**Universidade Lusófona**

LEI, LEIT e LIG

Bases de dados

**Trabalho prático de avaliação continua**

José Aser Lorenzo

Luis Alexandre Gomes

2 de Novembro 2021

[Introdução 3](#_Toc4990)

[Descrição funcional 3](#_Toc4991)

[Modelo de dados 3](#_Toc4992)

[Trabalho solicitado 5](#_Toc4993)

[Regras 7](#_Toc4994)

[Cotações 8](#_Toc4995)

[Formato de entrega e prazo 8](#_Toc4996)

# Introdução

Este trabalho é uma continuação do que foi realizado na cadeira de AED no semestre passado. Vamos aproveitar o enunciado e os ficheiros de dados, fazendo as necessárias adaptações à cadeira de bases de dados. Os enunciados de AED são disponibilizados para referência.

Com este trabalho pretende-se criar um conjunto de tabelas, suas relações e restrições, para que sirvam de suporte a uma aplicação que implemente as funcionalidades abaixo descritas.

# Descrição funcional

A musica é uma paixão que não deixa ninguém indiferente. Com este trabalho pretende-se criar uma base de dados que sirva de suporte a uma aplicação que faça a gestão de um catálogo de musica.

Cada musica será registada com as propriedades que constam no trabalho de AED. Pretendemos também registar a participação de um artista na musica, nomeadamente se foi interprete, compositor da letra, compositor da música, orquestrador, ilustrador, se fez arranjos musicais, se foi o produtor, ou outros tipos de participação. Um artista pode ter uma ou mais participações na musica.

Uma musica pode ter várias versões, como por exemplo a versão normal e a acústica. Para simplificação vamos considerar que as várias versões serão armazenadas como musicas diferentes, existindo uma tabela que as vai relacionar. Essa tabela pode também ser usada para armazenar outro tipo de relações, por exemplo entre musicas que partilham a mesma letra.

Também se pretende ter uma lista de álbuns, sendo cada album formado por uma ou mais faixas de musica. Quando uma musica tem várias versões, no album participa a versão que foi publicada. Existem alguns em que duas versões da musica são publicadas, razão pela qual as diferentes versões da musica são armazenadas como se fossem musicas diferentes.

Os utilizadores gostam de rotular as musicas, pelo que optámos por um mecanismo bastante aberto para o fazer. Uma tabela guarda os rótulos, que depois são livremente atribuídos às musicas. Exemplos de rótulos são: Pop, Rock, New Age, Clássica, Instrumental, Épica, New Age e Acústica. Na tabela de rótulos podem ser adicionados os que entender. A nossa base de dados vai também permitir rotular os artistas, usando o mesmo mecanismo.

# Modelo de dados

Para suportar este problema apresentamos o seguinte modelo de dados:

# Trabalho solicitado

Descrição geral do trabalho:

* Carregar os ficheiros de texto para as tabelas LOAD com o mesmo nome do ficheiro;
* Limpar dados carregados;
* Normalizar os dados carregados;
* Carregar os dados nas tabelas definitivas, respeitando as relações definidas no modelo; • Executar queries do trabalho de AED sobre o modelo final;

## Etapa 1

1. Carregar os ficheiros fornecidos na cadeira de AED para dentro das tabelas com o mesmo nome. No MySQL pode usar o comando LOAD para carregar os ficheiros. Todos os SGBDs têm utilitários de importação, como o LOAD anteriormente referido, o IMPDP do Oracle e o Import do SQL Server. O DBVisualizer também tem um utilitário de import universal. No nosso caso particular, como os ficheiros têm menos de 200000 linhas, também é possível carregar os ficheiros para dentro de uma folha Excel e, usando formulas, produzir um comando INSERT, que depois é copiado para o cliente e executado na BD.
2. Criar índices nas colunas musica\_id para acelerar as operações que vão ser feitas nos passos seguintes. Estes índices devem aceitar valores repetidos;
3. Limpar os dados carregados:
   1. 1.Retirar espaços no inicio e no fim;

3.2.Retirar “ no inicio e no fim;

3.3.Retirar [ no inicio e ] no fim;

3.4.Determine quantos artistas existem dentro de cada linha de load\_artists. Ordene por ordem decrescente do contador de artistas;

1. Normalizar os dados carregados:
   1. 1.Verificar se existem dados repetidos em musica\_id nas 3 tabelas de load;

4.2.Eliminar dados repetidos nas 3 tabelas de load. Para isso tem que ser adicionada uma coluna com um identificador temporário, que vai distinguir todas as linhas. Em Oracle podemos usar um group by para detectar os valores repetidos, sendo depois eliminado o id\_temp mais elevado para cada music\_id num subselect. O MySQL e o SQL Server possuem um DELETE JOIN, que com o identificador único permite eliminar as repetições;

## Etapa 2

1. Continuar com a normalização:

5.1.Detectar musica\_id de load\_song\_detail que não existem em load\_song e eliminar;

5.2.Juntar musica com detalhe de musica usando o musica\_id que já não tem valores repetidos. Só devem aparecer dados que existam nas duas tabelas;

5.3.Em load\_song\_artists adicione a coluna artista\_id e atribua valores com incrementos de 100 unidades. Em Oracle usar sequências e em MySQL usar variáveis de sessão;

5.4.Encontrar um mecanismo que permita listar o primeiro artista, ou o segundo, ou o nésimo, juntamente com com musica\_id e artista\_id inserido no passo anterior. O artista da coluna 1 será artista\_id+1, o da coluna 2 será artista\_id+2, etc. Foi por isto que os valores de artista\_id têm incrementos de 100 unidades entre linhas. Este query será usado para insert na tabela artista;

5.5.Escrever query que mostra o artista\_id com musica\_id. Começamos pelo artista que está na primeira posição, depois na segunda e assim sucessivamente até à posição n. Este query será usado para inserir dados em contribuição de artista;

1. Criar todas as tabelas do modelo incluindo restrições PK, UK, NOT NULL, CHK e FK. Defina as que entender necessárias tendo em conta o modelo apresentado.
2. Inserir dados na tabela musica executando o query que junta load\_song com load\_song\_detail. Use a instrução INSERT SELECT que permite inserir na tabela destino o resultado de um select;
3. Inserir dados na tabela artista executando o query que seleciona o primeiro artista e o respetivo numero. Repita para o segundo, terceiro até chegar ao n-ésimo. Use a instrução INSERT SELECT que permite inserir na tabela destino o resultado de um select;
4. Inserir dados nas restantes tabelas excepto contribuição artista. Use a descrição do enunciado e adicione 5 álbuns cada um com 5 faixas à sua escolha.

## Etapa 3

10.Detecte nomes de artistas repetidos na tabela artista e elimine. Para isso use a mesma técnica que usou anteriormente para limpar valores repetidos de “musica\_id", mas neste caso não precisa criar coluna com identificador, visto que já tem o artista\_id;

11.Desenvolva um query que permita detectar se todos os valores de musica\_id em load\_song\_artists existem na tabela musica. Os que não existirem devem ser eliminados;

12.Desenvolva um query que junta as tabelas “load\_song\_artists” e “artista” usando o nome do artista como critério de junção. No passo anterior ficou garantido que não temos nomes repetidos. Este tipo de junção vai ser muito lenta, pelo que necessita criar um índice sobre o campo “nome\_artistico”, que será removido quando terminar o passo 11. Este query deve ser executado sobre o nome que está na posição 1, depois na posição 2, até à posição n;

13.Insira dados em contribuição de artista usando o query anterior. O campo musica\_id vem de “load\_song\_artista” e o campo artista\_id vem de artista.

14.Produza as seguintes pesquisas, que constam do enunciado de AED:

14.1.Recebendo um ano, contar o número de canções editadas nesse ano;

14.2.Contar o número de musicas editadas por cada ano;

14.3.Contar o número de músicas com o mesmo titulo;

14.4.Adicionar uma coluna na tabela musica para guardar a duração em Minuto:segundo

14.5.Associar 20 rótulos a pelo menos 10 artistas diferentes, pelo alguns artistas têm mais que um rótulo. Fazer uma pesquise que mostre as associações que foram feitas, indicando o nome do artista e os rótulos que lhe foram associados, apresentando um rótulo por linha. Usando PL/SQL ou uma função especial pode alterar o resultado para apresentar todos os rótulos de um artista na mesma linha;

14.6.Escreva um query que conte o número de vezes que cada rótulo é usado, mostrando o resultado por ordem decrescente;

14.7.Dado um inteiro N, e os anos X e Y, escrever um query que retorna as N músicas mais dançáveis entre os anos X e Y (inclusive);

14.8.Retorna todos os artistas que num dado período apenas lançaram uma música, ordenados alfabeticamente pelo nome do artista.

14.9.Considerando dois números inteiros A e B, retornar uma lista com os N artistas que mais temas musicais têm, considerando apenas aqueles que têm entre A e B temas (ambos inclusive). Os artistas devem ser devolvidos por ordem decrescente do número de temas. Caso não existam N resultados, devem ser devolvidos os que existirem;

14.10.Devolver os nomes dos rótulos que existam na base de dados associadas a Artistas que tenham pelo menos uma música entre os anos A e B (ambos inclusive). A lista deve estar ordenada por ordem decrescente do número de ocorrências de musicas;

14.11.Produza um query que devolve o nome do artista e o ano em que publicou musica. O resultado deve ser ordenado por artista e ano. Adicione um filtro que mostra apenas resultados entre os anos A e B. Usando PL/SQL deve verificar se, para cada artista, há publicações em todos os anos do intervalo, escrevendo esses artistas numa tabela temporária. Juntando a tabela temporária com os artistas e as musicas, calculando uma média da popularidade do artista tendo em conta a popularidade das suas musicas. Mostre os artistas que publicaram consecutivamente entre os anos A e B ordenando o resultado de forma decrescente de popularidade média.

14.12.Faça um query que conta o número de musicas de cada artista e guarde numa tabela temporária os nomes dos artistas que têm mais de M temas (por exemplo 10). Criar uma tabela que tem duas colunas: palavra e contador. Usando PL/SQL vai contar o número de ocorrências de palavras nos nomes dos artistas que estão na tabela temporária, atualizando o contador de palavras. Apenas devem ser consideradas as palavras que tenham tamanho superior a L (por exemplo 3). No fim, produza um query que determina as N palavras que mais vezes ocorrem, apresentando o resultado por ordem decrescente de ocorrência de palavras;

# Regras

O modelo de dados apresentado anteriormente deve ser usado para o trabalho solicitado.

Terão que ser respeitados os nomes das tabelas e das colunas, assim como os respectivos tipos de dados. Se encontrar alguma falha ou pretender introduzir uma melhoria, apresente a sugestão ao professor.

Em cada etapa deverá ser entregue um ficheiro de texto contendo os comandos SQL que implementam o que é solicitado. Esse script poderá ser feito com a sintaxe MySQL/ MariaDB, MS-SQL Server ou Oracle, não sendo aceites scripts feitos noutros SGBDs.

Os comandos do script devem seguir a sequência dos desafios, o que significa que o script da etapa 2 tem que aproveitar os objetos criados na etapa 1 e não reconstruir tudo desde o início. Por exemplo, se na etapa 1 é criada uma tabela e na etapa 2 é pedida uma alteração à tabela, terá que resolver o problema recorrendo a um ALTER TABLE e nunca a um DROP seguido de CREATE.

Os comandos que respondem a um desafio devem ser precedidos por um comentário que identifique o respectivo número de desafio. A ordem dos desafios também tem que ser respeitada, porque segue uma sequência lógica. Um desafio não respondido deve ter na mesma o respectivo comentário identificativo.

Todos os comandos devem ser testados no respetivo SGBD, pelo que não podem ocorrer erros na sua execução. Quando os desafios estão encadeados, um erro na etapa precedente anula todas as que dela dependem. Por exemplo, se tiver um erro na criação das tabelas, não será possível avaliar a sua alteração, inserção de dados nem as suas consultas.

# Cotações

|  |  |
| --- | --- |
| **Etapa** | **Cotação** |
| 1 | 24% |
| 2 | 28% |
| 3 | 48% |

# Formato de entrega e prazo

Os 3 scripts deverão ser compactados usando o formato ZIP ou 7z, e o ficheiro resultante será submetido no Moodle.