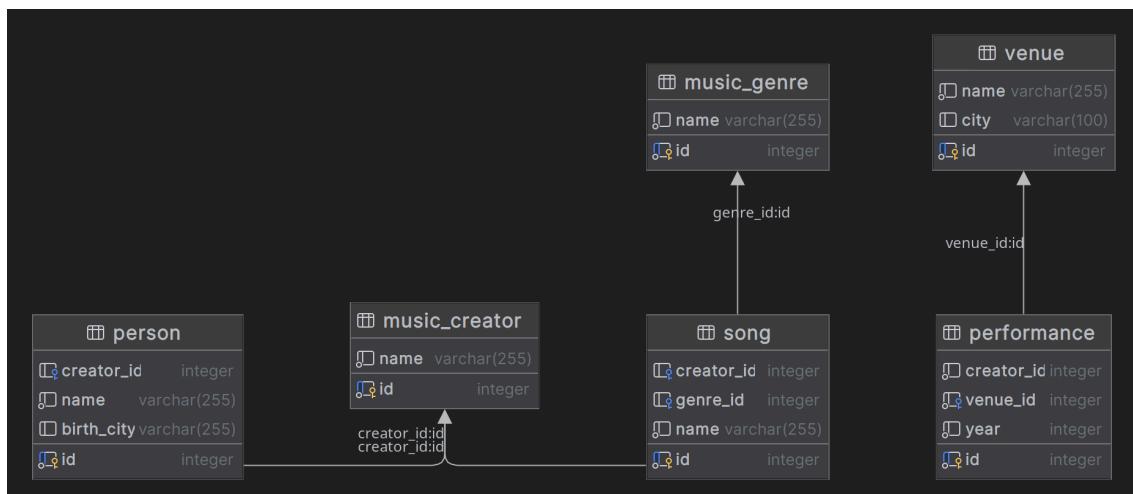
Song schema:

- *music\_creator*: generic table for bands and individual artists
- *person*: real person associated with music creators
- *music\_genre*: genres of songs
- *song*: metadata of songs performed by musicians

Event schema:

- *venue*: locations where concerts are held
- *performance*: info on a concert

The schemas can be related using the *creator\_id*, which consistently identifies the artists.



The event database holds 213 performances (recorded concert attendances) at 52 unique venues. The song database contains around 300 facts on genres, songs, and creators.

## DL Specification

DL axioms with description	
MusicCreator ≡ Band ∪ IndividualArtist	Every creator is a band or an individual artist.
Band ⊓ IndividualArtist ⊑ ⊥	An artist cannot be both an individual artist and a band at the same time.
IndividualArtist ⊑ Person	Every individual artist is a person.
Band ⊑ MusicCreator	Every band is a music creator.
IndividualArtist ⊑ MusicCreator	Every individual artist is a music creator.
Band ⊑ ∃ isMemberOf .Person	Every band has a member.
Song ⊑ ∃ isCreatedBy.MusicCreator	Every song has a creator.
Song ⊑ ∃ hasGenre.Genre	Every song has a genre.
Performance ⊑ ∃ performedAt.Venue	Every performance was performed at a venue.

## Knowledge Graph Mappings

Target	SQL Query
:creator/{id} a :IndividualArtist .	SELECT c.id, MAX(c.name) as name FROM song.music_creator c JOIN song.person p ON c.id = p.creator_id GROUP BY c.id HAVING COUNT(p.creator_id) = 1
:creator/{id} :name {name} .	
:person/{id} a :Person .	SELECT id, name, birth_city FROM song.person
:person/{id} :name {name} .	
:person/{id} :birthCity {birth_city} .	

:person/{id} :isMemberOf :creator/{creator_id} .	SELECT id, creator_id FROM song.person WHERE creator_id IS NOT NULL
:genre/{id} a :Genre .	SELECT id, name FROM song.music_genre
:genre/{id} a :name {name} .	
:song/{id} a :Song .	SELECT id, name, creator_id, genre_id FROM song.song
:song/{id} :name {name} .	
:song/{id} :isCreatedBy :creator/{creator_id} .	
:song/{id} :hasGenre :genre/{genre_id} .	
:venue/{id} a :Venue .	SELECT id, name, city FROM event.venue
:venue/{id} :name {name} .	
:venue/{id} :city {city} .	
:perf/{perf_id} a :Performance .	SELECT p.id as perf_id, p.year, p.venue_id, p.creator_id FROM event.performance p
:perf/{perf_id} :year {year} .	
:perf/{perf_id} :performedAt :venue/{venue_id} .	
:creator/{creator_id} :hasPerformance :perf/{perf_id} .	
:creator/{id} a :Band .	SELECT c.id, MAX(c.name) as name FROM song.music_creator c JOIN song.person p ON c.id = p.creator_id GROUP BY c.id HAVING COUNT(p.creator_id) > 1
:creator/{id} :name {name} .	

## Queries

All queries use the following prefix to refer to the music schema:  
 PREFIX : <<http://example.org/music#>>

## Q1 - Did an artist hold a concert in the city of Linz?

This query simply checks the venue's data property for the city. It is an ASK query, so it returns either true or false depending on whether the result is empty or non-empty.

```
ASK {
?venue :city "Linz" ;
a :Venue .
?perf :performedAt ?venue .
?creator :hasPerformance ?perf .
?person :isMemberOf ?creator .
}
```

### Result

True

### Query Rewriting

```
SELECT perf.creator_id, v.id
FROM event.venue v
JOIN event.performance perf
ON v.id = perf.venue_id
JOIN song.person pers
ON pers.creator_id = perf.creator_id
WHERE v.city = 'Linz'
LIMIT 1
```

## Q2 - Which Indie Pop songs exist?

To search for all Indie Pop songs and show the artists associated with them, this query filters the songs on the genre and then relates it to the artist.

```
SELECT ?songName ?artistName
WHERE {
?genre a :Genre ;
:name "Indie Pop" .
?song a :Song ;
:name ?songName ;
:isCreatedBy ?creator ;
:hasGenre ?genre .
?creator :name ?artistName .
}
```

### Result (subset)

The result is all songs and its creator name where the song genre is Indie Pop.

songName	artistName
Weit weg	Provinz
Zu spät um umzudrehen	Provinz

## Query Rewriting

```

SELECT DISTINCT
    s.id AS song_id,
    s.name AS song_name,
    c.id AS creator_id,
    c.name AS creator_name
FROM song.song s
JOIN song.music_genre g ON s.genre_id = g.id
JOIN song.music_creator c ON s.creator_id = c.id
JOIN (
    SELECT creator_id
    FROM song.person
    GROUP BY creator_id
) pers ON c.id = pers.creator_id
WHERE g.name = 'Indie Pop';

```

## Q3 - Which songs were written by individual artists?

Even though the database has no flag to signal if a creator is a band or individual artist, the SPARQL query can directly access the IndividualArtist entity. The rest of the query just joins the rest of the data for display.

```

SELECT ?songName ?artistName ?birthCity
WHERE {
    ?song a :Song ;
        :name ?songName ;
        :isCreatedBy ?artist .
    ?artist a :IndividualArtist .
    ?person :isMemberOf ?artist ;
        :name ?artistName ;
        :birthCity ?birthCity .
}

```

## Result (subset)

As output, the query delivers songs and information about the artist. Since the artist is an individual artist, the real name can be queried from the person and shown here.

songName	artistName	birthCity
Inside	Iryna Shvydka	Kyiv

Rainbow	Iryna Shvydka	Kyiv
Mallorca (Da bin ich daheim)	Mia Julia Brückner	Starnberg

## Q4 - Which venues had concerts of multiple genres?

Here, the grouping feature is used to check if the venues have groups of genres larger than 1. This way, we can see all venues with performances in different genres.

```

SELECT ?city ?venueName (GROUP_CONCAT(DISTINCT ?genreName; separator=", ") AS
?genres)
WHERE {
  ?venue a :Venue ;
    :city ?city ;
    :name ?venueName .

  ?perf :performedAt ?venue .
  ?creator :hasPerformance ?perf .
  ?song :isCreatedBy ?creator ;
    :hasGenre ?genre .

  ?genre :name ?genreName .
}
GROUP BY ?city ?venueName
HAVING (COUNT(DISTINCT ?genre) > 1)
ORDER BY ?city ?venueName

```

## Result

The following two venues have concerts of different genre types. This is true for others as well, but the data is limited, so there are only two result entries.

city	venueName	genres
Neußerling	Noppen Air	Austropop, Folk Punk
Linz	Posthof	Electro Rock, Hip-Hop