Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Letivo de 2020/2021

Movie Database

Ana Catarina Canelas, Ana Filipa Pereira, Carolina Santejo, Raquel Costa

Dezembro, 2020

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Movie Database

A93872, A89589, A89500, A89464

Dezembro,2020

Dedicatória

Em primeiro lugar, gostaríamos de prestar o nosso agradecimento ao Professor Orlando Belo pela organização desta Unidade Curricular de Base de Dados, em especial ao esforço realizado na sua adaptação à distância, face ao contexto pandémico atual da COVID-19 em que vivemos. Com sucesso conseguiu desafiar-nos e criar oportunidades para podermos desenvolver as nossas capacidades.

Gostaríamos também de agradecer ao Professor João Coelho pelo acompanhamento ao longo das aulas Práticas, que apesar da sua recente posição como Docente, conseguiu transmitir-nos todo o seu conhecimento com sucesso e ainda orientar-nos ao longo do nosso trabalho.

Por fim, gostaríamos também de agradecer aos nossos colegas de curso pelo seu contributo para o levantamento de dados do tema em causa neste trabalho, e que, além disso, proporcionaram um ambiente favorável à aprendizagem ao longo deste semestre.

“Teamwork is the ability to work together toward a common vision. The ability to direct individual accomplishments toward organizational objectives. It is the fuel that allows common people to attain uncommon results.” - Andrew Carnegie

Resumo

Ao longo deste projeto, foi desenvolvida uma base de dados de filmes, semelhante ao IMDB, cujo objetivo principal é a catalogação e avaliação de filmes por parte dos usuários. Esta implementação visa representar todo o processo desde a adição de um novo filme na BD, a adição de um novo *user*, a contemplação de todas as suas avaliações , a atribuição de prémios e nomeações, além de se ter em conta também, todas as pessoas que participam num dado filme, sejam atores ou então membros da *crew*, que desempenham outras funções.

Este projeto foi desenvolvido por etapas que serão detalhadas ao longo deste relatório.

Inicialmente, foi feita a análise e o levantamento de todos os requisitos essenciais ao desenvolvimento do sistema. Com isto, foi possível, utilizando o sofware "BRModelo" desenvolver o modelo conceptual de acordo com a metodologia abordada nas aulas. Após isto, no "MySQL Workbench" desenvolvemos o modelo lógico, tendo em conta as regras de mapeamento ER, e, a partir do script gerado (referente á criação da base de dados e das suas tabelas), foi feita a implementação física da nossa BD.

Área de Aplicação: <<Arquitetura, Desenvolvimento de Bases de Dados.>>

Palavras-Chave: << Base de Dados Relacional, "MySQL Workbench", "BRModelo", Análise de Requisitos, Modelo concetual, Modelo lógico, Modelo físico, ER, Mapeamento, Índices, *Forward engineering*, Vistas, Taxa de crescimento, *Queries,* Sistema de Gestão de Base de Dados >>

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Ao longo do último século, o mundo evoluiu mais do que em dez mil anos de história. E esta evolução está indissociavelmente ligada ao crescimento da tecnologia que é, atualmente, indispensável, como por exemplo na medicina, no ensino, na robótica entre outros. No entanto, algo que evoluiu e marcou gerações foi a indústria do entretenimento. Ao longo do século passado foram surgindo novas invenções, como as TV's ou as consolas de jogos, que mudaram por completo o modo de vida das pessoas. Mas se há algo que podemos afirmar, é que em pleno século 21 a grande revolução reside nos serviços de *streaming*. Poucos são aqueles que não conhecem o Spotify, a Netflix ou a HBO que fornecem um catálogo quase infinito de opções, seja elas músicas, filmes ou séries (entre muitos outros). Além disto, a crescente popularidade destes serviços, leva-os a grandes investimentos no seu conteúdo. A título de curiosidade, a Netflix investiu 17.3 mil milhões de dólares em conteúdo, apenas em 2020.

Tendo isto em conta, a escolha do nosso tema para este trabalho prático, baseou-se, essencialmente, no que acabou de ser referido. Visto que os serviços de *streaming*, que possuem milhares de opções para os seus milhões de utilizadores, que, naturalmente, têm diferentes preferências, surgiram assim, ferramentas cujo objetivo principal é auxiliar a pessoa na escolha de um filme ou série tendo em conta os seus gostos, além de guardar e gerir informações sobre os mesmos. Sejam preferências pelo género do filme, pelo cast, membros do *backstage*, pelos melhores ratings ou pelas premiações, sistemas como o IMDB (principal referência deste projeto), que segundo estudos estatísticos conta com cerca de 250 milhões de acessos mensais, selecionam uma lista de opções que satisfazem todos os requisitos do utilizador. Assim sendo, estas ferramentas revelam-se bastante úteis para uma pesquisa rápida e eficaz de diversos filmes ou séries, uma boa gestão das suas avaliações e das suas informações (por exemplo data de estreia, orçamentos, nomeações, premiações, membros do cast, etc.) além de desenvolverem uma estrutura segura e eficiente de gestão de dados capaz de suportar o constante surgimento de novos filmes, bem como o aumento de usuários que interagem com o sistema.

1.2 Apresentação do Caso de Estudo

Uma das grandes referências para a escolha e criação deste projeto foi o IMDB (*Internet Movie* Database*).* A pesquisa de filmes ou séries usando os mais diversos filtros, bem como as suas avaliações tornou-se algo bastante útil tendo em conta a quantidade de opções que surgem todos os dias. No entanto, o grupo achou mais relevante e interessante gerir informações de filmes que estão presentes apenas num único serviço de streaming, e não em vários. Isto deveu-se ao facto de que será muito mais útil para os *users* do *stream service* em questão (por exemplo Netflix), uma vez que a ferramenta apenas gere informações de filmes presentes no serviço pelo qual pagam e não em outros.
Deste modo, ao longo desta primeira fase do projeto, decidimos implementar uma base de dados capaz de gerir filmes, todas a sua informação, os usuários e as suas avaliações. No entanto é preciso referir que ferramentas como o IMDB são extremamente complexas além de lidarem com dados cuja quantidade encontra-se na ordem dos milhares, além de abranger milhões de usuários mensalmente. Deste modo, o nosso projeto será uma versão muito simplista destes tipos de sofware, contemplando apenas uma porção dos dados que eles possuem. No entanto, tivemos o cuidado de que a nossa base de dados fosse o mais realista possível independentemente de ser simples, contendo todos os dados que achamos mais relevantes.

1.3 Fundamentação da implementação da Base de Dados

Como já foi referido, atualmente, os serviços de streaming disponibilizam aos seus consumidores um catálogo quase infinito de filmes e séries. Visto que este catálogo possui milhares de opções, surgiu a necessidade de organizar e filtrar as possibilidades de acordo com critérios, como por exemplo o género do filme, as suas premiações ou ratings de outros usuários. No entanto, é percetível que ao lidarmos com dados cuja quantidade está na ordem dos milhares ou até mesmo milhões, torna-se inviável a utilização de métodos rudimentares para os gerir. Seria impensável escrever manualmente ou utilizar uma folha de cálculo para armazenar e gerir todo o catálogo da Netflix, por exemplo. Não só esta metodologia seria bem mais trabalhosa, mas também seria mais propícia a erros e perdas de informação. A utilização de uma base de dados, por etapas, começando com o modelo conceptual e terminando com a sua implementação física, permite uma melhor perceção do projeto na sua globalidade. Além disto, a BD permite uma gestão mais autónoma dos dados, fornece mecanismos de segurança essenciais e tem em conta a necessidade da constante adição de dados.
Concluindo, uma base de dados é uma alternativa, mais eficaz e segura comparada a outros métodos tradicionais de armazenamento e gestão de informação, sendo isto a motivação para o desenvolvimento deste trabalho prático.

1.4 Análise da Viabilidade do Processo

Ao longo deste projeto, o grupo tentou, através de diversos métodos, obter a informação essencial para que o resultado fosse o mais fidedigno e realista, independentemente da sua simplicidade. A nível de requisitos, foram feitas pesquisas constantes, nomeadamente em websites ou outras plataformas, cujo objetivo é semelhante ao da nossa BD. A nível da criação e implementação da base de dados tivemos sempre o cuidado de garantir que as metodologias usadas correspondiam às que foram abordadas nas aulas teóricas. Além disto, todas as dúvidas que se levantaram foram esclarecidas pela equipa docente que também teve a disponibilidade de nos dar feedback ao longo de todo o processo. Isto foi essencial para que o grupo tivesse confiança no trabalho e para que todo o processo seja viável.

1.5 Estrutura do Relatório

Este relatório visa explicar de forma clara e sucinta todo o raciocínio por detrás do desenvolvimento da nossa base de dados, desde a análise de requisitos até á sua implementação física. De forma a melhor explicar todo este processo dividimos o relatório em várias etapas:

- Na secção **Definição do Sistema** apresentamos o caso de estudo deste trabalho, uma contextualização do projeto, bem como a fundamentação da implementação da BD, ou seja, os motivos que justificaram a sua criação.
- Na secção **Levantamento e Análise de Requisitos** são apresentados os vários métodos de levantamentos de requisitos, seguindo-se a exposição dos requisitos de descrição, exploração e de controlo e terminando com a sua análise e validação geral.
- Na secção **Modelação Conceptual** são apresentadas todas as entidades do sistema, os respetivos atributos e os relacionamentos entre entidades (que poderá ter ou não atributos também). Isto é acompanhado com uma justificação detalhada dos tipos de dados escolhidos para cada atributo, bem como a justificação para eles serem opcionais, multivalorados etc. Por fim, é apresentado todo o modelo e é feita uma validação geral.
- Na secção **Modelação Lógica** é apresentada a proposta de modelo lógico criada pelo grupo, bem como uma explicação detalhada de cada uma das tabelas, dos relacionamentos entre elas e da escolha das chaves primárias. O modelo é validado através da normalização e é posteriormente reavaliado tendo em conta interrogações levantadas pelo utilizador.
- Na secção **Implementação Física** é apresentado o Sistema de gestão de base de dados escolhido bem como o modelo lógico convertido para esse mesmo Sistema. Além disto, as interrogações levantadas pelo utilizador, as 'queries' foram transcritas para o SQL. Posteriormente é apresentada a estimativa do espaço do disco de base de dados e a taxa de crescimento anual.
- Na secção **Conclusão e Trabalho Futuro** é feita uma autoavaliação do nosso projeto bem como possíveis novas funcionalidades da nossa BD.

2. Análise e Levantamento de Requisitos

2.1. Método de levantamento e de análise de requisitos adotado

O objetivo deste projeto consistiu no planeamento e implementação de uma Base de Dados cujo objetivo foi a gestão e organização de dados e informações relativas a filmes, além de gerir também vários usuários que fazem avaliações desses mesmos filmes e que interagem com o sistema. Deste modo, foi necessário recorrer a algumas metodologias de forma a reunir os requisitos essenciais para a realização de um trabalho realista e aceitável. Explicaremos de seguida os métodos usados:

- **Pesquisas e observações**: Como já foi referido, o IMDB foi uma das grandes referências para o desenvolvimento do nosso projeto. Desta forma, foram feitas várias pesquisas nesse site, e em outros com o mesmo efeito, por exemplo, *LetterBoxd*, *Rotten Tomatos, MetaCritic*, etc, analisando as suas funcionalidades e a informação que gerem. Com isto, escolhemos aquelas funções que consideramos mais interessantes, além de escolhermos as entidades e respetivos atributos que consideremos mais relevantes.

- **Entrevistas**: Visto que grande parte dos jovens utiliza serviços de streaming, e que o projeto consiste em armazenar filmes e gerir avaliações de usuários, decidimos perguntar a alguns colegas de curso, se caso fossem utilizar o nosso sistema que tipo de funcionalidades gostariam de encontrar e que informação relativa a filmes lhes interessaria saber. Desta forma, e tendo em conta toda a informação reunida, fizemos uma seleção daquilo que mais gente preferiu.

2.2. Requisitos Levantados

2.2.1 Requisitos de Descrição

- Um utilizador, quando se regista, terá de fornecer um username, um email e uma password. É lhe atribuído um nível inicial pré-definido.
- Quantos mais comentários tiver um usuário, maior será o seu nível.
- Quando a entidade competente adiciona um filme á base de dados terá de fornecer o nome desse mesmo filme, receita, duração, PG, data de estreia, país, língua, custo, o género e uma descrição.
- Um filme pode ter mais do que um género.
- Vários utilizadores podem ter uma conta.
- Vários filmes podem estar na base de dados.
- Quando se adiciona uma função é obrigatório especificar uma designação.
- Quando se adiciona uma pessoa que participa num filme (seja ator ou qualquer outra função), é necessário fornecer nome e género. Fornecer a data de nascimento é opcional .
- Quando um utilizador faz uma *review* de um filme terá de comentar e dar uma avaliação numérica de 1 a 10.
- Um utilizador apenas poderá avaliar um filme uma única vez.
- Um utilizador poderá avaliar quantos filmes quiser.
- Um utilizador tem acesso às informações dos filmes que estão no sistema.
- Uma pessoa que atua num filme, tem obrigatoriamente uma personagem e um protagonismo. O salário poderá ser conhecido ou não.
- Uma pessoa desempenha uma função num dado filme e o salário recebido poderá ser conhecido ou não.
- Quando se adiciona um prémio é necessário fornecer um nome e uma categoria.
- Uma pessoa que atua num filme ou desempenha qualquer outra função poderá ser nomeada a um ou vários prémios.
- Uma pessoa poderá vencer um prémio de carreira (este prémio não está relacionado a nenhum filme).
- Um filme poderá ser nomeado a um ou mais prémios.
- De uma nomeação, num determinado ano, é obrigatório surgir sempre um vencedor (filme e/ou pessoa).

2.2.2 Requisitos de Exploração

- Um utilizador poderá ter acesso á informação (nome, data de nascimento e género) de todas as pessoas que participaram num filme (atores, diretores, figurinistas, etc.).
- Um utilizador poderá ter acesso às avaliações de outros utilizadores.
- Um utilizador poderá ter acesso a todos os filmes nomeados a um prémio num dado ano.
- Um utilizador poderá saber o filme vencedor de um dado prémio, num determinado ano.
- Um utilizador poderá ter acesso a todas as pessoas nomeadas a um prémio pela sua participação num dado filme, num determinado ano.
- O sistema poderá determinar que pessoas venceram um dado prémio, num dado ano, pela sua participação num dado filme.
- O Sistema poderá filtrar filmes, de acordo com determinados critérios
- O sistema poderá determinar quantos filmes com um dado ator estrearam depois de um determinado ano.
- O sistema poderá determinar quais os atores que também executaram funções *backstage* num dado filme.
- O sistema poderá determinar quantas mulheres estiveram trabalharam num filme, seja como atrizes, diretoras, roteiristas, etc.
- O sistema poderá determinar o rating de um filme tendo em conta as avaliações dos utilizadores.
- O sistema poderá determinar os filmes com mais prémios vencidos.
- O sistema poderá determinar o *cast* e a *crew* de um filme.

2.2.3 Requisitos de Controlo

- Um utilizador do Sistema não poderá alterar o esquema da base de dados.
- A entidade responsável pelo controlo do sistema poderá adicionar e editar novos filmes, prémios, funções e pessoas, assim como, consultar as suas informações;

2.3. Análise e validação geral dos requisitos

Durante processo de levantamento de todos os requisitos, houve um diálogo e discussão constante entre os elementos do grupo de forma a garantir se estaríamos a ter em conta todos os requisitos principais. Além disto, fomos esclarecendo dúvidas coma equipa docente e pedindo algum feedback de forma a confirmar que estávamos no caminho certo. Tudo isto contribuiu para a validação dos requisitos.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Após o levantamento e a consequente análise dos requisitos, começamos o desenvolvimento do desenho do **Modelo Conceptual**. É nesta fase onde relacionamos todas as informações que obtivemos anteriormente entre si. O objetivo foi conseguir obter a implementação de um modelo de fácil compreensão e que suprimisse as necessidades do utilizador.

Para tal, usamos o Diagrama ER, uma vez que, trata-se de um fluxograma que ilustra como é que as "entidades", como por exemplo pessoas ou conceitos, de um sistema se relacionam entre si.

Este tipo de diagrama é composto por entidades, relacionamentos e atributos, além disso, também é descrita a cardinalidade, de forma a definir em termos numéricos as relações existentes dentro do nosso sistema. A identificação de entidades trata-se de encontrar os principais conceitos, objetos, pessoas e agrupá-los conforme as propriedades de cada um, estas propriedades é aquilo que consideramos como "atributos". Os atributos são responsáveis por caracterizar cada uma das entidades. Por fim, o relacionamento é o que permite criar associações entre as várias entidades existentes no sistema em causa e ainda atribuir-lhes uma cardinalidade de forma a representar restrições dos mesmos.
Sendo assim, a construção deste modelo permite descrever os requisitos de uma forma abstrata para a posterior construção da Base de Dados.

É crucial que durante o processo da conceção do modelo conceptual, todas as relações entre as entidades e a própria identificação das entidades estejam de acordo com aquilo que o utilizador pretende encontrar, de uma forma coerente e composta.

3.2. Identificação e caracterização das entidades

- **Filme:**

  A entidade filme é representada através do seu "**id"**, do "**Título"** e da sua respetiva "**Descrição"**. Além disso, temos outros atributos relacionados que revelam várias informações sobre o filme em questão, tal como a "**Data de Estreia"**, a "**Língua"** do filme, o "**País"** onde foi produzido, a sua "**Duração"** em horas e minutos, o "**Orçamento"** utilizado para a produção em dólares, a "**Receita"** global no total e também o "**PG"** (podendo ser este *parental guidance* ser pg-6, pg-12,pg-14 pg-16 ou pg-18).

- **Género:**

  Esta entidade representa o tipo do filme, podendo este ser de ação, aventura, romance, terror, comédia, suspense, animação, familiar, crime, drama, fantasia, *Sci-Fi*, mistério, biografia, thriller e musical. A designação do género é referida através do atributo "**Nome"** e além disso, esta entidade é também caracterizada pelo seu "**id"**.

- **Prémio:**

  É descrita através de três atributos: "**Categoria"**, "**Nome"**, "**id"**. A Categoria é o tipo de prémio que poderá ser atribuído a um filme ou a uma pessoa, tal como por exemplo "Melhor Filme", "Melhor Filme Estrangeiro", "Melhor Diretor", "Melhor Ator Principal", "Melhor Ator Coadjuvante", "Melhor Atriz Principal", "Melhor Atriz Coadjuvante", "Melhor Roteiro Original", "Melhor Trilha Sonora", "Melhor Figurino", "Melhores Efeitos Especiais", "Honorário" e "Melhor Filme de Animação". A categoria "Honorário" trata-se de um prémio atribuído pela cerimónia de premiação *The Academy Awards*, mais conhecida por *Oscars*, onde é entregue um prémio a uma pessoa de forma a honrar a sua carreira ao longo dos anos. Além disso, como existem também outras cerimónias de premiação bastante conhecidas, tal como por exemplo os "César Awards", os "BAFTA", etc., adicionamos o atributo Nome que se encarrega de dizer qual a cerimónia em questão.

- **Pessoa:**

  A entidade Pessoa, é caracterizada através dos atributos "**Nome"** e "**id"**. E além disso, ainda contém o "**género"** da pessoa, Feminino(F) ou. Masculino(M) e a respetiva **Data de Nascimento**.

- **Função:**

  Esta entidade trata-se da função desempenhada por uma pessoa que seja membro da *Crew* de um filme, seja Diretor/Produtor, Figurinista, Compositor, Técnico de Efeitos Especiais, ou Roteirista. Além do atributo "**Designação"** da função temos também o "**id"**.

- **User:**

  A entidade User representa todos os utilizadores da aplicação, e temos acesso a informações como o nome do User ("**Username"**), o "**email"**, e a "**password"**. É importante referir que cada username é único (daí a ausência do atributo id). Além disso, temos também o atributo "**Nível"**, que identifica quais os utilizadores que mais fazem reviews e atribuem ratings aos filmes, podendo serem caracterizados como *newbie* (praticamente não utilizou a aplicação), *active* (já escreveu algumas *reviews*) ou *big fan* (já fez uma quantidade significativa de *reviews*).

- **Review:**

  Esta entidade é representada através do seu "**id"**. Além disso, contém outros atributos tais como, a "**data"**, de modo a saber quando é que a *review* em causa foi publicada, o **comentário** que o utilizador fez (ou seja, a *review* em si) e ainda o **rating** dado ao filme de 0 a 10.

3.3. Identificação e caracterização dos relacionamentos

- **Relacionamento Filme - Género**

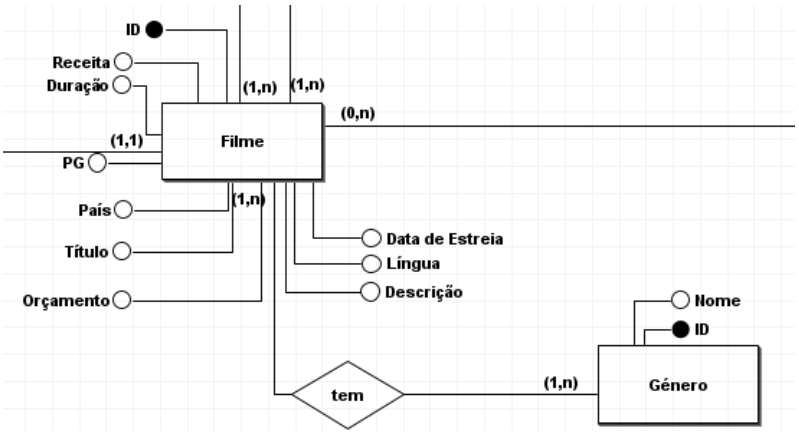


Figura 1 – Relacionamento Filme - Género

Relacionamento: Filme tem Género.

- Através da nossa pesquisa vimos que um filme poderia ter um ou mais géneros, além de que o mesmo género pode estar presente em vários filmes, portanto houve a necessidade da criação de uma entidade Género para que tal fosse possível representar na nossa futura Base de Dados. Além disso, em qualquer aplicação como o IMDb ou plataforma de Streaming podemos reparar que os filmes estão organizados por secções correspondentes a um género, podendo o mesmo filme aparecer em diferentes géneros.

Cardinalidade: Filme (1, n) – Género (1, n).

Atributos: Não possui atributos.

- **Relacionamento User - Review**

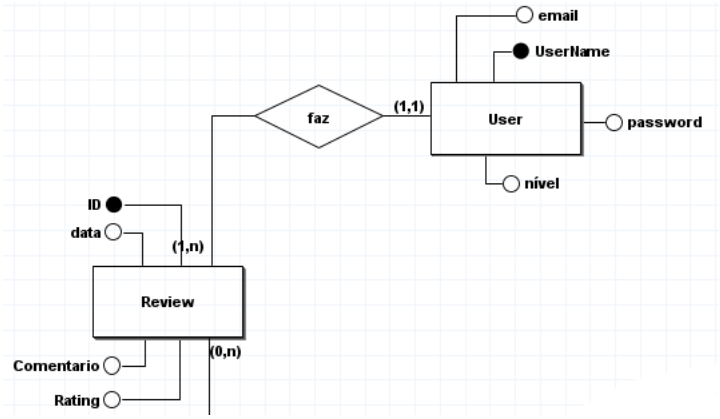


Figura 2 - Relacionamento User-Review

Relacionamento: User faz Review.

- Este relacionamento caracteriza a associação entre o User e a Review, sendo que, consideramos que um User faz apenas uma review para cada filme à sua escolha. Caso ele queira alterar a sua opinião em relação a um filme poderá editar a review que fez anteriormente. Além disso, esta associação é importante de modo a saber quem é que fez uma dada review a um filme.

Cardinalidade: User (1, 1) – Review (1, n).

Atributos: Não possui atributos.

- **Relacionamento Review - Filme**

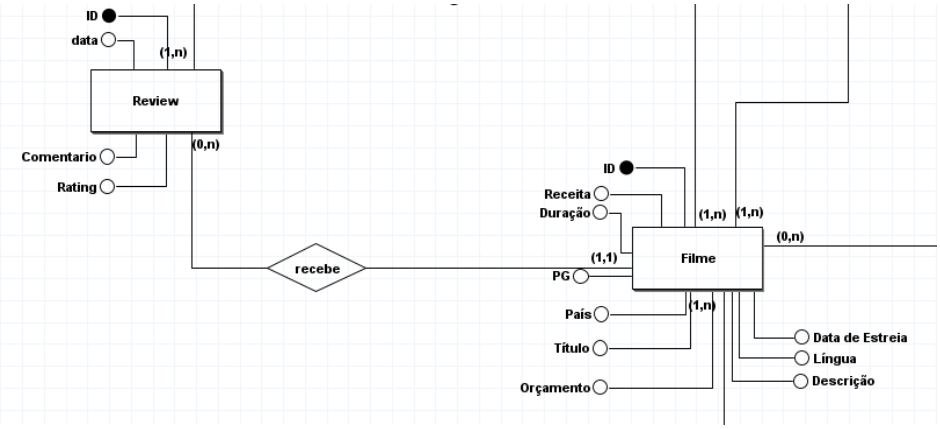


Figura 3 - Relacionamento Review-Filme

Relacionamento: Filme recebe Review.

- Qualquer utilizador poderá comentar e dar a sua opinião sobre vários filmes. Para tal é necessário saber quais as reviews dirigidas a um dado filme, daí o surgimento desta relação. Um filme poderá receber várias reviews de diversos utilizadores, como também poderá não ter nenhuma review. Por exemplo, caso o filme seja recente, é normal que ainda não tenha recebido comentários por parte dos utilizadores da aplicação.

Cardinalidade: Filme (1, 1) – Review (0, n)
Atributos: Não possui atributos.

- **Relacionamento Pessoa - Filme**

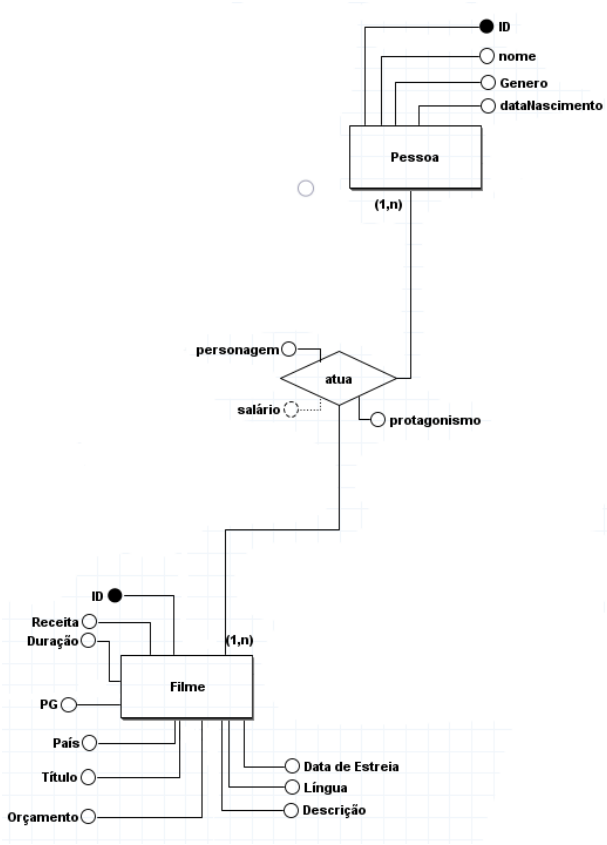


Figura 4 - Relacionamento Pessoa-Filme

Relacionamento: Pessoa atua Filme

- Esta associação está destinada às pessoas que participam num dado filme como atores. Esta relação permite saber qual a personagem desempenhada num filme em causa e o seu protagonismo. Sendo que a mesma pessoa pode participar em diversos filmes. Além disso, para alguns casos dá-nos a conhecer o salário do ator no filme. Deste modo a pessoa além dos atributos relacionados com a sua entidade ainda terá os atributos resultantes deste relacionamento.

Cardinalidade: Pessoa (1, n) – Filme (1, n)

Atributos: Personagem (Nome da personagem desempenhada); Protagonismo (Personagem Principal ou Secundária); Salário do ator (em dólares).

- **Relacionamento Pessoa – Filme – Prémio**

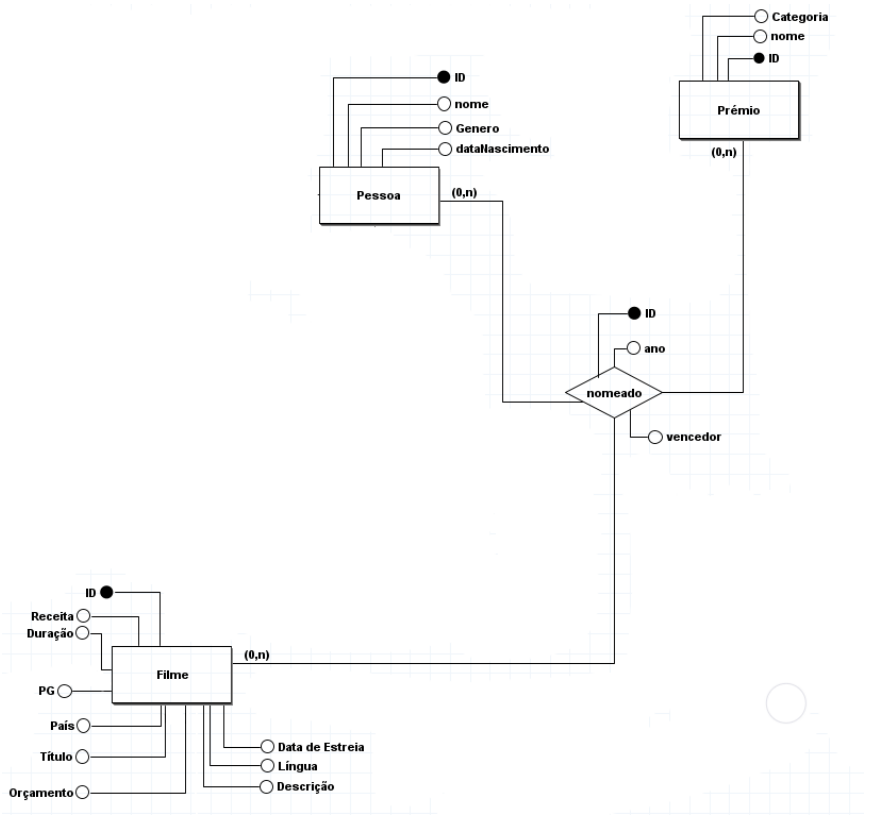


Figura 5 - Relacionamento Pessoa-Prémio-Filme

Relacionamento: Uma pessoa é nomeada a um prémio através de um dado filme. / Pessoa nomeada a um prémio. / Filme nomeado a um prémio.

- Este relacionamento ternário é o que nos permite associar entre si as entidades Pessoa, Filme e Prémio, uma vez que, uma pessoa que é nomeada para um dado prémio é sempre através do filme para o qual trabalhou, atuou ou produziu, exceto no caso do prémio "honorário", uma vez que este é atribuído a alguém devido à sua excelência ao longo da sua carreira ou pelo seu contributo para a indústria cinematográfica. Daí considerarmos Filme (0, n), pois o filme poderá não estar relacionado. Além disso, no caso de um filme ganhar o prémio de "Melhor Filme" também consideramos que não devemos relacionar uma Pessoa com esse Prémio, portanto optamos por Pessoa (0, n).

Cardinalidade: Pessoa (0, n) – Filme (0, n) – Prémio (0, n)

Atributos: Temos como atributos deste relacionamento: "Ano" e "Vencedor". O "ano" trata-se do ano em que ocorreu a nomeação e o "vencedor" identifica se uma dada pessoa ou filme ganhou ou não o prémio para o qual foi nomeado. Além disso, temos também um atributo identificador, o **id"**, uma vez que, como as entidades deste relacionamento poderão todas ser null e repetidas, nenhuma delas poderá ser uma chave primária na nossa futura *join table*, portanto surgiu a necessidade de considerar o atributo id.

- **Relacionamento Pessoa – Função – Filme**

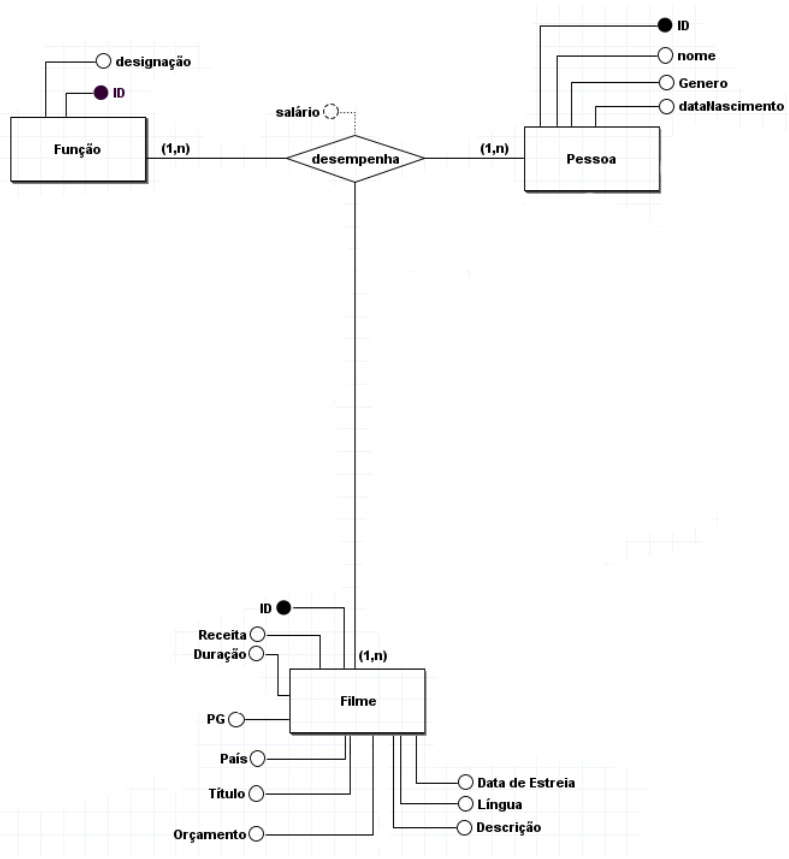


Figura 6 - Relacionamento Pessoa-Função-Filme

Relacionamento: Pessoa desempenha uma dada Função num Filme.

- Esta associação entre estas 3 entidades representadas na Figura 7 trata-se de um relacionamento ternário. Este relacionamento surgiu como uma alternativa ao uso das possíveis classes Ator / Diretor etc., devido ao elevado número de funções que é possível desempenhar na produção de um filme. Como vimos anteriormente (Figura 4) uma pessoa pode atuar, mas também pode desempenhar uma função de backstage, isto é, pode fazer parte da crew do filme. Além disso, consideramos ainda que uma pessoa recebe um certo salário de acordo com a função que desempenha num dado filme.

Cardinalidade: Pessoa (1, n) – Função (1, n) – Filme (1, n)
Atributos: Salário (quanto uma pessoa recebe em dólares num certo filme dada a sua função).

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos.

Entidade	Atributo	Tipo de dados	Identificador	Opcional	Composto	Multivalorado
Filme	id	INT	Sim.	Não.	Não.	Não.
	Nome	VARCHAR (45)	Não.	Não.	Não.	Não.
	Custo / Orçamento	FLOAT	Não.	Não.	Não.	Não.
	País	VARCHAR (20)	Não.	Não.	Não.	Não.
	Língua	VARCHAR (20)	Não.	Não.	Não.	Não.
	Duração	TIME	Não.	Não.	Não.	Não.
	Receita	FLOAT	Não.	Não.	Não.	Não.
	PG	INT	Não.	Não.	Não.	Não.
	Descrição	TINYTEXT	Não.	Não.	Não.	Não.
	DataEstreia	DATE	Não.	Não.	Não.	Não.
User	Username	VARCHAR (45)	Sim.	Não.	Não.	Não.
	Email	VARCHAR (45)	Não.	Não.	Não.	Não.
	Nível	VARCHAR (10)	Não.	Não.	Não.	Não.
	PalavraChave	VARCHAR (45)	Não.	Não.	Não.	Não.
Função	IDFunção	INT	Sim.	Não.	Não.	Não.
	Designação	VARCHAR (30)	Não.	Não.	Não.	Não.
Review	IDReview	INT	Sim.	Não.	Não.	Não.
	DataReview	DATETIME	Não.	Não.	Não.	Não.
	Comentario	TINYTEXT	Não.	Não.	Não.	Não.
	Rating	INT	Não.	Não.	Não.	Não.
Género	IDGenero	INT	Sim.	Não.	Não.	Não.
	Nome	VARCHAR (15)	Não.	Não.	Não.	Não.
Prémio	IDPrémio	INT	Sim.	Não.	Não.	Não.
	Categoria	VARCHAR (45)	Não.	Não.	Não.	Não.
	Nome	VARCHAR (45)	Não.	Não.	Não.	Não.
Pessoa	IDPessoa	INT	Sim.	Não.	Não.	Não.
	Nome	VARCHAR (45)	Não.	Não.	Não.	Não.
	DataDeNascimento	DATE	Não.	Sim.	Não.	Não.
	Género	VARCHAR (10)	Não.	Não.	Não.	Não.

Tabela 1 - Caracterização dos Atributos das Entidades

- **Atributos – Filme:**

O “**ID”** é o atributo identificador da entidade Filme. Todos os restantes atributos não são opcionais, pois são informações necessárias para guardar sobre o filme na futura base de dados e além disso, é informação que é conhecida previamente.

- **Atributos – User:**

O **“Username”** é o atributo identificador, uma vez que, é único e não poderá repetir-se na nossa base de dados, de forma a identificar e distinguir corretamente cada utilizador. O resto dos atributos não são opcionais pois consistem em dados que são obrigatoriamente necessários guardar sobre um dado User. Além disso, não são também multivalorados nem compostos.

- **Atributos – Função:**

O atributo identificador é o “**FunçãoID”**. A “**Designação”** trata-se do atributo que irá ter a designação da função em causa, logo não poderá ser opcional.

- **Atributos – Review:**

O **“IDReview”** é o atributo identificador da entidade Review. Nenhum dos atributos existentes é opcional, uma vez que se trata de informação obrigatória que deverá fazer parte de cada review, tal como o comentário que em si é feito ao filme, o rating e ainda a data e hora associada a essa mesmo review.

- **Atributos – Género:**

O “**IDGénero”** trata-se do atributo identificador, e “**Nome”** é o atributo que designa qual o género em causa, desta forma, não poderá ser um atributo opcional.

- **Atributos – Prémio:**

O atributo identificador é o “**IDPrémio”**. Os restantes são não opcionais uma vez que representam a cerimónia de premiação que irá entregar o prémio (nome) e a categoria do mesmo.

- **Atributos – Pessoa:**

O “**IDPessoa”** é o atributo identificador desta entidade. O atributo “**DataDeNascimento”** é opcional, uma vez que nem sempre é possível encontrar a data de nascimento de certas pessoas que trabalham para o filme devido à privacidade dos dados. Os restantes atributos são não opcionais, pois trata-se de informação que é imprescindível ter sobre uma dada pessoa para fazer parte da nossa futura base de dados.

Relacionamento			Atributo	Tipo de Dados	Identificador	Opcional	Composto	Multivalorado
Pessoa	**Filme**		Protagonismo	VARCHAR (45)	Não.	Não.	Não.	Não.
			Personagem	VARCHAR (45)	Não.	Não.	Não.	Não.
			Salário	FLOAT	Não.	Sim.	Não.	Não.
Pessoa	**Função**	**Filme**	Salário	FLOAT	Não.	Sim.	Não.	Não.
Pessoa	**Prémio**	**Filme**	Ano	INT	Não.	Não.	Não.	Não.
			Vencedor	VARCHAR (5)	Não.	Não.	Não.	Não.
			TableID	INT	Sim.	Não.	Não.	Não.

Tabela 2 - Caracterização dos Atributos dos Relacionamentos

- **Atributos – PessoaFilme**

O relacionamento "Pessoa atua num filme" trata-se de associar um dado ator com o filme em que ele desempenha um dado papel. Para tal é necessário saber qual a personagem que interpretou, daí o atributo "**Personagem"** não ser opcional, e ainda o "**Protagonismo"** dessa mesma personagem, sendo também não opcional. Além disso, temos ainda o atributo Salário, tratando-se de um valor aproximado do dinheiro que um ator ganhou com um dado filme. Como nem sempre esta informação é revelada trata-se de um atributo opcional.

- **Atributos – PessoaFunçãoFilme**

Este relacionamento associa as pessoas que desempenham uma função no backstage de um filme. Contém apenas o atributo "**Salário"** uma vez que este varia conforme o filme, uma pessoa tanto poderá ganhar mais ou menos dependendo para qual o filme trabalhou e o desempenho do mesmo na Receita Global. Este atributo é opcional uma vez que se trata de uma informação privada que nem toda a gente poderá querer revelar.

- **Atributos – PessoaPrémioFilme**

Este relacionamento tem o "TableID" como atributo identificador, uma vez que na futura *join table* não poderemos usar "PessoaID", "PrémioID" ou "FilmeID" como chaves primárias pois poderão tomar valor null, tal como vimos anteriormente. Além disso, temos também os atributos Ano e Vencedor que são não opcionais.

3.5. Detalhe ou generalização de entidades

No início da conceção deste modelo foi ponderado representar todas as pessoas integrantes de um filme, tal como Diretores, Atores, Figurinistas, Técnicos de Efeitos Especiais, etc. através de uma entidade que generalizasse todas estas hipóteses, uma vez que têm informações/atributos em comum.
Isto é, a nossa entidade Pessoa poderia ser a nossa "entidade-pai", e por exemplo poderíamos ter criado "entidades-filhos" tal como "Ator", "Diretor" ou "Produtor", de forma a poder representar uma hierarquia que mostrasse as dependências entre entidades de uma mesma categoria.
Como o grupo queria apresentar uma solução que incluísse as mais diversas funções que existem no backstage de um filme, de forma a que a futura base de dados não ficasse limitada e pudesse conter e ser capaz de adicionar pessoas cujo trabalho não é ser ator, diretor ou então produtor, optou-se por não desenvolver a solução que incluía a generalização/detalhe de entidades.
Para tal, implementamos na mesma a classe Pessoa, que contém todos os atributos/dados em comum entre todos os indivíduos que trabalham num filme, e ainda, uma classe Função. Esta classe Função irá conter todo o tipo de funções existentes e presentes no backstage de um filme, fazendo desta forma uma diferenciação entre membros da *Crew* e do *Cast*. Os membros do cast, isto é, os atores, vão estar representados através de um relacionamento com a classe Filme, cuja designação é "atua", tal como vimos anteriormente. Desta forma, uma Pessoa poderá desempenhar uma Função e/ou atuar num Filme.

3.6. Apresentação e explicação do diagrama ER

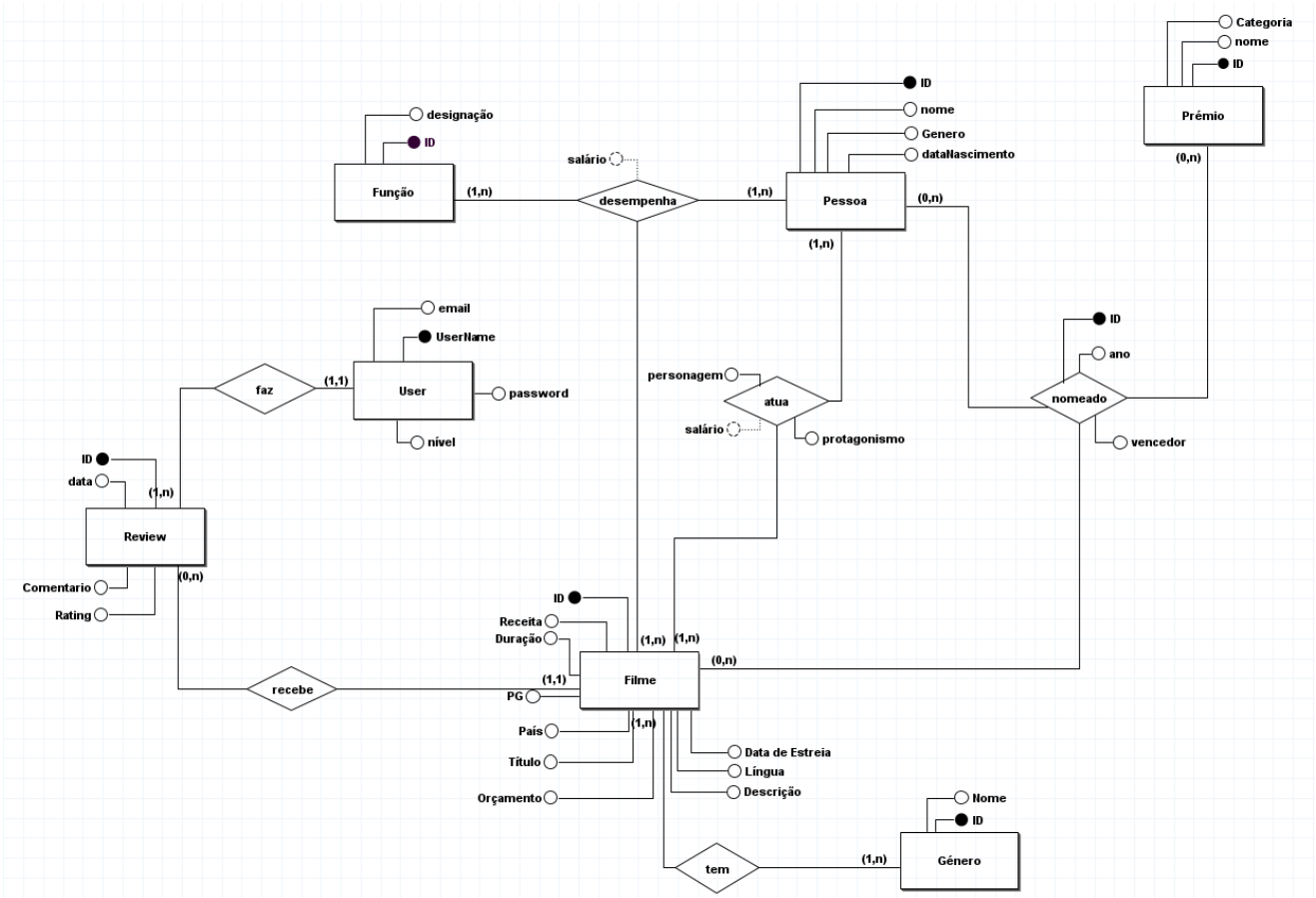


Figura 7 - Diagrama ER

Tal como vimos anteriormente, o nosso modelo conceptual baseia-se na estrutura de um típico website destinado à disponibilização de informações e dados relativos a uma grande variedade de filmes. Para tal, começamos por identificar as entidades mais importantes, tal como os respetivos atributos, e depois por criar relacionamentos entre elas, de modo a criar associações que consigam representar a futura base de dados e elucidar hipoteticamente um cliente sobre a resolução optada.

Agora que a vista do nosso modelo em geral está apresentada, iremos passar a explicar o raciocínio por detrás dos relacionamentos e entidades. Em primeiro lugar, temos a entidade **Filme** ligada a um ou mais **Género**s, que poderá receber várias **Review**s feitas pelos utilizadores (**User**). Além disso, uma **Pessoa** poderá desempenhar uma **Função** própria do *backstage* num filme e/ou atuar num filme, interpretando uma personagem. Tanto uma Pessoa como um Filme poderão ser nomeados para vários **Prémio**s, dependendo ou não um do outro. Isto é, uma pessoa poderá ganhar um prémio sem estar relacionada com um filme (pessoa, null, prémio), tal como é o caso dos prémios que honram a carreira ou o contributo de uma pessoa na indústria cinematográfica. E, um filme poderá ganhar a categoria de "Melhor Filme", onde consideramos que não há associação entre pessoa e filme (null, filme, prémio).

Para distinguir os nomeados dos vencedores, existe o atributo “vencedor” na relação **FilmePessoaPrémio**, que poderá tomar o valor de “Sim” caso se trate de um vencedor e “Não” caso contrário.

3.7. Validação do modelo de dados produzido

Para validar o modelo, realizamos uma discussão entre os membros do grupo para perceber se todos os requisitos foram cumpridos, e se o nosso modelo conseguiria descrever e incluir outros casos que fossem considerados “fora do habitual”.

Concluímos, após uma posterior análise dos requisitos e do nosso modelo, que conseguimos com sucesso criar um modelo que representasse de forma adequada o nosso sistema. Isto é, de acordo com o que foi estipulado conseguimos incluir todas as informações que um filme deveria ter, os dados relacionados com um User para este inscrever-se ou fazer login, para depois poder consultar filmes e fazer reviews dos mesmos, sendo que um User apenas faz uma review para cada filme, se quiser fazer uma nova review sobre um mesmo filme, assumimos que ele terá a possibilidade de editar a anterior. As informações de cada pessoa, o papel ou a função que poderá desempenhar num filme também foram bem conseguidas. Além disso, ainda tivemos a preocupação de ter em conta todos os casos em que uma pessoa poderia ter uma função de backstage e ainda atuar num dado filme, tal como por exemplo casos em que um ator seja também diretor no mesmo filme. Em relação aos prémios, tivemos o cuidado de conseguir diferenciar os filmes que foram nomeados e conseguiram vencer o prémio, daqueles que apenas foram nomeados. E, ainda, incluir os prémios que não associam uma pessoa a um filme, tal como, o prémio honorário ou então o prémio de melhor filme, os restantes consideramos que associam sempre uma pessoa a um filme uma vez que a pessoa ganha um prémio devido ao seu desempenho nele.

Resumindo, o nosso modelo consegue responder aos requisitos impostos e ainda verificar casos não tão usuais.

Modelação Lógica

3.8. Construção e validação do modelo de dados lógico

Para criar o modelo lógico utilizámos o MySQL Workbench.

Inicialmente construímos todas as tabelas correspondentes ás entidades do nosso modelo conceptual, levantámos as chaves primárias e atribuímos tipos aos atributos. Deste processo, realçamos alguns pontos:

- Na tabela "*User*" escolhemos como chave primária o "*username*", mas poderia ser o seu email. Foi apenas uma opção do grupo.
- Em todas as tabelas referentes a entidades, á exceção da tabela *User*, foi criado um atributo id cujo único objetivo é ser chave primária.
- Na tabela "*Review*" considerámos o tipo do atributo "comentário" TINYTEXT (255 bytes). Esta escolha deveu-se ao facto de que o grupo achou que seria da preferência dos utilizadores que as avaliações fossem ser curtas e sucintas.

Por último, foram criadas as tabelas referentes às relações entre as entidades.

- **Relacionamentos binários N:M**

  - **Relação entre filme e género**

    Visto que um filme possui vários géneros e o mesmo género pode estar presente em vários filmes, nota-se, aqui uma relação de N:M. Desta forma, foi necessário criar uma tabela para representar a relação entre estas duas entidades, na qual os atributos são as chaves primárias de filme e género (a chave primária da tabela relação é composta).

  - **Relação entre Filme e Pessoa**

    A relação entre filme e pessoa é exclusivo para atores, e visto que um ator atua em vários filmes e um filme tem vários atores (relacionamento N:M) foi necessária uma tabela para representar esta relação. Esta tabela correspondente á relação terá como atributos as chaves primárias de filme e

pessoa, a personagem que o ator interpretou, o seu protagonismo e, o seu salário se for conhecido (salário é um atributo opcional).

- **Relacionamentos ternários**

  Estes relacionamentos exigem quadro tabelas: as tabelas das três entidades envolvidas e a tabela do relacionamento.

  - **Relação entre Filme, Função e Pessoa**

    Num filme, é importante saber quem são os diretores, os roteiristas, os responsáveis pela banda sonora etc. Desta forma, foi necessário um relacionamento ternário de forma a saber que pessoa, executou determinada função num determinado filme. A tabela do relacionamento, terá como atributo as chaves primárias das tabelas "Pessoa", "Função" e "Filme", além de ter também o salário dessa pessoa, se for conhecido (atributo salário é opcional).

  - **Relação entre Filme, Prémio e Pessoa**

    Esta foi a tabela que mais se modificou ao longo do projeto e a que levantou mais dúvidas ao grupo.

    Inicialmente existia uma relação binária entre filme e pessoa, e uma ternária entre filme, pessoa e prémio. A primeira, servia para determinar os prémios a que os filmes foram nomeados e a segunda para saber a que prémios, determinada pessoa foi nomeada pela sua participação num dado filme. No entanto, decidimos retirar a relação binária entre Filme e Prémio e considerar o atributo correspondente á chave primária de Pessoa opcional (logo não fará parte da chave primária da tabela relação). Assim, cada vez que quisermos saber a quantos prémios de melhor filme, por exemplo, foi nomeado um filme, basta considerar que nesses casos o atributo "idPessoa" será null. Por outro lado, se quisermos saber quais as nomeações que uma pessoa teve pela sua participação num filme, teremos de considerar que nem o atributo "idPessoa", nem o atributo "idFilme" podem ser nulos.

    Posteriormente, decidimos incluir na nossa BD, prémios honorários, ou, seja prémios que uma pessoa recebe pela sua carreira, e não em relação a um dado filme. Para isto ser possível, tivemos de considerar que nestes casos o atributo "idFilme" será null.

    Visto que "idFilme" e "idPessoa" podem ser nulos, obviamente não farão parte da chave primária da tabela relação. Sobra apenas o atributo que é chave primária da tabela Prémio. Mas este não poderia ser, sozinho, chave primária uma vez que não conseguiríamos garantir a sua unicidade. Por exemplo, se quiséssemos considerar várias nomeações a um mesmo prémio, já não funcionaria. Desta forma, a solução que o grupo encontrou foi acrescentar um atributo "idTable" para servir de chave primária. Esta decisão foi contestada, no início, por alguns elementos do grupo, por acharem estranho a adição de um id numa tabela relação. Mas visto

que o relacionamento representava nomeações, e era impossível arranjar uma chave composta com os atributos que tínhamos, o grupo acabou por consentir esta abordagem.

Além disto, esta tabela do relacionamento ternário tem também, um atributo “ano” em que ano foi feita a nomeação, e um atributo “vencedor”, que informa se aquela nomeação foi vendida ou não.

3.9. Desenho do modelo lógico

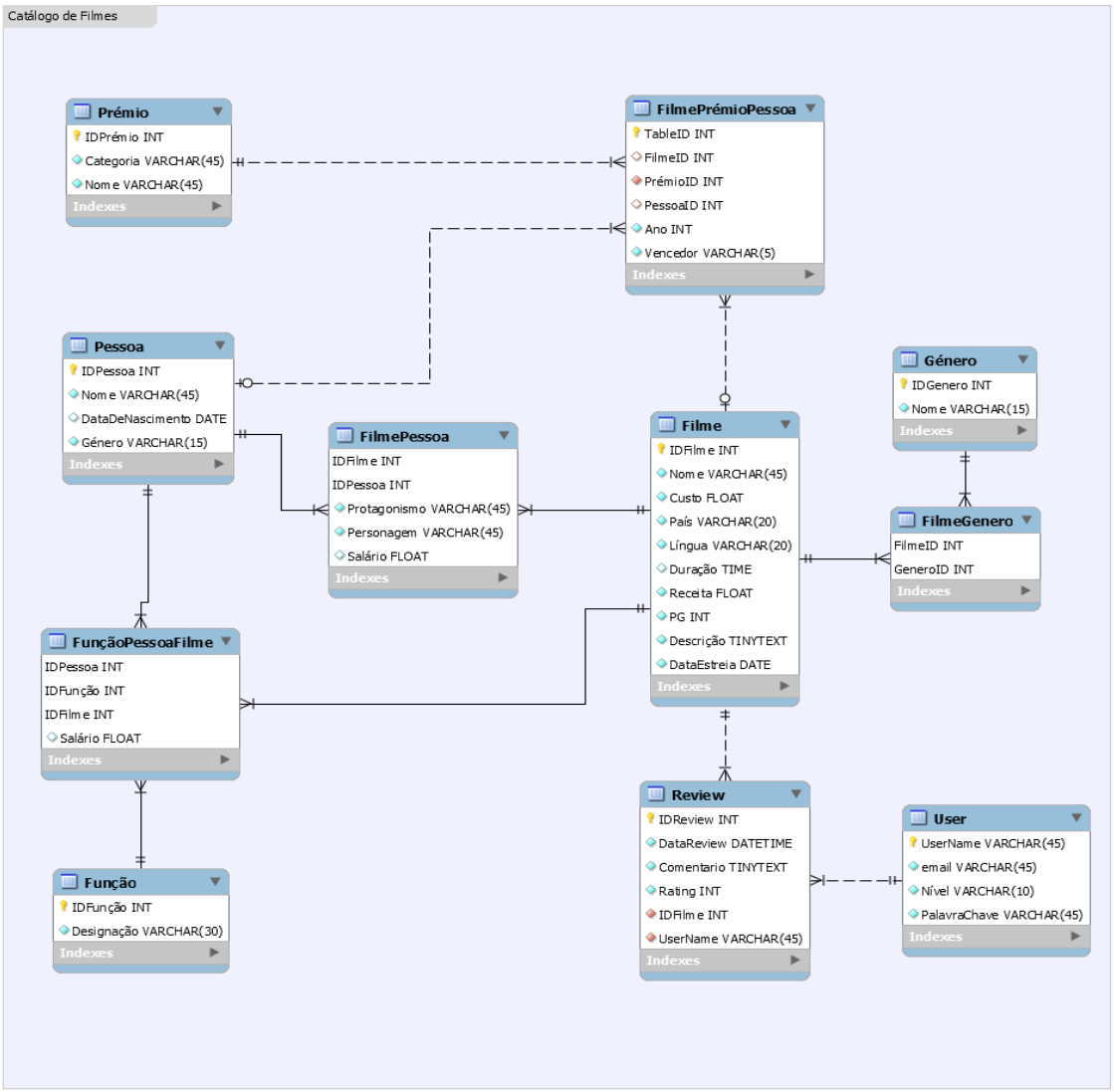


Figura 8 - Modelo Lógico

3.10. Validação do modelo através da normalização

- **Primeira Forma Normal**

A normalização de uma tabela na 1.ª Forma Normal (1FN) exige que a tabela tenha uma estrutura bidimensional correta, ou seja, cada linha deve corresponder a um só registo e cada coluna a um só campo.

No caso do nosso modelo, podemos dizer que está na primeira forma normal, porque cada entrada de uma tabela é atómica, ou seja, neste caso a cada atributo corresponde apenas um só valor. Isto é visível, por exemplo, no caso das *Reviews*. Um utilizador faz várias avaliações de filmes, mas estas encontram-se numa tabela á parte.

- **Segunda Forma Normal**

A 2.ª Forma Normal diz que a tabela tem de estar na 1ª FN e que cada atributo não chave tem de ser funcionalmente dependente da totalidade da chave primária e não apenas de uma parte dessa chave.

No nosso modelo, nenhum atributo em nenhuma tabela com chave primária composta depende apenas de uma parte de *primary key* e podemos ver isso nas seguintes tabelas (as únicas com chaves primárias):

- *FilmePessoa*

  A chave primária é formada pelo "IDPessoa" e "IDFilme".

  O salário, a sua personagem e respetivo protagonismo, depende do ator que a interpreta e do filme em questão.

- *FunçãoPessoaFilme*

  A chave primária é formada pelo "IDPessoa" , "IDFilme" e "IDFunção".

  O salário de uma pessoa que é membro da *crew* do filme, depende da pessoa, da função que desempenha e do filme em questão.

- **Terceira Forma Normal**

A 3.ª Forma Normal (3FN) diz que a tabela tem de estar na 2ª FN e que nenhum atributo não chave pode depender funcionalmente de algum outro atributo que não seja a chave primária.

Neste caso, como podemos verificar, não existem atributos que dependem entre si dentro de uma tabela. Por exemplo, inicialmente foram incluídos ambos os atributos "Idade" e "DataDeNascimento" na

tabela *Pessoa*. Como a idade de uma pessoa depende da sua data de nascimento, foi retirado o atributo "Idade", assegurando assim que a tabela *Pessoa* está na terceira forma normal.

3.11. Validação do modelo com interrogações do utilizador

De forma a verificar se o nosso modelo seria válido, levantámos algumas interrogações que terão de ser corretamente respondidas.

- **Interrogação 1 - Filmes mais premiados, organizados de forma decrescente**

$$Filme \bowtie (Filme.IDFilme = FilmePrémioPessoa.FilmeID)\ FilmePrémioPessoa$$

$$\sigma_{(FilmePrémioPessoa.Vencedor = \text{'}Sim\text{'})}$$

$$\gamma\ (Nome)$$

$$\pi\ (\rho_{Título,(Nome)}, \rho_{Prémios,(count(*))})$$

$$\pi\ (Título, Prémios)$$

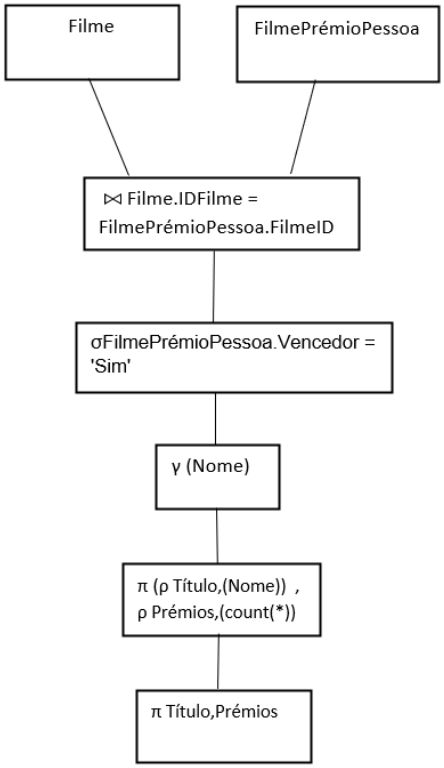


Figura 9 - Árvore Representativa da Interrogação 1

- **Interrogação 2 - Rating de um filme tendo por base as avaliações de *users***

$$Filme \bowtie (Review.IdFilme = Filme.IdFilme) Review$$

$$\sigma_{Filme.Nome = 'Aquaman'}$$

$$\pi(Filme.Nome,\ \rho_{RatingMedia,(AVG(Rating))})$$

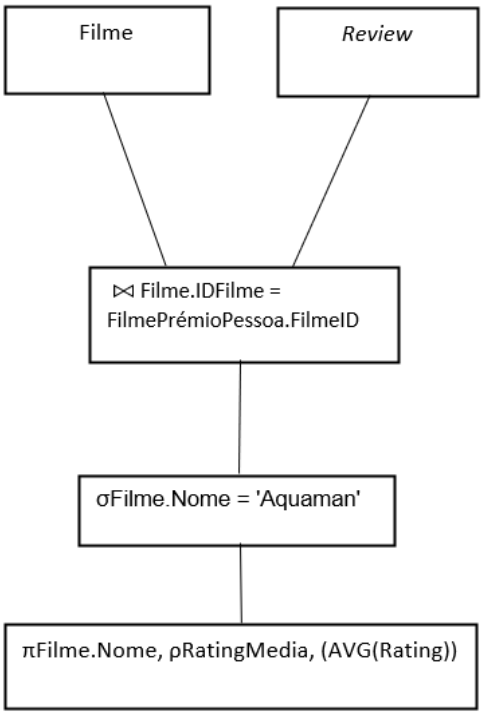


Figura 10 - Árvore Representativa da Interrogação 2

- **Interrogação 3 - Atores que também desempenham funções de *backstage***

$$Pessoa \bowtie (Pessoa.IDPessoa = FunçãoPessoaFilme.IDPessoa)\ FunçãoPessoaFilme$$
$$\bowtie (FunçãoPessoaFilme.IdPessoa = FilmePessoa.IDPessoa$$
$$\wedge\ FunçãoPessoaFilme.IdFilme = FilmePessoa.IDFilme)\ FilmePessoa$$
$$\pi(Nome)$$

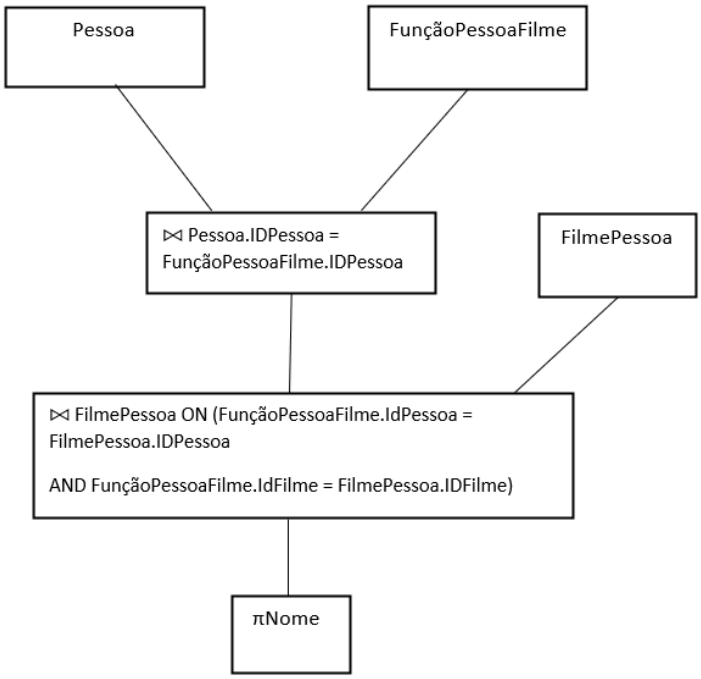


Figura 11 - Árvore Representativa da Interrogação 3

- **Interrogação 4 - Filmes com um dado ator que estrearam depois de um determinado ano**

$$Filme \bowtie (Filme.IDFilme = FilmePessoa.IDFilme)\ FilmePessoa$$
$$\bowtie (FilmePessoa.IdPessoa = Pessoa.IdPessoa)\ Pessoa$$
$$\sigma_{(Pessoa.Nome='Leonardo\ Dicaprio' \wedge YEAR(DataEstreia)>2005)}$$
$$\pi\ (\rho_{TituloFilme,(Filme.Nome)})$$

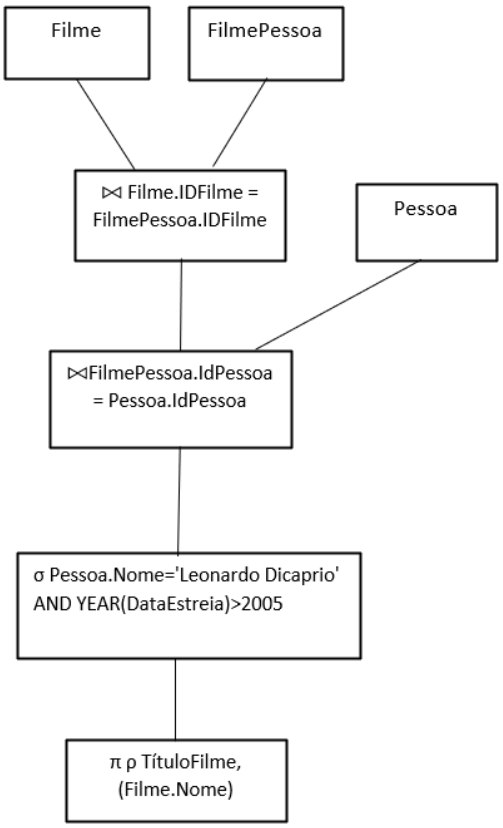


Figura 12 - Árvore Representativa da Interrupção 4

- **Interrogação 5 - *Cast* e *Crew* de um filme**

$$FilmePessoa \bowtie (FilmePessoa.IdPessoa = Pessoa.IDPessoa)\ Pessoa$$
$$\bowtie (FilmePessoa.IDFilme = Filme.IDFilme)\ Filme$$
$$\sigma_{Filme.Nome = 'Aquaman'}$$
$$\pi\ (Pessoa.Nome)$$
$$\cup$$
$$FunçãoPessoaFilme \bowtie (FunçãoPessoaFilme.IdPessoa = Pessoa.IDPessoa)\ Pessoa$$
$$\bowtie (FunçãoPessoaFilme.IDFilme = Filme.IDFilme)\ Filme$$
$$\sigma_{Filme.Nome = 'Aquaman'}$$
$$\pi\ (Pessoa.Nome)$$

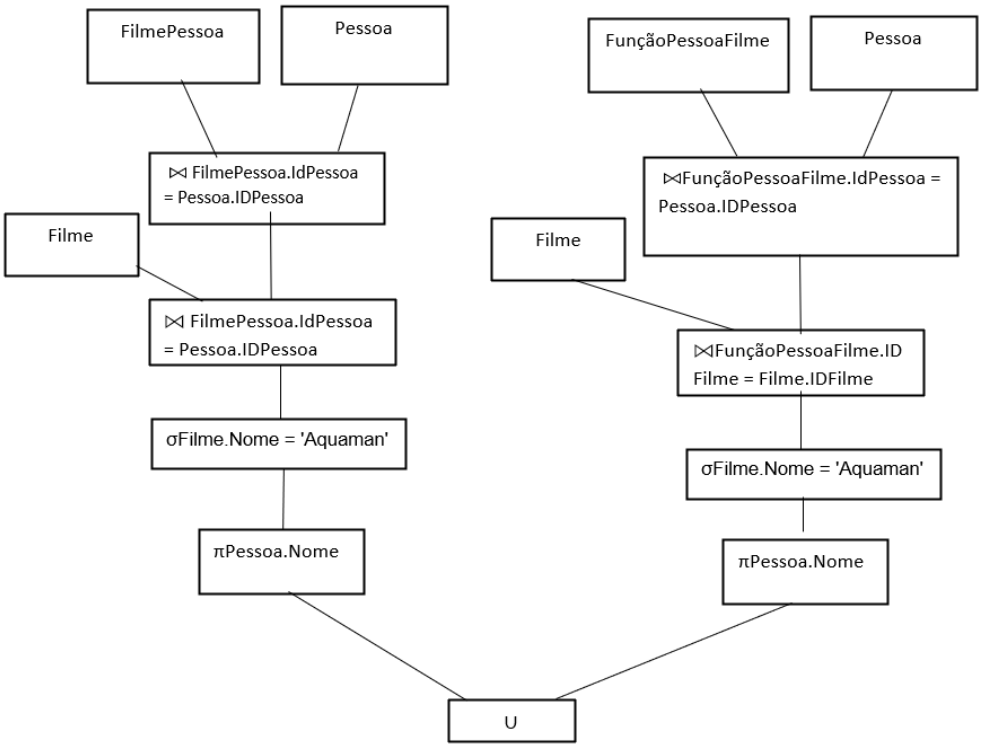


Figura 13 - Árvore Representativa da Interrupção 5

- **Interrogação 6 – Mulheres que participaram num filme, seja como atrizes ou funções *backstage***

$$Filme \bowtie (Filme.IdFilme = FilmePessoa.IdFilme)\ FilmePessoa$$
$$\bowtie (FilmePessoa.IdPessoa = Pessoa.IDPessoa)\ Pessoa$$
$$\sigma_{(Filme.Nome = \text{'Titanic'} \wedge Pessoa.Género = \text{'F'})}$$
$$\pi\ (Pessoa.Nome)$$
$$\cup$$
$$Filme \bowtie (Filme.IdFilme = FunçãoPessoaFilme.IdFilme)\ FunçãoPessoaFilme$$
$$\bowtie (FunçãoPessoaFilme.IdPessoa = Pessoa.IDPessoa)\ Pessoa$$
$$\sigma_{(Filme.Nome = \text{'Titanic'} \wedge Pessoa.Género = \text{'F'})}$$
$$\pi\ (Pessoa.Nome)$$

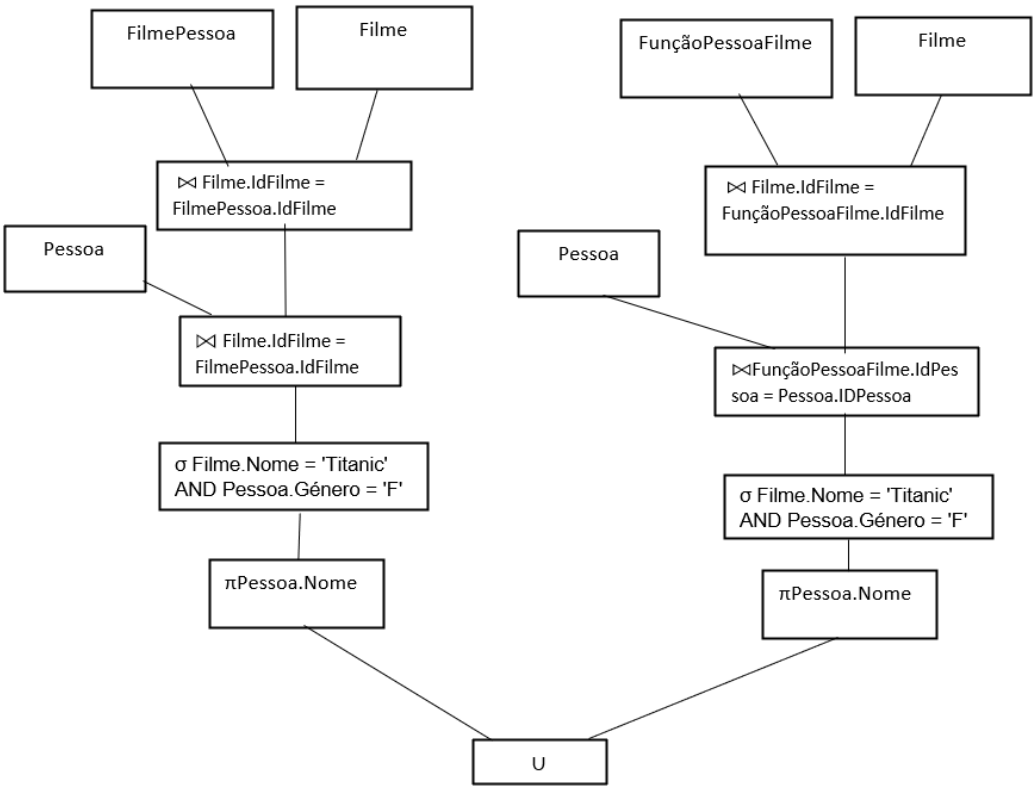


Figura 14 - Árvore Representativa da Interrupção 6

3.12. Revisão do modelo lógico produzido

Após a realização do modelo lógico houve o cuidado de garantir que todas tabelas, os seus atributos e as relações entre elas estivessem corretas. Assim, no final da sua concretização, foi solicitado o *feedback* dos docentes da UC, de forma a garantir que estávamos num bom caminho.

4. Implementação Física

4.1. Seleção do sistema de gestão de bases de dados

O sistema de gestão de base de dados escolhido pelo grupo foi o *MySQL*. A escolha deste deveu-se às suas inúmeras vantagens. Por um lado, é gratuito e está disponível para vários sistemas operativos o que eu foi uma mais valia dado que alguns elementos do grupo usam Linux e outros usam Windows. Por outro lado, o *MySQL WorkBench* possui uma ferramenta de criação do modelo lógico capaz de gerar o script correspondente, o que poupa bastante tempo.
Além disto tudo, foi o sistema aconselhado pela equipa docente.

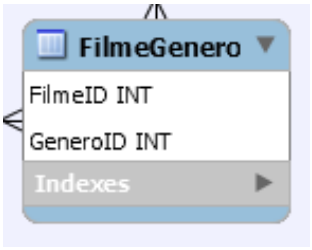
4.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

Visto que o processo de construção do modelo lógico foi realizado com recurso à ferramenta de desenho, administração e desenvolvimento de bases de dados, *MySQL Workbench*, foi utilizado o recurso *Forward Engineering* que esta disponibiliza. Assim, o programa vai gerar automaticamente um algoritmo de criação de todas as tabelas da nossa base de dados.

O código gerado correspondente a cada tabela do modelo lógico foi o seguinte:

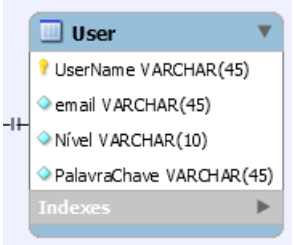

```
CREATE TABLE IF NOT EXISTS `filmeDB`.`Filme` (
  `IDFilme` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Custo` FLOAT NOT NULL,
  `País` VARCHAR(20) NOT NULL,
  `Língua` VARCHAR(20) NOT NULL,
  `Duração` TIME NULL,
  `Receita` FLOAT NOT NULL,
  `PG` INT NOT NULL,
  `Descrição` TINYTEXT NOT NULL,
  `DataEstreia` DATE NOT NULL,
  PRIMARY KEY (`IDFilme`))
ENGINE = InnoDB;
```

Figura 15 - Conversão da tabela "Filme"



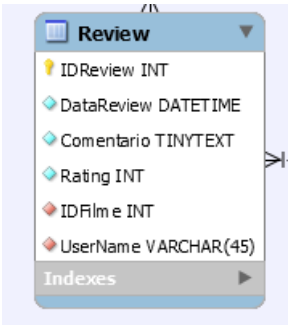

```
CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmeGenero` (
  `FilmeID` INT NOT NULL,
  `GeneroID` INT NOT NULL,
  PRIMARY KEY (`FilmeID`, `GeneroID`),
  INDEX `fk_Filme_has_Género_Género1_idx` (`GeneroID` ASC) VISIBLE,
  INDEX `fk_Filme_has_Género_Filme_idx` (`FilmeID` ASC) VISIBLE,
  CONSTRAINT `fk_Filme_has_Género_Filme`
    FOREIGN KEY (`FilmeID`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Género_Género1`
    FOREIGN KEY (`GeneroID`)
    REFERENCES `filmeDB`.`Género` (`IDGenero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 16 - Conversão da tabela "FilmeGenero"



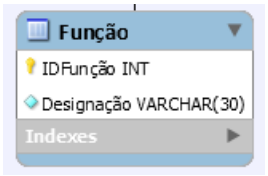

```
CREATE TABLE IF NOT EXISTS `filmeDB`.`User` (
  `UserName` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `Nível` VARCHAR(10) NOT NULL,
  `PalavraChave` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserName`))
ENGINE = InnoDB;
```

Figura 17 - Conversão da tabela "User"



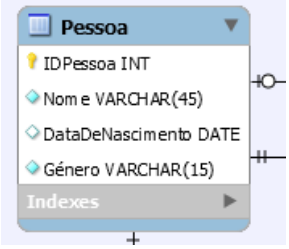

```
75  CREATE TABLE IF NOT EXISTS `filmeDB`.`Review` (
76    `IDReview` INT NOT NULL,
77    `DataReview` DATETIME NOT NULL,
78    `Comentario` TINYTEXT NOT NULL,
79    `Rating` INT NOT NULL,
80    `IDFilme` INT NOT NULL,
81    `UserName` VARCHAR(45) NOT NULL,
82    PRIMARY KEY (`IDReview`),
83    INDEX `fk_Review_Filme1_idx` (`IDFilme` ASC) VISIBLE,
84    INDEX `fk_Review_User1_idx` (`UserName` ASC) VISIBLE,
85    CONSTRAINT `fk_Review_Filme1`
86      FOREIGN KEY (`IDFilme`)
87      REFERENCES `filmeDB`.`Filme` (`IDFilme`)
88      ON DELETE NO ACTION
89      ON UPDATE NO ACTION,
90    CONSTRAINT `fk_Review_User1`
91      FOREIGN KEY (`UserName`)
92      REFERENCES `filmeDB`.`User` (`UserName`)
93      ON DELETE NO ACTION
94      ON UPDATE NO ACTION)
95  ENGINE = InnoDB;
```

Figura 18 - Conversão da tabela "Review"




```
101  CREATE TABLE IF NOT EXISTS `filmeDB`.`Função` (
102    `IDFunção` INT NOT NULL,
103    `Designação` VARCHAR(30) NOT NULL,
104    PRIMARY KEY (`IDFunção`))
105  ENGINE = InnoDB;
```

Figura 19 - Conversão da tabela "Função"




```
111  CREATE TABLE IF NOT EXISTS `filmeDB`.`Pessoa` (
112    `IDPessoa` INT NOT NULL,
113    `Nome` VARCHAR(45) NOT NULL,
114    `DataDeNascimento` DATE NULL,
115    `Género` VARCHAR(15) NOT NULL,
116    PRIMARY KEY (`IDPessoa`))
117  ENGINE = InnoDB;
```

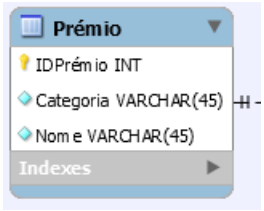
Figura 20 - Conversão da tabela "Pessoa"


```sql
CREATE TABLE IF NOT EXISTS `filmeDB`.`FunçãoPessoaFilme` (
  `IDPessoa` INT NOT NULL,
  `IDFunção` INT NOT NULL,
  `IDFilme` INT NOT NULL,
  `Salário` FLOAT NULL,
  PRIMARY KEY (`IDPessoa`, `IDFunção`, `IDFilme`),
  INDEX `fk_Pessoa_has_Função_Função1_idx` (`IDFunção` ASC) VISIBLE,
  INDEX `fk_Pessoa_has_Função_Pessoa1_idx` (`IDPessoa` ASC) VISIBLE,
  INDEX `fk_FunçãoPessoaFilme_Filme1_idx` (`IDFilme` ASC) VISIBLE,
  CONSTRAINT `fk_Pessoa_has_Função_Pessoa1`
    FOREIGN KEY (`IDPessoa`)
    REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pessoa_has_Função_Função1`
    FOREIGN KEY (`IDFunção`)
    REFERENCES `filmeDB`.`Função` (`IDFunção`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_FunçãoPessoaFilme_Filme1`
    FOREIGN KEY (`IDFilme`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 21 - Conversão da tabela “FunçãoPessoaFilme”

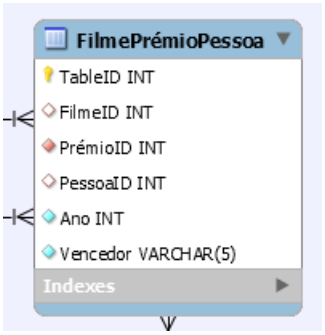

```sql
CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmePessoa` (
  `IDFilme` INT NOT NULL,
  `IDPessoa` INT NOT NULL,
  `Protagonismo` VARCHAR(45) NOT NULL,
  `Personagem` VARCHAR(45) NOT NULL,
  `Salário` FLOAT NULL,
  PRIMARY KEY (`IDFilme`, `IDPessoa`),
  INDEX `fk_Filme_has_Pessoa_Pessoa1_idx` (`IDPessoa` ASC) VISIBLE,
  INDEX `fk_Filme_has_Pessoa_Filme1_idx` (`IDFilme` ASC) VISIBLE,
  CONSTRAINT `fk_Filme_has_Pessoa_Filme1`
    FOREIGN KEY (`IDFilme`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Pessoa_Pessoa1`
    FOREIGN KEY (`IDPessoa`)
    REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 22 - Conversão da tabela “Review”



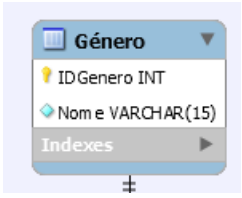

```
178 • CREATE TABLE IF NOT EXISTS `filmeDB`.`Prémio` (
179     `IDPrémio` INT NOT NULL,
180     `Categoria` VARCHAR(45) NOT NULL,
181     `Nome` VARCHAR(45) NOT NULL,
182     PRIMARY KEY (`IDPrémio`))
183   ENGINE = InnoDB;
```

Figura 23 - Conversão da tabela "Prémio"




```
189 • CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmePrémioPessoa` (
190     `TableID` INT NOT NULL,
191     `FilmeID` INT NULL,
192     `PrémioID` INT NOT NULL,
193     `PessoaID` INT NULL,
194     `Ano` INT NOT NULL,
195     `Vencedor` VARCHAR(5) NOT NULL,
196     INDEX `fk_Filme_has_Prémio_Prémio1_idx` (`PrémioID` ASC) VISIBLE,
197     INDEX `fk_Filme_has_Prémio_Filme1_idx` (`FilmeID` ASC) VISIBLE,
198     INDEX `fk_FilmePrémioPessoa_Pessoa1_idx` (`PessoaID` ASC) VISIBLE,
199     PRIMARY KEY (`TableID`),
200     CONSTRAINT `fk_Filme_has_Prémio_Filme1`
201       FOREIGN KEY (`FilmeID`)
202       REFERENCES `filmeDB`.`Filme` (`IDFilme`)
203       ON DELETE NO ACTION
204       ON UPDATE NO ACTION,
205     CONSTRAINT `fk_Filme_has_Prémio_Prémio1`
206       FOREIGN KEY (`PrémioID`)
207       REFERENCES `filmeDB`.`Prémio` (`IDPrémio`)
208       ON DELETE NO ACTION
209       ON UPDATE NO ACTION,
210     CONSTRAINT `fk_FilmePrémioPessoa_Pessoa1`
211       FOREIGN KEY (`PessoaID`)
212       REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
213       ON DELETE NO ACTION
214       ON UPDATE NO ACTION)
215   ENGINE = InnoDB;
```

Figura 24 - Conversão da tabela "FilmePrémioPessoa"




```
31 • CREATE TABLE IF NOT EXISTS `filmeDB`.`Género` (
32     `IDGenero` INT NOT NULL,
33     `Nome` VARCHAR(15) NOT NULL,
34     PRIMARY KEY (`IDGenero`))
35   ENGINE = InnoDB;
```

Figura 25 - Conversão da tabela "Género"

4.3. Tradução das interrogações do utilizador para SQL

Passagem do Requisito de Exploração "Quais os filmes com mais prémios vencidos, incluindo prémios relacionados aos atores" (Query 1) para *MySQL:*

O utilizador deverá poder ver quais os filmes premiados existentes na base de dados e organizados de forma decrescente começando pelo mais premiado.

Primeiramente, foi criada uma subquery a partir de "FROM". Dentro desta subquery, fazemos um count dos prémios de cada filme e organizamos por ordem desde o filme com mais prémios para o que tem menos. Depois usamos um INNER JOIN com FilmePrémioPessoa e Filme, sendo o FilmeID o atributo em comum entre estas duas tabelas. Além disso, como queremos apenas os filmes premiados, usamos "WHERE" de modo a obter apenas os filmes cujo atributo "Vencedor" na tabela FilmePrémioPessoa equivale a "Sim". O "GROUP BY" irá agrupar o resultado numa coluna de acordo com o nome de cada filme.

```
SELECT Título, Premios
FROM  (
    SELECT Nome AS Título, count(*) AS Premios
           ,rank() OVER (ORDER BY count(Nome) DESC)
    FROM Filme
    INNER JOIN FilmePrémioPessoa ON Filme.IDFilme = FilmePrémioPessoa.FilmeID
    WHERE FilmePrémioPessoa.Vencedor = 'Sim'
    GROUP  BY Nome
    ) sub;
```

Figura 26 - Query 1

Passagem do Requisito de Exploração "Qual o rating de um filme tendo por base as avaliações dos utilizadores" (Query 2) para *MySQL:*

A *Query* 2 deverá ser capaz de calcular a média dos ratings atribuídos pelos "User's" a um dado filme. Para traduzir esta interrogação para SQL foi necessário acessar a tabela "Filme" para encontrar o ID do filme correspondente ao nome dado. A essa tabela foi feito um INNER JOIN da tabela "Review", em que foram apenas selecionadas as entradas correspondentes ao "IDFilme" pretendido e, no final foi feita a média de todos os valores da coluna "Rating".

```
SELECT  Filme.Nome, AVG(Rating) AS RatingMedia
FROM filme
INNER JOIN Review ON Review.IDFilme = Filme.IdFilme
WHERE Filme.Nome = 'Aquaman';
```

Figura 27 - Query 2

Passagem do Requisito de Exploração "Quais os atores que também executaram outras funções (diretor, roteirista etc) num dado filme" (Query 3) para *MySQL:*

Para resolver esta query foi necessária a utilização de dois INNER JOIN. O primeiro serviu para associar a tabela Pessoa á tabela FunçãoPessoaFilme, de forma a saber o nome, entre outros dados, pessoas que executaram determinada função num dado filme. No entanto, nós queremos saber quais destas pessoas, também foram atores. Para isto, utilizamos o segundo INNER JOIN que relaciona a tabela anterior á tabela FilmePessoa, sendo que a condição de junção é a igualdade de "idPessoa", uma vez que queremos saber quem executa uma dada função e é ator em simultâneo, e é também a igualdade dos atributos "idFilme", pois caso uma determinada pessoa atuasse num filme e fosse diretor noutro, apareceria na tabela resultado, mas apenas queremos atores que executaram funções backstage num mesmo filme.

```
SELECT DISTINCT Nome
FROM Pessoa
INNER JOIN FunçãoPessoaFilme ON Pessoa.IDPessoa = FunçãoPessoaFilme.IDPessoa
INNER JOIN FilmePessoa ON (FunçãoPessoaFilme.IdPessoa = FilmePessoa.IDPessoa AND FunçãoPessoaFilme.IdFilme = FilmePessoa.IDFilme);
```

Figura 28 - Query 3

Passagem do Requisito de Exploração "Quantos filmes de um dado ator estrearam depois de uma data" (Query 4) para *MySQL:*

Para resolver esta query foi necessária a utilização dois INNER JOIN. O primeiro serviu para associar a cada filme os ids de todos os atores que nele participaram. O segundo INNER JOIN serviu para associar os ids das pessoas aos nomes dos respetivos atores. Ao fim disto, tínhamos associado a cada filme o nome de todos os seus atores e bastou utilizar um *WHERE* de forma a filtrar apenas os filmes que estrearam após um dado ano com um dado ator (escolhemos o ator Leonardo Dicaprio e o ano 2005 apenas para exemplo)

```
SELECT Filme.Nome AS TítuloFilme
FROM Filme
INNER JOIN FilmePessoa ON Filme.IDFilme = FilmePessoa.IDFilme
INNER JOIN Pessoa ON FilmePessoa.IdPessoa = Pessoa.IdPessoa
WHERE Pessoa.Nome='Leonardo Dicaprio' AND YEAR(DataEstreia)>2005;
```

Figura 29 - Query 4

Passagem do Requisito de Exploração "Determinar o *cast* e a *crew* de um filme" (Query 5) para *MySQL:*

A *Query* 5 deverá ser capaz de apresentar o nome de todas as pessoas envolvidas num dado filme. Para traduzir esta interrogação para SQL foi necessário juntar ambas as tabelas "FilmePessoa" e "FunçãoPessoaFilme" visto que, na primeira estão todos os elementos do *Cast* e, na segunda os da *Crew*. A cada uma delas foi feito um INNER JOIN da tabela "Pessoa" para que sejam acessados os seus nomes. Por último foi necessário o uso de mais um INNER JOIN da tabela "Filme" para conseguir aceder ao "IDFilme" referente ao nome do filme fornecido.

```
SELECT DISTINCT Pessoa.Nome
FROM FilmePessoa
INNER JOIN Pessoa ON FilmePessoa.IdPessoa = Pessoa.IDPessoa
INNER JOIN Filme ON FilmePessoa.IDFilme = Filme.IDFilme
WHERE Filme.Nome = 'Aquaman'
UNION
SELECT DISTINCT Pessoa.Nome
FROM FunçãoPessoaFilme
INNER JOIN Pessoa ON FunçãoPessoaFilme.IdPessoa = Pessoa.IDPessoa
INNER JOIN Filme ON FunçãoPessoaFilme.IDFilme = Filme.IDFilme
WHERE Filme.Nome = 'Aquaman';
```

Figura 30 - Query 5

Passagem do Requisito de Exploração "Mulheres que participaram num filme seja como atrizes ou trabalhadoras de *backstage*" (Query 6) para *MySQL:*

Esta query foi realizada com o intuito de tentar aumentar a complexidade na resolução das *queries*. O utilizador poderá obter o nome das mulheres que participam num dado filme, seja como atrizes ou como trabalhadoras de backstage. Para a resolução desta query foi necessário a utilização de um UNION de forma a unir o resultado dos dois SELECT. No primeiro select, usamos dois INNER JOIN, um do "FilmePessoa" com Filme onde o atributo em comum é o IDFilme, e outro entre FilmePessoa e Pessoa, sendo o atributo em comum o "IDPessoa". De seguida usamos o WHERE de forma a obter apenas os resultados cujo género corresponde a "F" (feminino), e cujo filme corresponde ao que tivemos em consideração, neste caso usamos como exemplo o filme Titanic. Sendo assim, obtemos todas as atrizes que participaram no filme, isto é, todos os membros do cast que são do sexo feminino.

Em relação ao segundo SELECT, a ideia é basicamente a mesma aplicada ao primeiro, mas neste, os "INNER JOIN" em vez de usarmos a tabela FilmePessoa, usamos a tabela FunçãoFilmePessoa, uma vez que neste caso queremos saber as mulheres que fazem parte da Crew do filme. A aplicação do WHERE é a mesma de modo a obtermos apenas as mulheres que trabalharam no backstage do filme Titanic.

```sql
SELECT DISTINCT Pessoa.Nome
FROM Filme
INNER JOIN FilmePessoa ON Filme.IdFilme = FilmePessoa.IdFilme
INNER JOIN Pessoa ON FilmePessoa.IdPessoa = Pessoa.IDPessoa
WHERE Filme.Nome = 'Titanic' AND Pessoa.Género = 'F'
UNION
SELECT DISTINCT Pessoa.Nome
FROM Filme
INNER JOIN FunçãoPessoaFilme ON Filme.IdFilme = FunçãoPessoaFilme.IdFilme
INNER JOIN Pessoa ON FunçãoPessoaFilme.IdPessoa = Pessoa.IDPessoa
WHERE Filme.Nome = 'Titanic' AND Pessoa.Género = 'F' ;
```

Figura 31 - Query 6

- **Nota Importante:**

O atributo "nível" de um utilizador deverá incrementar cada vez que o número de reviews deste atinge um determinado valor. Relembramos que consideramos a existência de 3 níveis, sendo estes: *newbie*, *active* e *big fan*. Cada um destes tem associado um número mínimo de reviews necessárias para um utilizador atingir o nível respetivo. Para isto acontecer, seria necessário a implementação de um *trigger* que faria esta atualização.

4.4. Escolha, definição e caracterização de índices em SQL

Após a criação do modelo lógico do projeto em SQL, foi utilizada a ferramenta *forward engineering*, que gerou o script com as instruções necessárias para a criação das tabelas e da base de dados. Por definição, esta ferramenta já cria alguns índices, no entanto decidimos criar outros que o grupo achou relevante. No entanto, dado a dimensão da nossa base de dados, não se justifica a criação de uma grande quantidade de índices.

Decidimos criar três índices sobre o atributo língua e PG da tabela filme e sobre o atributo "AnoPrémio" da tabela "*FilmePrémioPessoa*". Isto torna mais eficiente *queries* que impliquem a procura de filmes de uma dada língua ou PG, e vencedores de um dado prémio num determinado ano.

```
CREATE INDEX idx_Língua
    ON Filme (Língua);

CREATE INDEX idx_AnoPrémio
    ON FilmePrémioPessoa (Ano);

CREATE INDEX idx_PG
    ON Filme (PG);
```

Figura 32 - Índices

É de realçar que estes índices são apenas exemplo e poderiam ter sido escolhidos outros. Mas, como já foi referido, o impacto na eficiência será muito reduzido devido á pequena quantidade de dados na base de dados.

4.5. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

	Atributo	Tipo	Tamanho
Filme	IDFilme	INT	4 bytes
	Nome	VARCHAR (45)	45 bytes
	Custo	FLOAT	4 bytes
	País	VARCHAR (20)	20 bytes
	Língua	VARCHAR (20)	20 bytes
	Duração	TIME	5 bytes
	Receita	FLOAT	4 bytes
	PG	INT	4 bytes
	Descrição	TINYTEXT	255 bytes
	DataEstreia	DATE	3 bytes
			364 bytes

Tabela 3 - Estimativa do espaço: Filme

- Existem 9 filmes na nossa base de dados, pelo que a tabela *"Filme"* ocupará 9* 364 = 3276 bytes.

	Atributo	Tipo	Tamanho
Género	IDGénero	INT	4 bytes
	Nome	VARCHAR (15)	15 bytes
			19 bytes

Tabela 4 - Estimativa do Espaço: Género

- Existem 16 géneros na nossa base de dados pelo que a tabela *"Género"* ocupará 16*19 = 304 bytes.

	Atributo	Tipo	Tamanho
User	UserName	VARCHAR (45)	45 bytes
	Email	VARCHAR (45)	45 bytes
	Nível	VARCHAR (10)	10 bytes
	PalavraChave	VARCHAR (45)	45 bytes
			145 bytes

Tabela 5 - Estimativa do Espaço: User

- Existem 10 *users* na nossa base de dados pelo que a tabela *"User"* ocupará 10*145 = 1450 bytes.

	Atributo	Tipo	Tamanho
Função	IDFunção	INT	4 bytes
	Designação	VARCHAR (30)	30 bytes
			34 bytes

Tabela 6 - Estimativa do Espaço: Função

- Existem 6 funções na nossa base de dados pelo que a tabela *"Função"* ocupará 6*34 = 34 bytes.

	Atributo	Tipo	Tamanho
Prémio	IDPrémio	INT	4 bytes
	Categoria	VARCHAR (45)	45 bytes
	Nome	VARCHAR (45)	45 bytes
			94 bytes

Tabela 7 - Estimativa do Espaço: Prémio

- Existem 16 prémios na nossa base de dados pelo que a tabela *"Prémio"* ocupará 16*94 = 1504 bytes.

	Atributo	Tipo	Tamanho
Pessoa	IDPessoa	INT	4 bytes
	Nome	VARCHAR (45)	45 bytes
	DataNascimento	DATE	3 bytes
	Género	VARCHAR (15)	15 bytes
			67 bytes

Tabela 8 - Estimativa do Espaço: Pessoa

- Existem 63 pessoas na nossa base de dados pelo que a tabela *"Pessoa"* ocupará 63*67 = 4221 bytes.

	Atributo	Tipo	Tamanho
Review	IDReview	INT	4 bytes
	DataReview	VARCHAR (45)	45 bytes
	Comentario	INT	4 bytes
	Rating	VARCHAR (20)	20 bytes
	IDFilme	VARCHAR (20)	20 bytes
	UserName	VARCHAR (45)	45 bytes
			138 bytes

Tabela 9 - Estimativa do Espaço: Review

- Existem 14 *reviews* na nossa base de dados pelo que a tabela "*Review"* ocupará 14*138 = 1932 bytes.

	Atributo	Tipo	Tamanho
FilmeGenero	FilmeID	INT	4 bytes
	GeneroID	INT	4 bytes
			8 bytes

Tabela 10 - Estimativa do Espaço: FilmeGenero

- Existem 25 elementos na tabela "*FilmeGenero"* pelo que esta ocupará 25*8 = 200 bytes.

	Atributo	Tipo	Tamanho
FunçãoPessoaFilme	IDPessoa	INT	4 bytes
	IDFunção	INT	4 bytes
	IDFilme	INT	4 bytes
	Salário	FLOAT	4 bytes
			16 bytes

Tabela 11 - Estimativa do Espaço: FunçãoPessoaFilme

- Existem 44 elementos na tabela "*FunçãoPessoaFilme*" pelo que esta ocupará 44*16 = 704 bytes.

	Atributo	Tipo	Tamanho
FilmePrémioPessoa	TableID	INT	4 bytes
	FilmeID	INT	4 bytes
	PrémioID	INT	4 bytes
	PessoaID	INT	4 bytes
	Ano	INT	4 bytes
	Vencedor	VARCHAR (5)	5 bytes
			25 bytes

Tabela 12 - Estimativa do Espaço: FilmePrémioPessoa

- Existem 32 elementos na tabela "*FilmePrémioPessoa*" pelo que esta ocupará 32*25 = 800 bytes.

	Atributo	Tipo	Tamanho
FilmePessoa	IDFilme	INT	4 bytes
	IDPessoa	INT	4 bytes
	Protagonismo	VARCHAR (45)	45 bytes
	Personagem	VARCHAR (45)	45 bytes
	Salário	FLOAT	4 bytes
			102 bytes

Tabela 13 - Estimativa do Espaço: FilmePessoa

- Existem 28 elementos na tabela "*FilmePessoa*" pelo que esta ocupará 28*102 = 2856 bytes.

Tendo em conta todos os valores apresentados anteriormente, a base de dados necessita de 17281 bytes, ou seja, 17.281 kilobytes.

Após algumas pesquisas, concluímos que o crescimento de utilizadores de um dado *serviço streaming,* em média*,* ronda os 11.9% ao ano. Além disto, o aumento da quantidade filmes disponíveis ronda os 10.7% ao ano. Consideremos estes valores e que não existem mudanças significativas quanto aos géneros, prémios ou funções já definidas previamente. Por outro lado, é de realçar o seguinte:

- se houver um aumento do número de filmes, haverá também um aumento de pessoas (atores, diretores, entre outros) e de filmes de um dado género presentes na base de dados (isto afeta as tabelas "*Pessoa*" e "*FilmePessoa*" e *"FilmeGenero"*). Consideramos esta taxa de crescimento igual ao dos filmes (10.7%/ano).

- Se houver um aumento do número de pessoas e do número de filmes, haverá também um aumento de pessoas e filmes nomeados a prémios (isto afeta a tabela "*FilmePrémioPessoa"*). Consideramos esta taxa de crescimento igual ao dos filmes (10.7%/ano).

- Se houver um aumento do número de pessoa e de filmes haverá também um aumento de pessoas a executar dadas funções num dado filme (isto afeta a tabela "*FunçãoPessoaFilme"*). Consideramos esta taxa de crescimento igual ao dos filmes (10.7%/ano).

- Se há um aumento de utilizadores, haverá um aumento de avaliações (isto afeta a tabela "*Review"*). Consideramos esta taxa de crescimento igual á dos utilizadores (11.9%/ano)

Com isto obtemos o seguinte:

0.107 * 3276 + 0.107 *4221 + 0.107 * 200 + 0.107 * 704 + 0.107 * 800 + 0.107 * 2856 + 0.119 * 1450 + 0.119 * 1932 = 1692.557 bytes

Concluindo, a nossa base de dados possui crescimento anual de 1692.557 bytes, ou seja, aproximadamente 1.693 kilobytes/ano.

4.6. Definição e caracterização das vistas de utilização em SQL

O grupo decidiu criar algumas vistas, de forma a que os utilizadores possam ver o nível em que se encontram, os dados mais relevantes de um filme e as *reviews* de outros utilizadores.

```
CREATE VIEW view_Filme AS
SELECT Nome ,Língua,Duração,PG,Descrição
FROM Filme;

CREATE VIEW view_User AS
SELECT nível
FROM User;

CREATE VIEW view_Reviews AS
SELECT Nome,UserName,dataReview,Rating,Comentario
FROM Filme INNER JOIN Review ON Filme.idFilme = Review.IdFilme;
```

Tabela 14 - Vistas

4.7. Revisão do sistema implementado

Após concluirmos a implementação física da nossa base de dados, houve uma reavaliação global por parte do grupo de forma a garantir que não havia incoerências ou problemas que não tenham sido detetados anteriormente.

5. Conclusão e trabalhos futuros

Ao longo desta primeira fase do projeto, foi possível consolidar todo o conhecimento adquirido nas aulas teóricas quanto ao processo de criação de uma base de dados. No entanto, e como já era esperado, foram surgindo várias dificuldades, que o grupo pensa ter conseguido ultrapassar com sucesso.

Durante o levantamento e análise de requisitos, houve alguma discussão entre os elementos do grupo relativamente á relevância de alguns atributos e entidades.

Posteriormente, durante a modelação concetual, as grandes dificuldades residiram essencialmente em identificar relacionamentos que pudessem ser redundantes e em definir cardinalidades dos relacionamentos ternários.

Durante a fases posteriores, surgiram algumas adversidades, nomeadamente em relação ao tipo dos atributos e em relação á tabela *FilmePrémioPessoa*, como já foi explicado neste relatório. No entanto, o grupo tentou sempre, através de discussões, reavaliações, mudanças e alterações constantes, ir criando um projeto sólido de forma a que não aparecessem redundâncias ou incoerências inesperadas.

Desta forma, pensamos ter um projeto bem conseguido, tendo em conta que este foi o nosso primeiro contato com a disciplina, com as metodologias lecionadas e com as ferramentas *BRModelo* e *MySQL Workbench*.

Referências

Connolly, T., Begg, C., Database Systems: A Practical Approach to Design, Implementation, and Management, Addison Wesley, Global Edition, 26 Sep 2014

Lista de Siglas e Acrónimos

BD Base de Dados

TV Televisão

ID Identidade/Identificador

ER Entity-Relationship

UC Unidade Curricular

Anexos

I. Anexo 1 - Modelo Físico no MySQL

```
-- MySQL Workbench Forward Engineering

-- -----------------------------------------------------
-- Schema filmeDB
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `filmeDB` DEFAULT CHARACTER SET utf8 ;
USE `filmeDB` ;
-- DROP SCHEMA `filmeDB`

-- -----------------------------------------------------
-- Table `filmeDB`.`Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Filme` (
  `IDFilme` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Custo` FLOAT NOT NULL,
  `País` VARCHAR(20) NOT NULL,
  `Língua` VARCHAR(20) NOT NULL,
  `Duração` TIME NULL,
  `Receita` FLOAT NOT NULL,
  `PG` INT NOT NULL,
  `Descrição` TINYTEXT NOT NULL,
  `DataEstreia` DATE NOT NULL,
  PRIMARY KEY (`IDFilme`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`Género`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Género` (
  `IDGenero` INT NOT NULL,
  `Nome` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`IDGenero`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`FilmeGenero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmeGenero` (
  `FilmeID` INT NOT NULL,
  `GeneroID` INT NOT NULL,
  PRIMARY KEY (`FilmeID`, `GeneroID`),
  INDEX `fk_Filme_has_Género_Género1_idx` (`GeneroID` ASC) VISIBLE,
  INDEX `fk_Filme_has_Género_Filme_idx` (`FilmeID` ASC) VISIBLE,
  CONSTRAINT `fk_Filme_has_Género_Filme`
    FOREIGN KEY (`FilmeID`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Género_Género1`
    FOREIGN KEY (`GeneroID`)
    REFERENCES `filmeDB`.`Género` (`IDGenero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`User`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`User` (
  `UserName` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `Nível` VARCHAR(10) NOT NULL,
  `PalavraChave` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`UserName`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`Review`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Review` (
  `IDReview` INT NOT NULL,
  `DataReview` DATETIME NOT NULL,
  `Comentario` TINYTEXT NOT NULL,
  `Rating` INT NOT NULL,
  `IDFilme` INT NOT NULL,
  `UserName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`IDReview`),
  INDEX `fk_Review_Filme1_idx` (`IDFilme` ASC) VISIBLE,
  INDEX `fk_Review_User1_idx` (`UserName` ASC) VISIBLE,
  CONSTRAINT `fk_Review_Filme1`
    FOREIGN KEY (`IDFilme`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Review_User1`
    FOREIGN KEY (`UserName`)
    REFERENCES `filmeDB`.`User` (`UserName`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`Função`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Função` (
  `IDFunção` INT NOT NULL,
  `Designação` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`IDFunção`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `filmeDB`.`Pessoa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Pessoa` (
  `IDPessoa` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `DataDeNascimento` DATE NULL,
  `Género` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`IDPessoa`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`FunçãoPessoaFilme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`FunçãoPessoaFilme` (
  `IDPessoa` INT NOT NULL,
  `IDFunção` INT NOT NULL,
  `IDFilme` INT NOT NULL,
  `Salário` FLOAT NULL,
  PRIMARY KEY (`IDPessoa`, `IDFunção`, `IDFilme`),
  INDEX `fk_Pessoa_has_Função_Função1_idx` (`IDFunção` ASC) VISIBLE,
  INDEX `fk_Pessoa_has_Função_Pessoa1_idx` (`IDPessoa` ASC) VISIBLE,
  INDEX `fk_FunçãoPessoaFilme_Filme1_idx` (`IDFilme` ASC) VISIBLE,
  CONSTRAINT `fk_Pessoa_has_Função_Pessoa1`
    FOREIGN KEY (`IDPessoa`)
    REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pessoa_has_Função_Função1`
    FOREIGN KEY (`IDFunção`)
    REFERENCES `filmeDB`.`Função` (`IDFunção`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_FunçãoPessoaFilme_Filme1`
    FOREIGN KEY (`IDFilme`)
```

```
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`FilmePessoa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmePessoa` (
  `IDFilme` INT NOT NULL,
  `IDPessoa` INT NOT NULL,
  `Protagonismo` VARCHAR(45) NOT NULL,
  `Personagem` VARCHAR(45) NOT NULL,
  `Salário` FLOAT NULL,
  PRIMARY KEY (`IDFilme`, `IDPessoa`),
  INDEX `fk_Filme_has_Pessoa_Pessoa1_idx` (`IDPessoa` ASC) VISIBLE,
  INDEX `fk_Filme_has_Pessoa_Filme1_idx` (`IDFilme` ASC) VISIBLE,
  CONSTRAINT `fk_Filme_has_Pessoa_Filme1`
    FOREIGN KEY (`IDFilme`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Pessoa_Pessoa1`
    FOREIGN KEY (`IDPessoa`)
    REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`Prémio`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`Prémio` (
  `IDPrémio` INT NOT NULL,
  `Categoria` VARCHAR(45) NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`IDPrémio`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `filmeDB`.`FilmePrémioPessoa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `filmeDB`.`FilmePrémioPessoa` (
  `TableID` INT NOT NULL,
  `FilmeID` INT NULL,
  `PrémioID` INT NOT NULL,
  `PessoaID` INT NULL,
  `Ano` INT NOT NULL,
  `Vencedor` VARCHAR(5) NOT NULL,
  INDEX `fk_Filme_has_Prémio_Prémio1_idx` (`PrémioID` ASC) VISIBLE,
  INDEX `fk_Filme_has_Prémio_Filme1_idx` (`FilmeID` ASC) VISIBLE,
  INDEX `fk_FilmePrémioPessoa_Pessoa1_idx` (`PessoaID` ASC) VISIBLE,
  PRIMARY KEY (`TableID`),
  CONSTRAINT `fk_Filme_has_Prémio_Filme1`
    FOREIGN KEY (`FilmeID`)
    REFERENCES `filmeDB`.`Filme` (`IDFilme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Prémio_Prémio1`
    FOREIGN KEY (`PrémioID`)
    REFERENCES `filmeDB`.`Prémio` (`IDPrémio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_FilmePrémioPessoa_Pessoa1`
    FOREIGN KEY (`PessoaID`)
    REFERENCES `filmeDB`.`Pessoa` (`IDPessoa`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

II. Anexo 2 - Povoamento da Base de Dados

```
-- Implementação e Exploração de Sistemas de Bases de Dados Relacionais
-- O caso de Estudo da "Movie Database"
-- Povoamento da Base de Dados Relacional

USE `filmeDB` ;
-- DELETE FROM `filmeDB`.`Filme`;
-- SELECT * FROM `filmeDB`.`Filme`;
INSERT INTO `filmeDB`.`Filme`
-- Custo é em doláres
(IdFilme,Nome,Custo,País,Língua,Duração,Receita,PG,Descrição,DataEstreia)
VALUES
        ('1','Titanic','200000000','EUA','Inglês','3:14','1200000000','12','A young aristocrat falls in love with a kind but poor artist','1997-12-19'),
  ('2','The Shawshank Redemption','25000000','EUA','Inglês','2:22','58300000','16','Two imprisoned men bond over a number of years, finding redemption','1994-11-23'),
        ('3','Aquaman','160000000','EUA','Inglês','2:23','1148000000','12','Arthur Curry goes on a quest to prevent a war between  ocean and land.','2018-11-21'),
        ('4','Parasite','11400000','Coreia do Sul','Coreano','2:12','264400000','14','Greed and class discrimination threaten the relationship between  Park family and the Kim clan.','2019-05-21'),
  ('5','Gladiator','103000000','EUA','Inglês','2:35','46050000','12','A former Roman General sets out to exact vengeance against the corrupt emperor','2000-05-19'),
  ('6','By The Sea','10000000','EUA','Inglês','2:02','3334927','18','A couple tries to repair their marriage while staying at a hotel in France.','2015-12-09'),
        ('7','Inception','160000000','EUA','Inglês','2:28','836800000','12','A thief who steals corporate secrets through the use of dream-sharing technology','2010-06-22'),
  ('8','Inside Out','175000000','EUA','Inglês','1:35','857611174','6','Young Riley conflicts on how to navigate on new city.','2015-06-18'),
  ('9','La Vie en Rose','25000000','França','Francês','2:20','86300000','12','Biopic of the iconic French singer Édith Piaf.','2009-04-25');
```

```
 -------//---------

-- DELETE FROM `mydb`.`Género`;
-- SELECT * FROM `mydb`.`Género`;
INSERT INTO `filmeDB`.`Género`
(IDGenero,Nome)
VALUES
            ('1','Ação'),
   ('2','Aventura'),
            ('3','Romance'),
            ('4','Terror'),
   ('5', 'Comédia'),
   ('6','Suspense'),
   ('7', 'Animação'),
   ('8', 'Familiar'),
   ('9', 'Crime'),
   ('10', 'Drama'),
   ('11','Fantasia'),
   ('12', 'Sci-Fi'),
   ('13','Mistério'),
   ('14', 'Biografia'),
   ('15', 'Thriller'),
   ('16', 'Musical');
   -----//------


-- DELETE FROM `filmeDB`.`FilmeGenero`;
-- SELECT * FROM `filmeDB`.`filmeGenero`;

INSERT INTO `filmeDB`.`FilmeGenero`
(FilmeID,GeneroID)
VALUES
            ('1','10'),
   ('1','3'),
   ('2','10'),
            ('2','9'),
            ('3','1'),
```

```
   ('3','2'),
   ('3','11'),
            ('4','10'),
   ('4','5'),
   ('4','15'),
   ('4','4'),
   ('5', '1'),
   ('5', '2'),
   ('5', '10'),
   ('6','3'),
   ('6','10'),
   ('7', '1'),
   ('7', '2'),
   ('7', '12'),
   ('8', '2'),
   ('8', '5'),
   ('8', '8'),
   ('9', '10'),
   ('9', '14'),
   ('9', '16');

   -------------------------

   INSERT INTO `filmeDB`.`Função`
   -- SELECT * FROM `filmeDB`.`Função`;
(IDFunção,Designação)
VALUES
   ('1','Diretor'),
   ('2','Figurinista'),
   ('3','Compositor'),
   ('4','Técnico de Efeitos Especiais'),
   ('5','Roteirista');


    INSERT INTO `filmeDB`.`User`
    -- SELECT * FROM `filmeDB`.`User`;
(Username,email,nível,PalavraChave)
VALUES
```

```
('MaryC','maryc@gmail.com','newbie','mary'),
('AndyW','andyW@hotmail.com','big fan','andy'),
('kelinhaKosta','kelinhaKosta@gmail.com','active','kelinha'),
('CatarinaSantejo','catSantejo@hotmail.com','newbie','catarina32'),
('JosieD','josie@outlook.com','active','josie'),
('HelenR','helenR@hotmail.com','newbie','helen123'),
('RichardE','richardE@gmail.com','newbie','richardM'),
('PaulP','paulp@gmail.com','newbie','paulnoC'),
('RossT','rosst@outlook.com','active','rossie'),
('jamesQ','jamesquinn@hotmail.com','newbie','jamess');



INSERT INTO `filmeDB`.`Review`
    -- SELECT * FROM `filmeDB`.`Review`;
(IDReview,dataReview,Comentario,Rating,IDFilme,UserName)
VALUES
('1','2011-11-14','Probably the best French film for years!','10','9','MaryC'),
('2','2018-01-01','Boring, shallow and overrated...','2','8','PaulP'),
('3','2005-11-25','"Gladiator" brought a poetic vision in a new and very cinematically richly way...','9','5','kelinhaKosta'),
('4','2001-06-13','Great Story but Far From Historical.','6','5','PaulP'),
('5','2018-07-03','Despite a lot of plot flaws and conveniences, this really is one of the best films ever made.','10','1','JosieD'),
('6','2020-01-12','An original dark comedy about class struggles.','8','4','HelenR'),
('7','2020-01-04','Too hyped, not too bad.','6','4','PaulP'),
('8','2007-05-18','Marion is spectacular but the film is needlessly melodramatic...','7','9','jamesQ'),
('9','1998-08-02','The closest thing to poetic perfection Hollywood has ever produced.','10','2','AndyW'),
('10','2020-11-07','Amber heard did a terrible acting job and should be fired.','1','3','HelenR'),
('11','2019-01-06','The most overrated generic superhero film you will ever see.','4','3','RossT'),
('12','2016-03-30','A tender and accurate portrayal ','8','6','CatarinaSantejo'),
('13','2016-10-29','Ridiculously overrated, with plot holes that are impossible to ignore.','4','6','RichardE'),
('14','2010-07-15','Sci-fi perfection. A truly mesmerizing film.','8','7','MaryC');

----------
```

```
INSERT INTO `filmeDB`.`Prémio`
    -- SELECT * FROM `filmeDB`.`Prémio`;

(IDPrémio,Categoria,Nome)
VALUES
('1','Melhor Filme','Oscar'),
('2','Melhor Filme Estrangeiro','Oscar'),
('3','Melhor Diretor','Oscar'),
('4','Melhor Ator Principal ','Oscar'),
('5','Melhor Ator Coadjuvante','Oscar'),
('6','Melhor Atriz Principal ','Oscar'),
('7','Melhor Atriz Coadjuvante','Oscar'),
('8','Melhor Roteiro Original','Oscar'),
('9','Melhor Trilha Sonora','Oscar'),
('10','Melhor Figurino','Oscar'),
('11','Melhores Efeitos Especiais','Oscar'),
('12','Honorário' ,'Oscar'),
('13','Melhor Filme de Animação','Oscar'),
('14','Melhor Ator Principal','Cesar'),
('15','Melhor Filme','Cesar'),
('16','Melhor realizador','Cesar');


   INSERT INTO `filmeDB`.`Pessoa`
    -- SELECT * FROM `filmeDB`.`pessoa`;
(IDPessoa,Nome,DataDeNascimento,Género)
VALUES
('1','Angelina Jolie','1975-06-04','F'),
('2','Walt Disney','1901-12-05','M'),
('3','James Cameron','1954-08-16','M'),
('4','Kate Winslet','1975-10-05','F'),
('5','Gloria Stuart','1910-07-04','F'),
('6','Deborah Lynn Scott','1954-10-29','F'),
('7','Robert Legato','1956-05-06','M'),
('8','James Horner','1953-08-14','M'),
('9','Morgan Freeman','1937-07-01','M'),
('10','Frank Darabont','1959-01-28','M'),
('11','Leonardo DiCaprio','1974-11-11','M'),
```

('12','Thomas Newman','1955-10-20','M'),

('13','Tim Robbins','1958-10-16','M'),

('14','Bob Gunton','1945-11-15','M'),

('15','Bob Williams','1974-02-13','M'),

('16','Elizabeth McBride','1995-05-17','F'),

('17','James Wan','1977-02-27','M'),

('18','David Leslie Johnson-McGoldrick ','1942-10-05','M'),

('19','Kym Barrett','1965-10-11','F'),

('20','Rupert Gregson-Williams','1966-11-12','M'),

('21','Blair Berens','1995-06-28','F'),

('22','Jason Momoa','1979-08-01','M'),

('23','Amber Heard','1986-04-22','F'),

('24','Willem Dafoe','1955-07-22','M'),

('25','Nicole Kidman','1967-06-20','F'),

('26','Bong Joon Ho','1967-06-20','M'),

('27','Se-yeon Choi','1967-06-20','F'),

('28','Jaeil Jung','1967-06-20','M'),

('29','Hyo-kyun Hwang','1976-05-26','M'),

('30','Song Kang Ho','1967-02-25','M'),

('31','Lee Sun Kyun','1975-03-02','M'),

('32','Cho Yeo Jeong','1981-02-10','F'),

('33','Choi Woo Shik','1990-03-26','M'),

('34','Ellen Mirojnick','1949-07-07','F'),

('35','Gabriel Yared','1949-11-07','M'),

('36','Jennifer C. Bell','1964-08-26','F'),

('37','Mélanie Laurent','1983-02-21','F'),

('38','Brad Pitt','1963-12-18','M'),

('39','Russell Crowe','1964-04-07','M'),

('40','Joaquin Phoenix','1974-10-28','M'),

('41','Ridley Scott','1937-11-30','M'),

('42','Janty Yates','1950-05-26','F'),

('43','Hans Zimmer','1957-07-12','M'),

('44','John Nelson','1941-12-06','M'),

('45','David Franzoni','1947-03-04','M'),

('46','Chris Corbould', '1958-03-05','M'),

('47','Christopher Nolan','1970-07-30','M'),

('48','Jeffrey Kurland','1952-03-10','M'),

('49','Marion Cotillard','1975-09-30','F'),

```
('50','Ellen Page','1987-02-21','F'),
('51','Amy Poehler','1971-09-16','F'),
('52','Phyllis Smith','1951-07-10','F'),
('53','Kaitlyn Dias','1999-05-11','F'),
('54','Pete Docter','1968-10-09','M'),
('55','Michael Giacchino','1967-10-10','M'),
('56','Frank Aalbers',NULL,'M'),
('57','Emmanuelle Seigner','1966-06-22','F'),
('58','Gérard Depardieu','1948-12-27','M'),
('59','Olivier Dahan','1967-06-26','M'),
('60','Isabelle Sobelman', NULL,'F'),
('61','Christopher Gunning','1944-08-05','M'),
('62','Marit Allen','1941-09-17','F'),
('63','Jan Holub','1983-05-03','M');



INSERT INTO `filmedb`.`FilmePrémioPessoa`
-- SELECT * FROM `filmeDB`.`FilmePrémioPessoa`;
(TableID,FilmeID,PrémioID,PessoaID,Ano,Vencedor)
VALUES
('1', NULL,'12','2','1932','Sim'),
('2','1','1',NULL,'1998','Sim'),
('3','1','3','3','1998','Sim'),
('4','1','6','4','1998','Não'),
('5','1','7','5','1998','Não'),
('6','1','9','8','1998','Sim'),
('7','1','10','6','1998','Sim'),
('8','1','11','7','1998','Sim'),
('9','2','1',NULL,'1995','Não'),
('10','2','4','9','1995','Não'),
('11','2','8','10','1995','Não'),
('12','2','9','12','1995','Não'),
('13','4','1',NULL,'2020','Sim'),
('14','4','8','26','2020','Sim'),
('15','4','3','26','2020','Sim'),
('16','4','2',NULL,'2020','Sim'),
('17','6','1',NULL,'2001','Sim'),
```

```
('18','6','4','39','2001','Sim'),
('19','6','10','42','2001','Sim'),
('20','6','11','44','2001','Sim'),
('21','6','5','40','2001','Não'),
('22','6','3','41','2001','Não'),
('23','6','8','45','2001','Não'),
('24','6','9','43','2001','Não'),
('25','7','11','46','2011','Sim'),
('26','7','1',NULL,'2011','Não'),
('27','7','8','47','2011','Não'),
('28','7','9','43','2011','Não'),
('29','8','13',NULL,'2016','Sim'),
('30','8','8','54','2016','Não'),
('31','9','6','49','2008','Sim'),
('32','9','10','62','2008','Não');


        INSERT INTO `filmedb`.`FunçãoPessoaFilme`
  -- SELECT * FROM `filmeDB`.`FunçãoPessoaFilme`;
(IDPessoa,IDFunção,IDFilme,Salário)
VALUES
('3','1','1','200000'),
('6','2','1',NULL),
('7','4','1',NULL),
('8','3','1',NULL),
('3','5','1','200000'),
('10','1','2','16000'),
('12','3','2',NULL),
('15','4','2',NULL),
('16','2','2',NULL),
('10','5','2','16000'),
('17','1','3','98000'),
('19','2','3','9500'),
('20','3','3',NULL),
('21','4','3',NULL),
('18','5','3','98000'),
('26','1','4','100000'),
('27','2','4',NULL),
```

```
('28','3','4','9000'),
('29','4','4',NULL),
('26','5','4','100000'),
('1','1','5','110000'),
('34','2','5',NULL),
('35','3','5','11000'),
('36','4','5','9500'),
('1','5','5','110000'),
('41','1','6',NULL),
('42','2','6',NULL),
('43','3','6','15000'),
('44','4','6',NULL),
('45','5','6',NULL),
('47','1','7',NULL),
('48','2','7',NULL),
('43','3','7','165000'),
('46','4','7',NULL),
('47','5','7',NULL),
('54','1','8','15050'),
('55','3','8',NULL),
('56','4','8',NULL),
('54','5','8',NULL),
('59','1','9','12000'),
('62','2','9',NULL),
('61','3','9',NULL),
('63','4','9',NULL),
('60','5','9',NULL);


        INSERT INTO `filmedb`.`filmePessoa`
  -- SELECT * FROM `filmeDB`.`filmePessoa`;
(IDFilme,IDPessoa,Protagonismo,Personagem,Salário)
VALUES
('1','4','Principal','Rose Dewitt Bukater','1000000'),
('1','5','Secundária','Rose Dewitt Bukater','50000'),
('1','11','Principal','Jack Dawson','20000000'),
('2','9','Principal','Ellis Boyd Redding','2000000'),
('2','13','Principal','Andy Dufresne','300000'),
```

('2','14','Secundária','Warden Norton','50000'),

('3','22','Principal','Arthur','800000'),

('3','23','Principal','Mera','500000'),

('3','24','Secundária','Vulko','100000'),

('3','25','Secundária','Atlanna','100000'),

('4','30','Principal','Ki Taek','620000'),

('4','31','Secundária','Dong Ik','400000'),

('4','32','Principal','Yeon Kyo','550000'),

('4','33','Secundária', 'Ki Woo','360000'),

('5','1','Principal','Vanessa','5000000'),

('5','37','Secundária','Lea', '200000'),

('5','38','Principal','Roland','2000000'),

('6','39','Principal','Maximus','600000'),

('6','40','Secundária','Commodus','400000'),

('7','11','Principal','Cobb','30000000'),

('7','49','Secundária','Mal','300000'),

('7','50','Secundária','Ariadne','500000'),

('8','51','Principal','Joy','100000'),

('8','52','Secundária','Sadness','50000'),

('8','53','Principal','Riley','20000'),

('9','49','Principal','Edith Piaf','800000'),

('9','57','Secundária','Titine','60000'),

('9','58','Secundária','Louis Leplée','50000');