Data Science - Spotify 1 Million

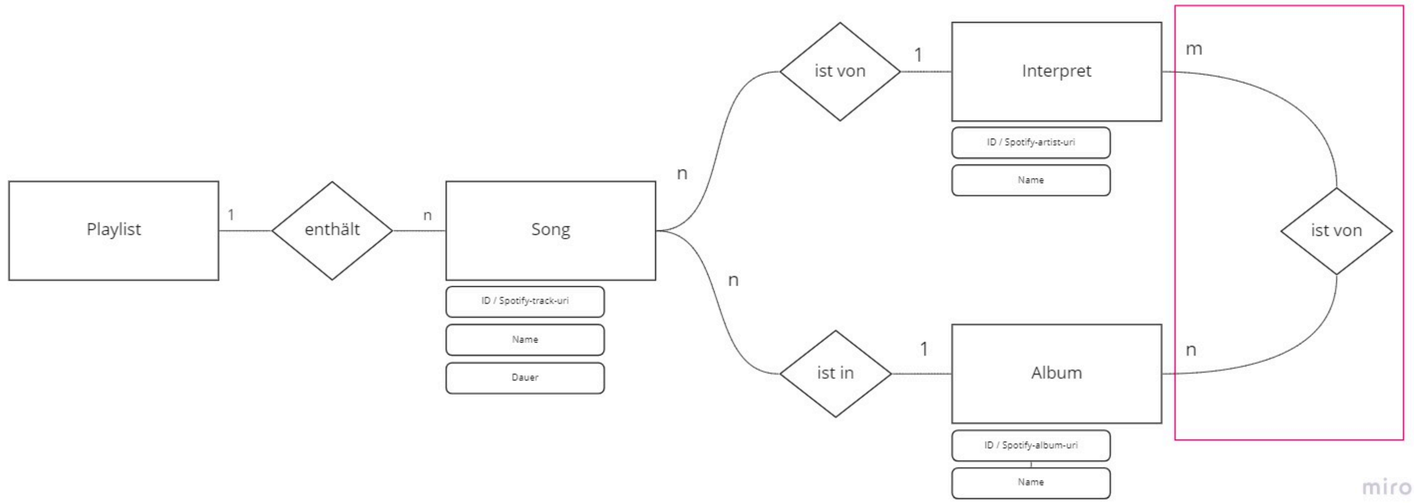
Zwischenpräsentation

Loreen Bies (1072354), Oliver Renth (1112441), Jan Ningelgen (1120440) – 21.12.2021

Konstruktion des ER-Models

- Interne Spotify-Uri oder dedizierten Primary-Key?
- Alle Daten speichern? Ist die Songdauer relevant?
- Welche Kardinalitäten gelten?

Version 1



Datenanalyse

Was muss die Datenbank abbilden können?

- Gibt es einen Song welcher Teil von mehreren Alben ist?
- Wie sind “feature”-Songs abgespeichert?
- Gibt es Alben von mehreren Künstlern?

Gibt es einen Song welcher Teil von mehreren Alben ist?

```python
from os import listdir
from os.path import isfile, join
import json

path = 'spotify_million_playlist_dataset/data/'

def for_song_in_playlist(tracklist, dictionary):
    for song in tracklist:
        if song['track_uri'] not in dictionary.keys():
            dictionary[song['track_uri']] = []
        album = song['album_uri']
        if album not in dictionary[song['track_uri']]:
            dictionary[song['track_uri']].append(song['album_uri'])

def for_playlists_in_file(json_data, dictionary):
    for playlist in json_data['playlists']:
        for_song_in_playlist(playlist['tracks'], dictionary)

def for_all_files(dir_path):
    count = 0
    track_album_dict = {}
    allfiles = [f for f in listdir(dir_path) if isfile(join(dir_path, f))]
    for file in allfiles:
        for_playlists_in_file(json.load(open(dir_path + file)), track_album_dict)
        count += 1
        if (not count % 50):
            print(count * 100 // len(allfiles), '%')
    return track_album_dict

di = for_all_files(path)

d = {k : len(v) for k, v in di.items()}
print(max(d.values())) # = 1
```

Nein.

Wie sind “feature”-Songs abgespeichert?

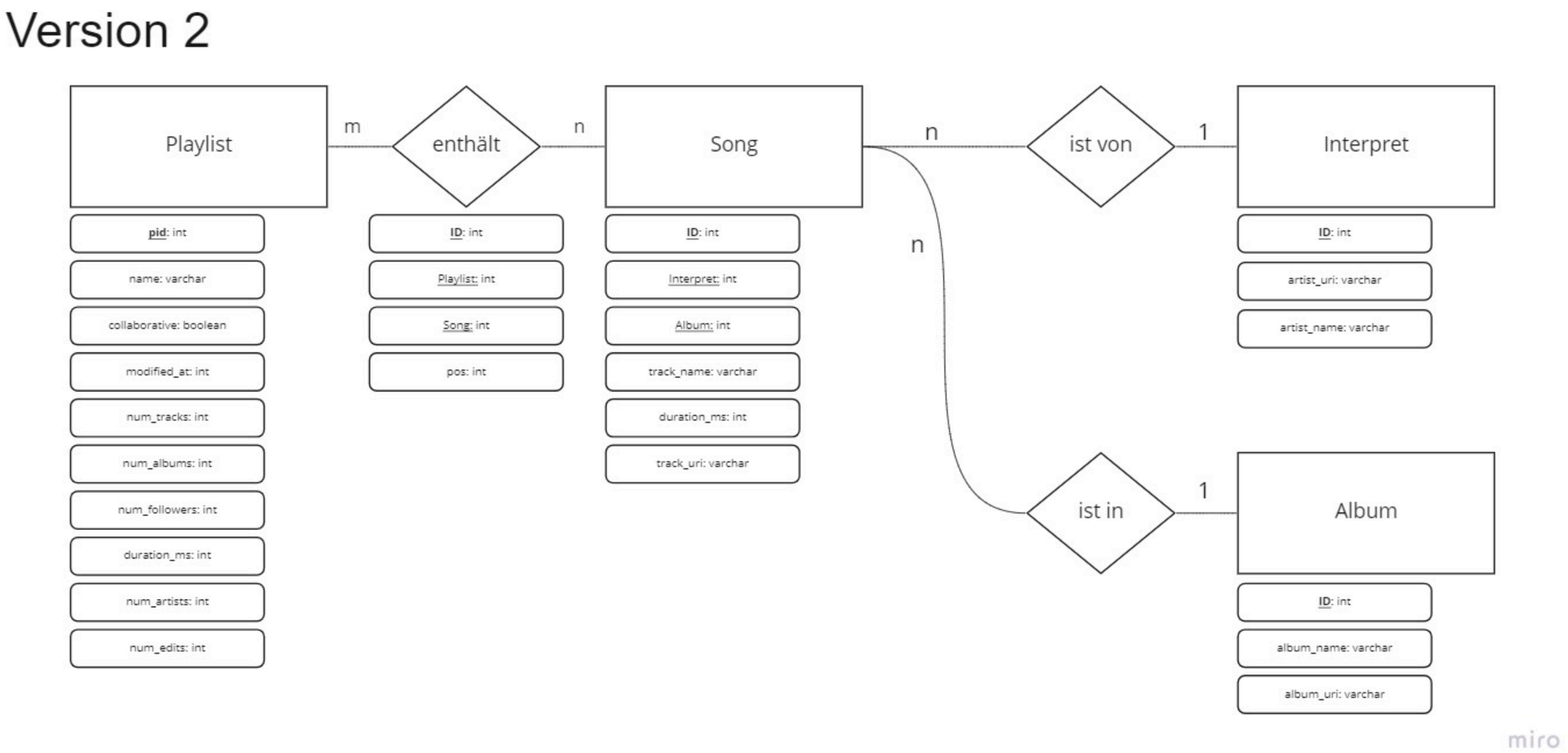
**Der Song wird für
jeden Künstler mit
einer eigenen Track-Uri
separat aufgeführt.**

Gibt es Alben von mehreren Künstlern?

Ja!

Beispiel:
Christmas 100 - 100 Great Christmas Hits and Classic Song

Resultat:


Version 2
Playlist
m
enthält
n
Song
n
ist von
1
Interpret
pid: int
name: varchar
collaborative: boolean
modified_at: int
num_tracks: int
num_albums: int
num_followers: int
duration_ms: int
num_artists: int
num_edits: int
ID: int
Playlist: int
Song: int
pos: int
ID: int
Interpret: int
Album: int
track_name: varchar
duration_ms: int
track_uri: varchar
ID: int
artist_uri: varchar
artist_name: varchar
n
ist in
1
Album
ID: int
album_name: varchar
album_uri: varchar
miro

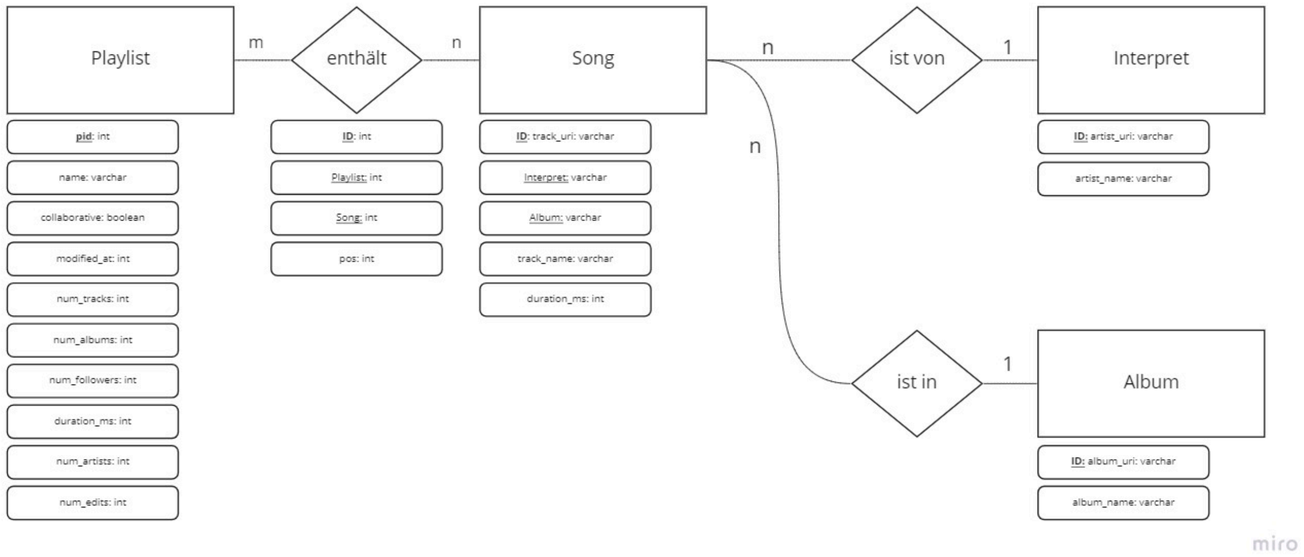
Datenbankimport

Und damit einhergehende Probleme

- Umsetzung eines dedizierten Primärschlüssels kompliziert
- Python-Listen können nicht alle Daten fassen
- Pandas-Dataframe ist zu langsam
- Psycog2 wird langsam bei Datenbankimport von großen Datenmengen

Lösungsansätze: Kein dedizierter Primärschlüssel

Version 3



Eintragen eines Fremdschlüssels schwierig, weil Referenz fehlt

Lösungsansätze: Python-Listen fassen nicht alle Daten

```
allFilenames = os.listdir(PATH)
for dieNaechsten10filenamen in tqdm(np.array_split(allFilenames, 100)):

    # Listen leeren:
    artists_dictList = []
    albums_dictList = []
    songs_dictList = []
    playlists_dictList = []
    playlist_enthaelt_song = []

    for filename in dieNaechsten10filenamen:

        # aktuelles File in Liste laden
        readFile2dictLists(filename)

    # die Listen aus den 10 Files zu Dataframes machen
    dataframes = listsToFrames()

    dataframesInDatenbankSchreiben()
```

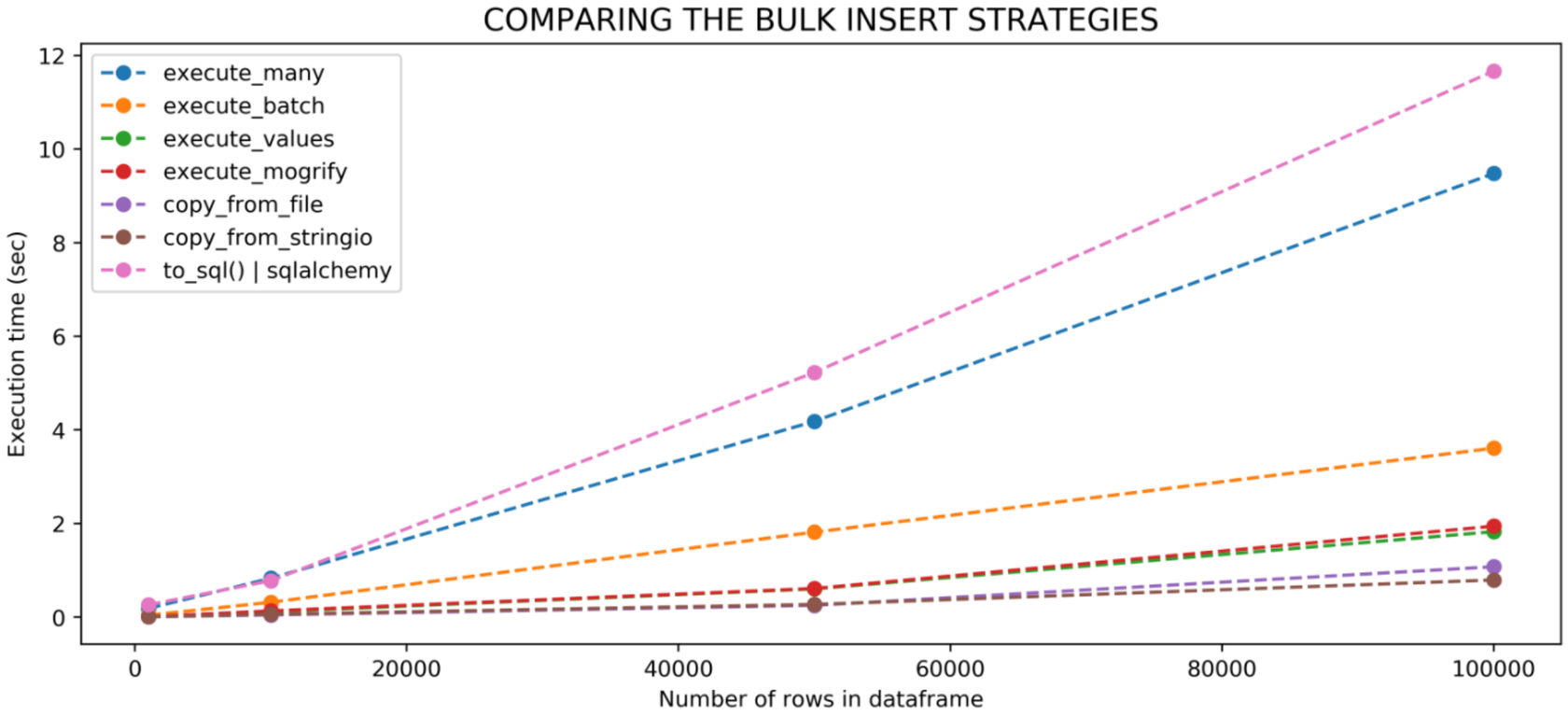
Schrittweiser Import der Playlists

Lösungsansätze: Pandas-Dataframe ist zu langsam

Append auf Pandas-Dataframe
ist sehr langsam

Stattdessen:
Python Listen und Dictionaries

Lösungsansätze: Import beschleunigen



https://naysan.ca/2020/05/09/pandas-to-postgresql-using-psycopg2-bulk-insert-performance-benchmark/

Lösungsansätze: Import beschleunigen

```python
def copy_from_stringio(conn, df, table):

    # dataframe als CSV in memory buffer speicher
    buffer = StringIO()
    df.to_csv(buffer, index=False, header=False, sep=";")
    buffer.seek(0)

    cursor = conn.cursor()
    try:
        # aus dem Memory Buffer per psycog2 in PSQL laden
        cursor.copy_from(buffer, table, sep=";")
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        sys.exit()
    cursor.close()
```

Ausblick

- PyTorch Überlegungen verworfen
- Start mit Content-based-filtering