Faculdade de Ciências da Universidade do Porto

Base de Dados (CC2005) 2021/22

Relatório do projeto de Base de Dados

Hugo Silva up202004447

João Malheiro up202004449

Tomás Ferreira up202006594

1. Introdução

Este relatório foi realizado no âmbito do trabalho prático da unidade curricular de Base de Dados. Com este projeto, procuramos criar uma base de dados (BD) que simula um sistema de streaming como o Spotify.

A escolha do tema foi algo unânime: todos nós tivemos a ideia de executar uma base de dados do Spotify foi determinante para a harmonia aquando da distribuição e realização das tarefas.

Começamos imediatamente por fazer um *brainstorm* de ideias acerca daquilo que pretendíamos incluir, identificando prontamente o que nas nossas opiniões são pilares essenciais para uma representação que nem o Spotify. A partir desse ponto analisamos as nossas ideias para coincidirem com os requisitos do trabalho, obtendo assim um grupo de conceitos a integrar no desenho conceptual da nossa BD, de modo a criar um Modelo Entidade-Relacionamento (ER). Tendo o Modelo ER feito, solidificamos as suas bases de modo a retirar o essencial para o primeiro passo dependente de um Sistema de Gestão de Bases de Dados (SGBD): o elaborar de um desenho lógico. Foi executado todo um mapeamento do posterior modelo conceptual (Modelo ER) para algo lógico e concreto, algo que já usufruiria de um SGBD: um Modelo Relacional. Após polir as impurezas do nosso Modelo Lógico (Relacional), passamos a um novo mapeamento, mas agora com base no Modelo Lógico com o objetivo de tentar extrair a sua natureza para um desenho físico interno ao SGBD. Por via do SQL, obtivemos assim uma representação real de um Modelo Físico. Por fim e após o máximo cuidado e aperfeiçoamento deste último modelo, avançamos para uma implementação de cariz físico da nossa BD, instanciando-a com alguns dados de modo a testar a sua eficiência.

2. Requisitos

O universo representado é de uma plataforma de Streaming de música denominada de Spotify, porém representaremos tal plataforma de uma forma mais simplificada. Para tal, esta plataforma necessita de, obviamente, músicas e por consequência, pessoas que ouçam ou publiquem tais músicas dentro da plataforma.

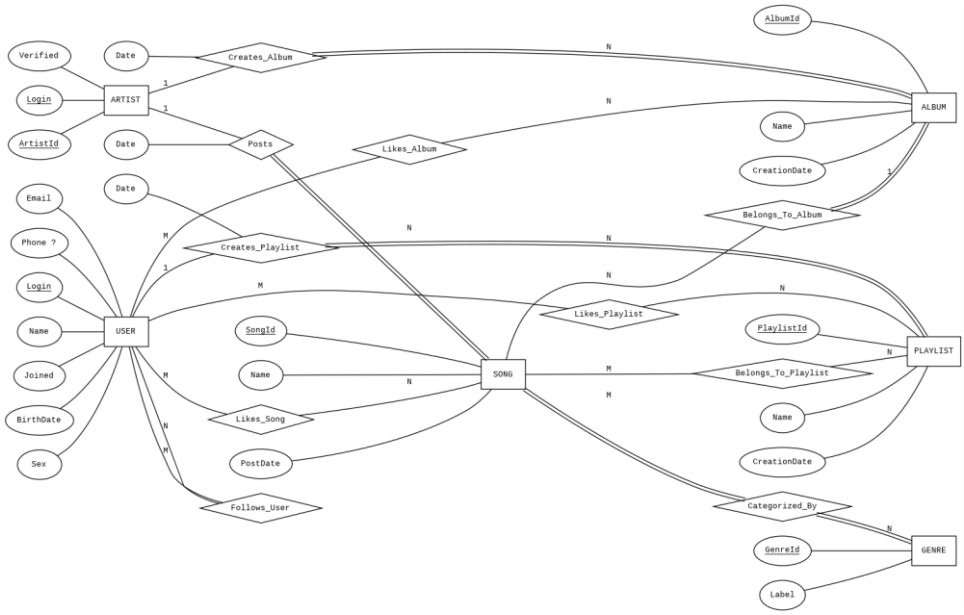
2.1 Modelo Entidade-Relacionamento

Tendo em conta o que foi dito em **Requisitos**, tivemos a necessidade de criar inicialmente **3 entidades-tipo, que** à medida que fomos evoluindo com as ideias do projeto acabaram por se tornar **6 entidades-tipo totais**:

- **User** : O centro da nossa base de dados. Responsável por ouvir e criticar músicas e criar **Playlists**. **Users** são distinguidos por número de Login diferente e único para cada um. Também possuem Nome, em junção com outros atributos.
- **Artist :** Um artista é nada mais que um **User** capaz de dar post a **Songs** e também capaz de criar **Albums.** Artista, porém, tem um id (**ArtistId**) específico e único que os identifica como, de facto, artistas.
- **Song :** Músicas são também um elemento crucial na nossa base de dados. Para identificação única, cada música possuí um **Id** único determinante.
- **Playlist:** Listas que podem ou não conter músicas variadas (não tendo em atenção se as músicas são do mesmo artista por exemplo). Ou seja, listas livres.
- **Album:** Listas que possuem um conjunto de músicas do mesmo artista (nem todas as músicas de tal artista necessitam estar no mesmo álbum).
- **Genre:** Caracterizam as músicas com um sistema parecido a tags. Uma música pode ter um ou vários géneros.

A partir dessas 6 entidades criamos 9 relacionamentos:

- **User Follows User**: Com participação parcial de ambas as partes já que users podem ou não seguir e ser seguidos por outros users e com cardinalidade **N:M**;
- **User Creates Playlist**: Com participação total por parte da playlist porque uma playlist tem de ser criada por um user e participação parcial por parte do user pois um user pode não criar playlists e cardinalidade **1:N** porque um user pode criar várias playlists, mas uma playlist só pode ser criada por um user;
- **User Likes:** O user pode gostar de diferentes músicas (**Likes_Song**), diferentes albums (**Likes_Album**) e ainda diferentes Playlists (**Likes_Playlist**). Estas relações seguem sempre a mesma lógica, ou seja, cardinalidade **M:N** com participação parcial em ambos os lados (Visto que um user pode ou não gostar de uma música, album ou playlist, e vários users podem gostar de várias músicas, albums ou playlists). São na totalidade 3 relações;
- **Artist Posts Song:** Com participação total da parte de Song e parcial de artista porque toda a song tem de ser produzida por um Artista mas um artista pode não ter songs e cardinalidade **1:N**. Decidimos também adicionar a esta relação um atributo de **Date** de upload;
- **Song Belongs_to:** Com participação parcial de ambas porque uma song pode não pertencer a uma playlist/album e uma playlist/album pode não ter songs e cardinalidade **N:M** porque uma playlist/album pode ter várias songs e vice-versa;
- **Artist Creates Album:** Com participação parcial da parte do artista porque um artista pode não ter albums e participação total por parte do album porque um album tem de ser de um artista e cardinalidade **1:N** porque um artista pode não ter albums como pode ter vários e um album só pode ser criado por um artista;
- **Song Categorized_by Genre:** Com participação total de ambas as partes porque uma Song tem de ter um género atribuído e um género tem de ser atribuído a uma Song e cardinalidade **N:M** porque uma Song pode ter vários géneros associados e um género é associado a uma ou mais Songs;

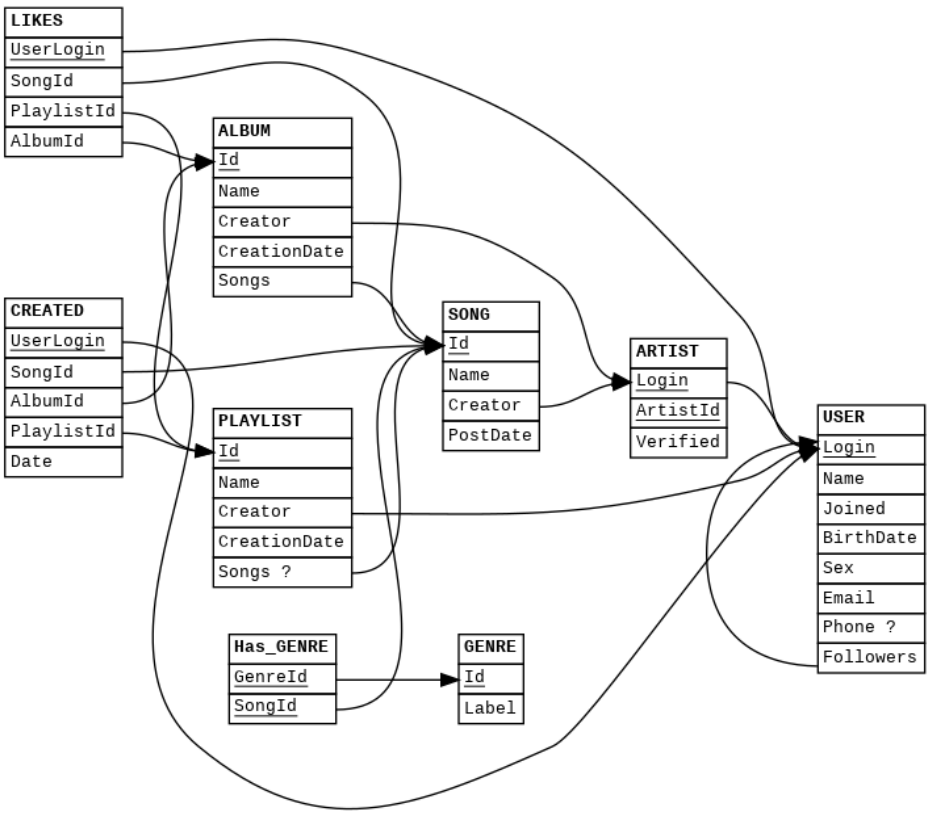


2.2 Transição de Modelo ER para Relacional

A transição entre Modelos Conceptual e Lógico é algo a se ser tratado com larga precaução, já que como foi referido o novo modelo fica agora dependente de um SGBD. Vamos explicar por detalhe todas as alterações dentro do esqueleto do modelo ER para o elaborar de um Modelo Relacional coeso.

Embora a maioria dos atributos em cada entidade se manteve, devido ao funcionamento do modelo Relacional, conseguimos adaptar várias relações e incorporar nas tabelas das entidades.

a. Um bom exemplo disso é a tabela **USER** que agora passa a possuir um atributo **Follower?**, que faz referência ao próprio **USER.Login**, visto que users podem seguir users, como foi visto no modelo ER;
b. O atributo **ARTIST.Login** passa a fazer referência a um **USER.Login**, visto que um artista é um **USER;**
c. Tabelas **SONG, ALBUM e PLAYLIST** passam a ter um atributo novo, **Creator**, que fazem referência ao criador da música **(ARTIST.Login)**, album **(ARTIST.Login)**, playlist **(USER.Login)**. Para além disso **ALBUM** e **PLAYLIST** passam a possuir **Songs** e **Songs?**, respetivamente, que fazem referência a **SONG.Id;**
d. Uma tabela nova **LIKES** que possui o equivalente às três relações **SONG_LIKE**, **ALBUM_LIKE** e **PLAYLIST_LIKE**. Esta tabela possui referências a **USER.Login** e evidentemente a **SONG.Id**, **ALBUM.Id** e **PLAYLIST.Id**;
e. A mesma ideia expressa em d. foi aplicada para a criação de músicas, albums e playlist. Possuímos agora uma tabela nova **CREATED** com as mesmas referências presentes em **LIKES**;
f. Uma tabela **HAS_GENRE** que serve de tabela de ligação entre **SONG** e **GENRE**, fazendo então referência a **SONG.Id** e **GENRE.Id**;



2.3 Transição de Relacional para SQL

A transição de modelo Relacional para SQL foi mais simples visto que o funcionamento do SQL se assemelha bastante a alguns dos comportamentos do modelo Relacional. Porém ainda foram feitas algumas alterações para que o acesso aos dados ficasse mais organizado e em geral, mais agradável de navegar.

a. Criação de tabelas **PLAYLIST_SONG** e **ALBUM_SONG**, que facilitam saber que músicas pertencem a uma certa playlist ou album;
b. Divisão da tabela **LIKES** em **SONG_LIKED**, **ALBUM_LIKED** e **PLAYLIST_LIKED** pois após testagens, a divisão de LIKES em três tabelas mostrou-se ser mais *user-friendly.*
c. Criação da tabela **FOLLOWER** para o mesmo efeito de facilitação de acesso já mostrado em a. e b.;

2.4 Considerações finais sobre volumes de dados

O código que será enviado já terá alguns dados introduzidos para que quem aceder a Base de Dados possa já testar e "brincar" um pouco. Embora o número de dados que introduzimos em SQL não passe das dezenas neste momento, esta Base de Dados foi desenhada para aceitar milhares de **USERS**, **SONGS**, **ALBUMS**, **PLAYLISTS**, **ARTISTS** e **GENRES**.