Information Integration in GLAV

SAPIENZA
UNIVERSITÀ DI ROMA

Data gathering

The data we integrated derives from 4 CSV files found on Kaggle. The information deals with Spotify songs raw data, which is combined also with Artists data and Youtube data in some of the sources.

The following links present the original data we used for our projects:

- https://www.kaggle.com/datasets/pieca111/music-artists-popularity
- https://www.kaggle.com/datasets/salvatorerastelli/spotify-and-youtube
- https://www.kaggle.com/datasets/ambaliyagati/spotify-dataset-for-playing-around-with-sql
- https://www.kaggle.com/datasets/muhmores/spotify-top-100-songs-of-20152019

Technologies used

- Pentaho Data Integration: processes the raw sources and populates the SQL databases (and analogous CSV sources for debugging and accessibility of the data)
- PostgreSQL: contains the materialization of the solution, defines the global schema and its constraints, runs the queries
- Python: virtualization scripting approach to our source files, comparing them with the result of the query run on the materialized schema

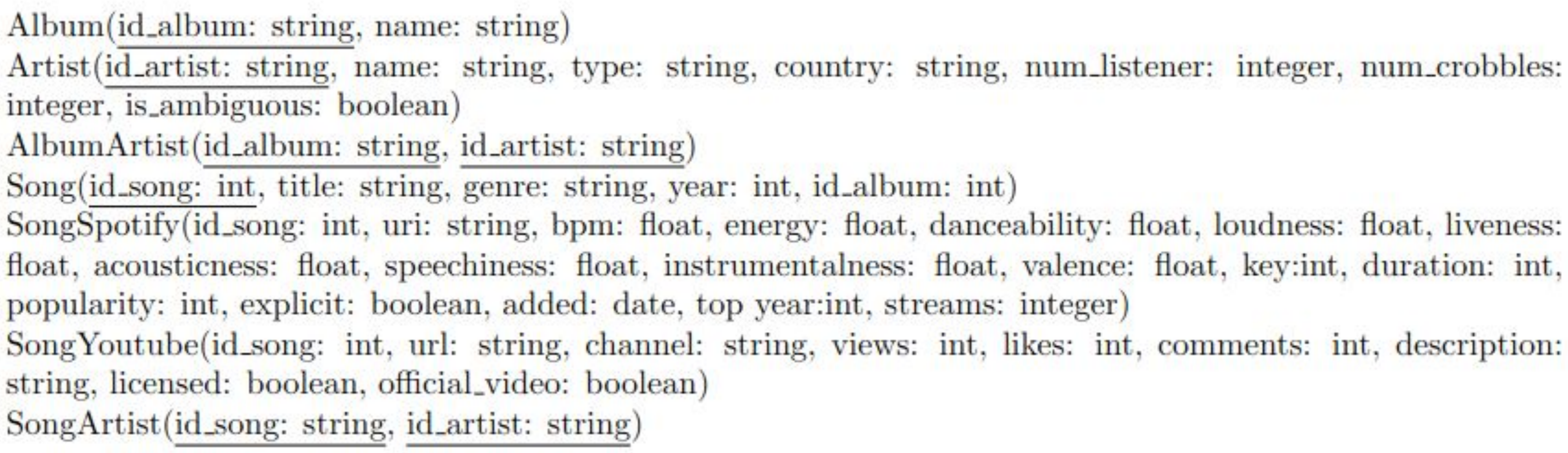
Source schema

As the source schema, we chose to maintain the original structure of our sources. Although, one of them presented a list of artists for each song, that we chose to deserialize in order to store the artists separately.

$$\begin{aligned}
&\mathrm{T1}(\vec{a})\\
&\mathrm{T2}(\vec{b})\\
&\mathrm{T3}(c_1, c_2, c_3, c_5, c_6, c_7, c_8)\\
&\mathrm{T3}_{artists}(c_2, c_4)\\
&\mathrm{T4}(\vec{d})
\end{aligned}$$

The Global Schema

In the global schema we normalized the information about Song, Artist, and Album. Moreover, we created additional tables to model the N:N relations (e.g. SongArtist).
In addition we employed vertical fragmentation to divide from the song general information those regarding Spotify and Youtube



Album(id_album: string, name: string)
Artist(id_artist: string, name: string, type: string, country: string, num_listener: integer, num_crobbles: integer, is_ambiguous: boolean)
AlbumArtist(id_album: string, id_artist: string)
Song(id_song: int, title: string, genre: string, year: int, id_album: int)
SongSpotify(id_song: int, uri: string, bpm: float, energy: float, danceability: float, loudness: float, liveness: float, acousticness: float, speechiness: float, instrumentalness: float, valence: float, key:int, duration: int, popularity: int, explicit: boolean, added: date, top year:int, streams: integer)
SongYoutube(id_song: int, url: string, channel: string, views: int, likes: int, comments: int, description: string, licensed: boolean, official_video: boolean)
SongArtist(id_song: string, id_artist: string)

Global Schema Constraints

Only foreign keys on the IDs and key constraints.
We have a key constraint for every ID and for all the other fields except the ID. Moreover, all the fields are unique in each tuple of SongArtist, AlbumArtist, SongSpotify and SongYoutube.

$$
\begin{aligned}
&Song[id_{album}] \subseteq Album[id_{album}]\\
&AlbumArtist[id_{album}] \subseteq Album[id_{album}]\\
&AlbumArtist[id_{artist}] \subseteq Artist[id_{artist}]\\
&SongArtist(id_{artist}) \subseteq Artist[id_{artist}]\\
&SongArtist(id_{song}) \subseteq Song[id_{song}]\\
&SongSpotify[id_{song}] \subseteq Song[id_{song}]\\
&SongYoutube(id_{song}) \subseteq Song[id_{song}]
\end{aligned}
$$

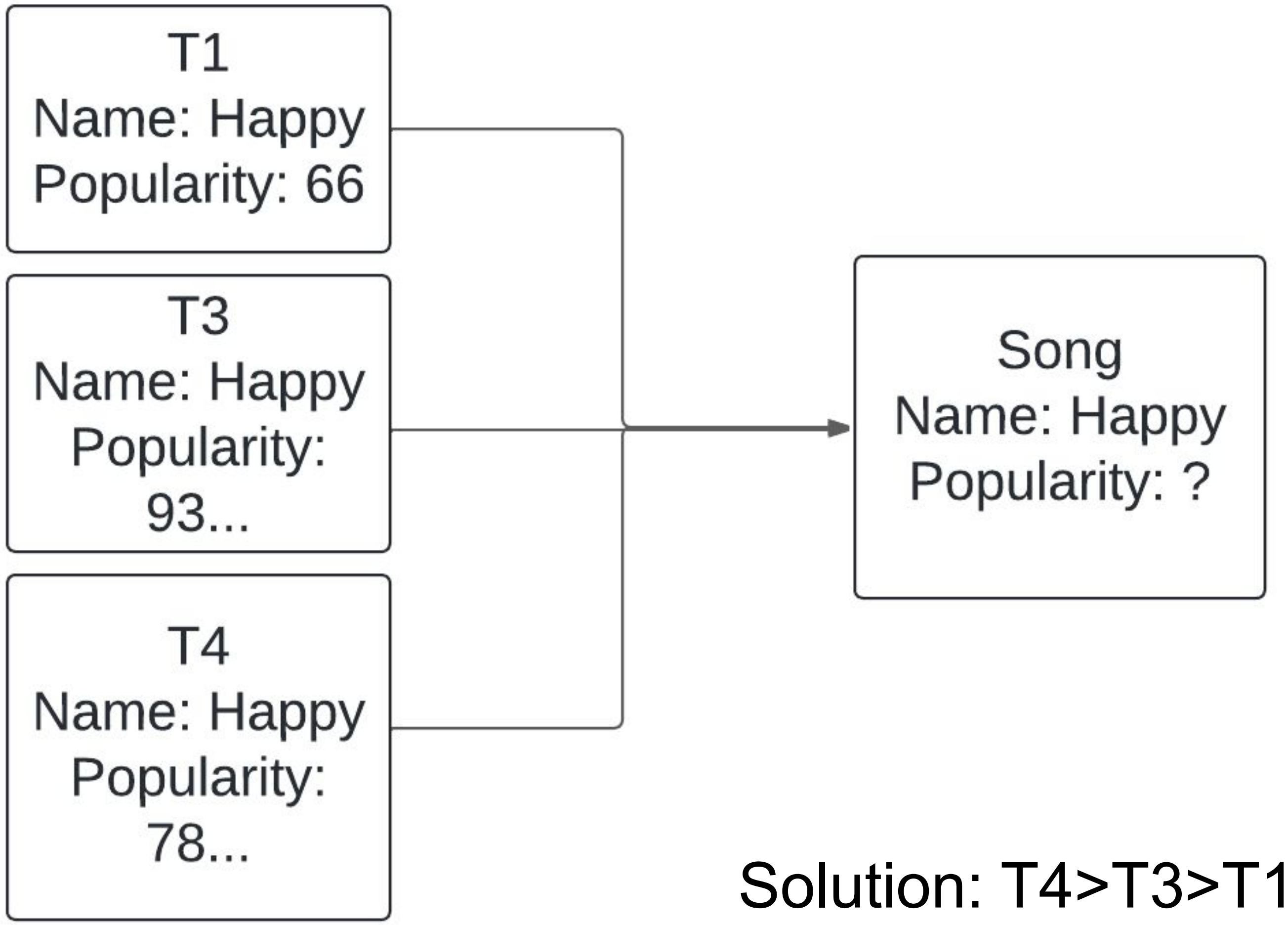
Mapping

Our mapping is based on a hierarchical order among the sources for common data. When the information about one song is present in more than one source we take the common fields from T4, then from T3 if missing in T4 and lastly from T1.

In addition, the information about the artists is combined from all the sources when possible.

Our mapping assertions are FOL-CQ GLAV mapping assertions (pairs <qs, qg> where qs is a FOL query and qg is a CQ).

Mapping problem: same fields, different values



Solution: T4>T3>T1

The result of the integration

We materialized the result of our Information Integration System as a Solution (not the Universal, of course, as we use SQL to materialize the result)

Missing information about in the sources was transformed into SQL Nulls (different from the Marked Nulls used in the course).

The information about the IDs was materialized using sequential integers.

Queries

We prepared 5 queries on the result of the integration (both in FOL and in SQL).

Moreover, we prepared an additional query on the Global Schema to run directly on the sources in a virtualized fashion.
We proved the correctness of the query and we produced a python script running the query both on our materialization and on the sources to empirically show it returned the same tuples (despite some encoding problems).