Report - BDA - Grupo 15

1. Descrição do dataset e Design das Bases de Dados

O dataset escolhido pode ser adquirido em: https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset.

Este dataset consiste num conjunto de tracks do Spotify e as suas informações, como os artistas que produziram a track, o álbum a que ela pertence, a sua popularidade, etc..

Escolhemos este dataset, pois pensámos que poderíamos extrair 3 tabelas do mesmo com relações coerentes entre as mesmas, ainda que este apenas contenha 1 ficheiro csv.

O esquema da base dados MySQL é o seguinte:

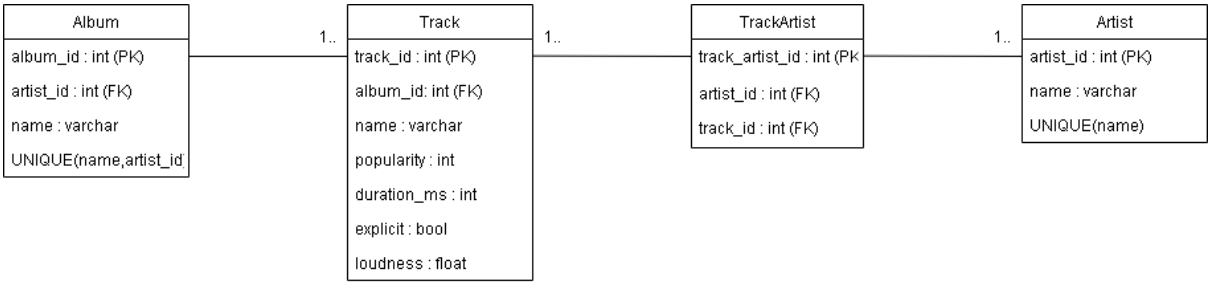

Figura 1. Esquema da base de dados MySQL

Há uma tabela para cada entidade relevante, ou seja, uma tabela para os artistas, álbuns e tracks. Como a relação entre Tracks e Artists é uma many-to-many (um artista pode ter várias tracks e uma track pode ter vários artistas), é necessária uma tabela intermediária (TrackArtist) para estabelecer o mapeamento entre elas.

O esquema da base dados MongoDB é o seguinte:

```
// Album Collection
{
    _id : ObjectId
    name : str
    artist : ObjectId
    tracks: [Object]
}
```

```
// Track Collection
{
    _id : ObjectId
    artists : [Object]
    name : str
    popularity : int
    duration_ms : int
    explicit : bool
    loudness : float
}
```

```
// Artist Collection
{
    _id : ObjectId
    name : str
}
```

Figuras 2,3 e 4. Esquemas das collections da base de dados Mongo

Existe uma collection por entidade (como em MySQL), ou seja, uma collection para os artistas, outra para os albúns e outra para as tracks. No entanto, como MongoBD é uma base de dados não relacional, não necessitamos de uma collection TrackArtists para relacionar as tracks com os artistas. Deste modo, cada álbum tem uma lista das suas tracks e cada track tem uma lista dos produtores (artistas).

2. Criação das Bases de Dados

A criação das bases de dados MongoDB e MySQL estão no ficheiro creating_dbs.ipynb. Antes de se começar a criar as tabelas/coleções, processaram-se os dados do dataset. O processamento dos dados consistiu em: selecionar as colunas pretendidas, remover linhas duplicadas, e nas strings que representavam os nomes dos artistas, das tracks e dos álbuns removeram-se caracteres especiais não reconhecidos pelo MySQL, espaços no início e fim, e foram transformadas de forma a ficarem todas apenas com letras minúsculas.

Para preencher as bases de dados, o processo foi praticamente o mesmo para ambas. Percorreu-se o dataset e iam sendo adicionados os valores dos registos a inserir na database em listas. No fim, tinham-se listas com os valores necessários para se efetuar a inserção, e era executado um execute_many/insert_many com essas listas. É relevante notar que para se respeitar as UNIQUE constraints, foram declarados sets que guardavam quais registos já tinham sido "vistos", de forma a ignorar repetidos para não inserir linhas repetidas nas tabelas/coleções.

3. Queries

As queries estão implementadas no ficheiro queries.ipynb

3.1. MySQL

A primeira e a segunda query são efetuadas usando a tabela das Tracks. Na primeira query são selecionadas as tracks que têm *loudness* entre -6.28 e -5.20 e são mostradas as colunas *name* e *loudness* por ordem crescente. Na segunda query são selecionadas as tracks que têm *popularity* superior a 80 e são mostradas as colunas *name* e *popularity* por ordem decrescente.

```
query= """
SELECT t.name, t.loudness
FROM tracks t
WHERE t.loudness
BETWEEN -6.28 AND -5.20 ORDER BY t.loudness ASC
"""
```

MySQL - Query 1

```
query= """
SELECT name, popularity
FROM tracks
WHERE tracks.popularity > 80
ORDER BY tracks.popularity DESC
"""
```

MySQL - Query 2

A terceira query é efetuada usando tabelas artists, albums e tracks da base de dados. São selecionados os artistas que têm álbuns com pelo menos 7 tracks e são mostradas as colunas name, album_name, popularidade média das tracks e o número de tracks do álbum. Os resultados são ordenados pela popularidade média das tracks do álbum por ordem decrescente.

A Query 4 é efetuada usando todas as tabelas da base de dados. Nesta query são selecionados os nomes dos artistas, o nome da sua track com features (com outros artistas) mais popular, o nome do álbum a que pertence e a popularidade dessa track. Os resultados vêm ordenados pela popularidade das tracks, por ordem decrescente.

A Query 5 é um update à tabela artists que modifica um nome de um artista existente (“dan berk” neste caso) pelo o nome “ new_artist_name”

```
query= """
SELECT a1.name, t1.name, al1.name, t1.popularity
  FROM artists a1
  INNER JOIN track_artists ta1 ON a1.id = ta1.artist_id
  INNER JOIN tracks t1 ON ta1.track_id = t1.id
  INNER JOIN albums al1 ON t1.album_id = al1.id
  WHERE t1.id IN (
    SELECT ta2.track_id
    FROM track_artists ta2
    GROUP BY ta2.track_id
    HAVING COUNT(*) > 1
    ) AND
    t1.popularity = (
    SELECT MAX(t2.popularity)
    FROM tracks t2
    INNER JOIN track_artists ta2 ON t2.id = ta2.track_id
    WHERE  ta2.artist_id = a1.id AND t2.id IN (
      SELECT ta3.track_id
      FROM track_artists ta3
      GROUP BY ta3.track_id
      HAVING COUNT(*) > 1
      )
    )
  GROUP BY a1.id, t1.id
  Order BY t1.popularity DESC
"""
```

MySQL - Query 4

```
query = """
SELECT a.name, al.name, AVG(t.popularity)
AS avg_popularity, COUNT(*) AS number_of_tracks
FROM artists a
INNER JOIN albums al ON a.id = al.artist_id
INNER JOIN tracks t ON al.id = t.album_id
GROUP BY a.name,al.name
HAVING COUNT(*) >= 7
ORDER BY avg_popularity DESC
"""
```

MySQL - Query 3

```
query= """
UPDATE artists
SET name = 'new_artist_name'
WHERE name='dan berk'
"""
```

MySQL - Query 5

As Querys abaixo são inserts em cada uma das tabelas da base de dados. Começa-se por inserir um artista na tabela artist, insere um álbum na tabela de álbuns com o novo artista inserido acima, insere uma track na tabela de tracks usando o novo álbum e artista inseridos, por fim insere na tabela track_artists a nova track inserida acima e o novo artista inserido para fazer a relação.

```
query = """
INSERT INTO artists (name) VALUES ('inserted_artist')
"""
mycursor.execute(query)

#insert an album of this artist with name "inserted_album"
query = """
INSERT INTO albums (name, artist_id) VALUES ('inserted_album', (SELECT id FROM artists WHERE name = 'inserted_artist'))
"""
mycursor.execute(query)

#insert a track of this album with name "inserted_track" and popularity 100, duration_ms 5000, explicit 1, loudness -5
query = """
INSERT INTO tracks (name, album_id, popularity, duration_ms, explicit, loudness) VALUES ('inserted_track', (SELECT id FROM albums WHERE name = 'inserted_album'), 100, 5000, 1, -5)
"""
mycursor.execute(query)

#insert in track artists the relation between the inserted track and the inserted artist
query = """
INSERT INTO track_artists (track_id, artist_id) VALUES ((SELECT id FROM tracks WHERE name = 'inserted_track'), (SELECT id FROM artists WHERE name = 'inserted_artist'))
"""
```

MySQL - Query 6

3.2. MongoDB

As Queries para a base de dados MongoDB, tem a mesma funcionalidade das correspondentes queries à base de dados MySQL.

```
mydoc = tracks.find({"loudness": {"$gte": -6.28,"$lte": -5.20}},{"name": 1,"loudness": 1,"_id": 0}).sort("loudness", 1)
```

MongoDB - Query 1

```
# query the name,duration,explicit,loudeness and popularity of tracks with popularity > 80, ordered by popularity in descending order
mydoc = tracks.find({"popularity": {"$gt": 80}}, {"_id":0,"name":1,"popularity":1}).sort("popularity", -1)
```

MongoDB - Query 2

```
pipeline = [
    {"$lookup": {
        "from": "artists",
        "localField": "artist_id",
        "foreignField": "_id",
        "as": "artist_obj"
    }},
    {"$unwind": "$tracks"},
    {"$group": {
        "_id": {"artist_name": "$artist_obj.name", "album_name": "$name"},
        "number_of_tracks": {"$sum": 1},
        "average_popularity": {"$avg": "$tracks.popularity"}
    }},
    {"$match": {"number_of_tracks": {"$gte": 7}}},
    {"$sort": {"average_popularity": -1}}
]
```

MongoDB-Query 3

```
pipeline = [
    {"$unwind": "$tracks"},
    {"$match": {"$expr": {"$gt": [{"$size": "$tracks.artists"}, 1]}}},
    {"$unwind": "$tracks.artists"},
    {"$lookup": {
        "from": "artists",
        "localField": "tracks.artists._id",
        "foreignField": "_id",
        "as": "artist_obj"
    }},
    {"$unwind": "$artist_obj"},
    {"$sort": {"tracks.popularity": -1}},
    {"$group": {
        "_id": "$artist_obj.name",
        "track_name": {"$first": "$tracks.name"},
        "album": {"$first": "$name"},
        "popularity": {"$first": "$tracks.popularity"}
    }},
    {"$sort": {"popularity": -1}}
]
```

MongoDB - Query 4

```
# update the artists with name "dan berk" to "new_artist_name"
artists.update_one({"name": "dan berk"}, {"$set": {"name": "new_artist_name"}})
tracks.update_many({"artists.name": "dan berk"}, {"$set": {"artists.$.name": "new_artist_name"}})
```

MongoDB - Query 5

```
# Insert an artist with name "inserted_artist"
artist_data = {"name": "inserted_artist"}
artist_id = artists.insert_one(artist_data).inserted_id

# Insert a track of this album with name "inserted_track" and other attributes
track_data = {
    "name": "inserted_track",
    "artists": [{"_id": artist_id, "name": "inserted_artist"}],
    "popularity": 100,
    "duration_ms": 5000,
    "explicit": 1,
    "loudness": -5
}
track_id = tracks.insert_one(track_data).inserted_id

# Insert an album of this artist with name "inserted_album"
album_data = {"name": "inserted_album", "artist_id": artist_id,"tracks": [track_data]}
album_id = albums.insert_one(album_data).inserted_id
```

MongoDB - Query 6

4. Otimizações

De forma a otimizar a performance das queries, foram utilizados indexes e foram feitas alterações a algumas queries. Foram criados índices nas colunas usadas nas queries, como por exemplo na coluna loudness da tabela/coleção tracks. Também foram acrescentados hashed indexes nas colunas relativas a ids, para tentar acelerar o processo dos JOINs. A criação dos indexes está codificada no ficheiro queries.ipynb. Nas queries complexas (Queries 3 e 4) em MongoDB, foram encontradas formas melhores de as escrever. Na Querie 3 executou-se o match no início em vez de no fim, de forma a filtrar as linhas mais cedo. Na Querie 4 removeu-se o lookup, visto ser possível obter o resultado pretendido sem ele.

```
pipeline = [
    {"$lookup": {
        "from": "artists",
        "localField": "artist_id",
        "foreignField": "_id",
        "as": "artist_obj"
    }},
    #match mais cedo
    {"$match": {"$expr": {"$gte": [{"$size": "$tracks"}, 7]}}},
    {"$unwind": "$tracks"},
    {"$group": {
        "_id": {"artist_name": "$artist_obj.name",
                "album_name": "$name"},
        "number_of_tracks": {"$sum": 1},
        "average_popularity": {"$avg": "$tracks.popularity"}
    }},
    # {"$match": {"number_of_tracks": {"$gte": 7}}},
    {"$sort": {"average_popularity": -1}}
]
```

MongoDB - Query 3 otimizada

```
pipeline = [
    {"$unwind": "$tracks"},
    {"$match": {"$expr": {"$gt": [{"$size": "$tracks.artists"}, 1]}}},
    {"$unwind": "$tracks.artists"},
    # {"$lookup": {
    #     "from": "artists",
    #     "localField": "tracks.artists._id",
    #     "foreignField": "_id",
    #     "as": "artist_obj"
    # }},
    {"$unwind": "$tracks.artists"},
    {"$sort": {"tracks.popularity": -1}},
    {"$group": {
        "_id": "$tracks.artists.name",
        "track_name": {"$first": "$tracks.name"},
        "album": {"$first": "$name"},
        "popularity": {"$first": "$tracks.popularity"}
    }},
    {"$sort": {"popularity": -1}}
]
```

MongoDB - Query 4 otimizada

Depois destas otimizações, foram feitas medidas para os tempos de execução das queries, tendo se obtido os seguintes resultados:

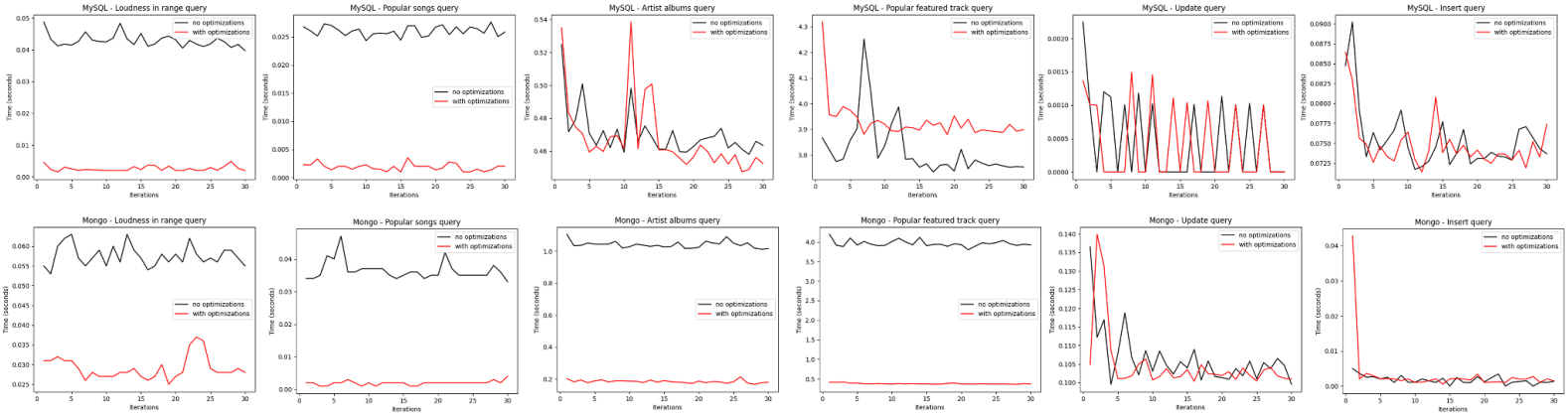

Figura 5. Plots com os tempos de execução de cada query antes e depois da otimização

Para as queries simples em MongoDB foi possível utilizar o método explain, que para além de informar o tempo da execução, também informa o número de documentos analisados durante a query. Antes das otimizações as Queries 1 e 2 analisavam 77698 documentos da coleção tracks. Após as otimizações, a Query 1 analisou apenas 10768 documentos, e a Query 2 apenas 470. Não verificamos para as queries 3 e 4, pois não foi possível executar a função explain() sobre os aggregates. Não foi possível obter o número de documentos analisados com o explain do MySQL.

5. Como replicar o projeto

No ficheiro create_dbs.ipynb está todo o processo para criar e popular as bases de dados MySQL e MongoDB. De forma a conseguir executar os passos, é necessário alterar as credenciais para aceder às bases de dados e ter as mesmas a serem executadas em background. Depois de inseridas credenciais válidas, basta fazer Run All e as bases de dados serão geradas automaticamente.

As queries e a definição dos indexes estão no ficheiro queries.ipynb. Para cada query há a possibilidade de correr a query uma vez e ver os resultados, e há a opção de correr a query 30 vezes (para se ter uma melhor noção da performance das queries), medindo os tempo de execução caso depois queira escrever um csv. Deve comentar e descomentar o código de acordo com o que pretende fazer.