Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Unidade Curricular de

Bases de Dados

Ano Lectivo de 2015/2016

Biblioteca de Filmes

André Silva (A70377), Bruno Carvalho (A67847),

João Fontes (A71184), Pedro Fortes (A64309)

Novembro 2015

Data de Recepção	
Responsável	
Avaliação	
Observações	

Biblioteca de Filmes

André Silva (A70377), Bruno Carvalho (A67847),
João Fontes (A71184), Pedro Fortes (A64309)

Novembro 2015

Resumo

O projeto apresentado contextualiza – se na unidade curricular de Bases de Dados e tenta superar os objetivos apresentados nela.

Todas as temáticas lecionadas e apresentadas no programa pelos docentes são de alguma forma implementadas aqui desde a base analítica de requisitos, modulação conceptual, lógica e transacional do problema até um futuro desenvolvimento físico de uma plataforma.

A proposta transparente aqui surge no âmbito da implementação de um sistema de gestão de filmes denominado Biblioteca de Filmes.

As várias fases e secções desde documento apresentam o que foi descrito anteriormente de forma detalhada para resolução e apresentação do problema idealizado pelo grupo de projeto.

No final deste excerto terminamos com uma análise crítica fundamentada com pontos relativos a possíveis adições/extensões da plataforma e algumas preocupações/dificuldades encontradas no desenrolar do trabalho.

Por fim, incluímos ainda algumas tabulações e anexos que nos permitem demonstrar numa forma mais clara e percetível o processo necessário para a elaboração de uma proposta desta natureza.

Área de Aplicação: Desenho e arquitetura de sistemas de base de dados

Palavras-Chave: Base de Dados, Modelação, Especificação, Chen, Entidade, Requisito, Relacional, Conceptual, Atributo, Chave.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

1.1. Contextualização

Para este projeto, inserido na disciplina de Bases de Dados do Mestrado Integrado em Engenharia Informática, foi apresentada alguma liberdade na temática do trabalho por parte da equipa docente. Como tal, foi sugerido pelo nosso grupo a implementação de uma plataforma que permitisse armazenar filmes, ou seja, o que para nós surge como uma "Biblioteca de Filmes".

1.2. Apresentação do Caso de Estudo

Devido à explosão crescente na evolução das tecnologias e dados criados por elas surge a necessidade de gerir, organizar, sistematizar e ordenar este enorme conteúdo de informação. A nosso ver, a melhor forma de o fazer será recorrendo à informatização de serviços dantes considerados rigorosamente manuais, baseados no trabalho da pessoa e no armazenamento em arquivo físico. Claramente esta antiga linha de pensamento apresenta as suas desvantagens, entre as quais, demasiadas horas de trabalho e utilização de um espaço real tanto maior quanto aquilo que armazenarmos, o que introduz custos muito elevados.

Com a introdução de uma plataforma eletrónica poderemos organizar toda esta informação de forma mais simples e com menor horas de trabalho no que toca à inserção dos dados na BD. Também não nos teremos que preocupar com os custos adjacentes ao espaço utilizado por estes elementos, visto que certamente será inferior e utilizará menos recursos materiais que o antigo conceito físico.

Mas não são estas as principais vantagens visto que, a maior preocupação e a ideia cerne desta implementação é a análise dos dados. Ou seja, o grande interesse aqui surge no facto de esta permitir realizar perguntas (*Queries*) a partir dos dados existentes.

Na seguinte demonstração, nascida puramente em discussão, representamos um pouco daquilo que os sistemas de BD permitem. Uma pequena biblioteca que permite acumular utilizadores, filmes, personagens, atores e realizadores que serão futuramente relacionados e organizados consoante o levantamento de requisitos apresentados na secção 2 deste documento.

1.3. Motivação e Objetivos

Vivemos num mundo afogado em sistemas digitais e como tal faz todo o sentido implementar novas formas de organização e gestão no que toca a tecnologias outrora analógicas (Ex: Cassetes, filmes de película).

Para se juntar a esta situação surge nas pessoas a necessidade de poderem ter acesso a informação e poderem partilhar as suas opiniões onde e quando assim desejarem sem terem que carregar com o peso destes objetos que são agora considerados raridades e também elementos de coleção.

Aqui entra a nossa idealização de uma plataforma (Biblioteca de Filmes) onde os seus utilizadores poderão gerir uma lista de filmes, consultar a sua informação e revelar opiniões através de um sistema de classificações e de comentário. De notar que todos estes dados serão inseridos na plataforma por um administrador de sistema.

1.4. Estrutura do Relatório

No presente relatório iremos incidir nas temáticas realizadas na unidade curricular de Bases de Dados desde a apresentação dos requisitos (Destinados aos objetivos com que a plataforma de filmes deve coincidir), modelo conceptual (Identificação das entidades e relações existentes entre si), transações, modelo lógico e físico (Implementação de uma BD na sua forma recorrendo à linguagem do *SQL* e criação de queries para resolução de problemas colocados), tudo isto tendo em conta o projeto em desenvolvimento da biblioteca de filmes.

2. Levantamento e Análise de Requisitos

Neste ponto recorremos a uma pequena reunião de grupo para obtermos os requisitos necessários para a implementação da biblioteca de filmes a partir da opinião de todos e das suas experiências com este tipo de plataformas já existentes no mercado.

1 - A biblioteca é constituída por vários filmes, sendo que cada um recebe uma descrição, um título, uma capa, uma duração (Horas, minutos e segundos), o seu ano de lançamento, um ou mais géneros (Cada um definido por um nome e descrição dentro da própria biblioteca) e uma classificação geral, que é calculada a partir de uma média parcial das classificações recebidas pelos utilizadores.

2 - A avaliação recebida pelo filme por parte dos utilizadores consiste numa nota com valor entre 0 e 10 (Classificação) e comentários deixados por estes.

3 - Um filme pode não receber avaliação (Ex: ter acabado de sair e ainda ninguém o ter visualizado), tal como um utilizador pode escolher não avaliar nenhum filme.

4 - O utilizador que usa a plataforma efetua um registo numa dada data no qual indica o seu nome, introduz uma foto de perfil, os seus contactos (Telefone, *facebook* e correio eletrónico) e ainda a palavra-chave para mais tarde efetuar *login* juntamente com o correio eletrónico referido.

5 - Após visualizar um filme, o utilizador tem a possibilidade de o adicionar à sua lista de favoritos.

6 - Existe a possibilidade de o utilizador não ver nenhum filme ou de não ter filmes favoritos tal como é possível existirem filmes na biblioteca que não tenham sido visualizados ou adicionados aos favoritos por ninguém.

7 - Para cada filme guarda-se a informação dos atores, personagens e realizadores.

8 - Tanto para o ator como para o realizador guarda-se a informação relativa a nome, data de nascimento e a sua biografia.

9 – Na biografia indicamos um resumo pessoal e uma lista de acontecimentos definidos por uma data e uma descrição.

10 - É possível existirem atores que tenham entrado no mesmo filme, como também é possível que vários realizadores tenham realizado o mesmo filme, o que leva ao aparecimento do mesmo registo/acontecimento em que se entrou/realizou o filme nas suas biografias.

11 – Cada ator protagoniza pelo menos uma ou mais personagens nos filmes em que entra. De apontar que existe a possibilidade de um mesmo ator realizar o papel de várias personagens num único filme.

12 - Existe a possibilidade de existir filmes sem atores (Como por exemplo filmes de animação onde só existem personagens com as vozes dos atores).

13 - Pode existir filmes que não tenham personagens nem atores (Filmes onde não surja nenhum diálogo e apenas surjam imagens e pequenas filmagens artísticas incluindo ambientes paisagísticos).

14 – É impossível existirem filmes sem pelo menos um realizador.

15 - É impossível uma biografia ter uma lista de registo de acontecimentos vazia, pois uma pessoa não é ator/realizador sem ter entrado/realizado pelo menos um filme.

3. Caracterização dos Perfis de Utilização

Com este projeto pretendemos de forma explícita e intuitiva disponibilizar uma plataforma para os perfis de utilização em questão.

3.1. Administrador

Este elemento terá a responsabilidade de gerir todos os dados existentes na biblioteca de filmes recorrendo a um SGBD, como por exemplo a ferramenta MySQL.

Terá a preocupação de criar, preencher, atualizar e até mesmo eliminar todo o conteúdo vivo da BD.

Refere – se assim basicamente à lista de utilizadores, filmes, géneros, atores, personagens, realizadores e todas as relações existentes entre estes que serão clarificadas de forma mais explícita na secção 4 desde documento (Ex: Classificar, gostar, visualizar …).

3.2. Utilizador

Principal elemento e centro objetivo do desenvolvimento deste sistema/ferramenta. O utilizador aqui referido terá, idealmente, a possibilidade de consultar uma lista de filmes da nossa BD, tendo acesso à informação de cada um (Titulo, capa, duração, data de lançamento, descrição, géneros e classificação) e todas as outras entidades relacionadas, tal como as personagens (Nome), atores e realizadores (Nome, data de nascimento e biografia).

Adicionalmente o utilizador poderá ainda interagir com estes recursos, desde a simples leitura/processamento da informação até à possibilidade de classificar e comentar um dado filme (Avaliação), assinalar como visualizado ou até mesmo adicioná – lo à sua lista de favoritos.

Por fim, na visão de um futuro desenvolvimento, introduzimos na plataforma um perfil para cada utilizador (Data de registo, nome, palavra-chave, foto e contactos (Telefone, correio eletrónico e *facebook*)), onde cada um utilizará o seu correio eletrónico e palava – passe para realizar *login* no sistema.

Neste protótipo o perfil pode apenas ser visualizado pelo próprio.

4. Modelo Conceptual

Nesta primeira fase do projeto introduzimos a primeira temática apresentada na unidade curricular de Bases de Dados que entra em conformidade com a bibliografia referida na secção final deste documento, desde a averiguação de entidades, atributos, chaves e relacionamentos que iriam compor o modelo.

Para a elaboração desde mesmo considerámos a utilização da ferramenta yEd (Editor gráfico) e baseámo – nos inteiramente na lista de requisitos apresentada na secção 2 deste documento.

4.1. Identificação de Tipos de Entidades

Nesta secção define-se todas as entidades existentes no modelo conceptual, o seu significado e outros termos pelas quais poderão ser identificadas (*Aliases*).

Nome da Entidade	Descrição	Aliases	Ocorrência
Filme	Termo geral que descreve todos os filmes presentes na BD	Documento, Obra Cinematográfica	Um filme é uma obra cinematográfica realizada por um realizador onde existem personagens que são representadas por atores. Um filme passa a existir no sistema no momento que é lançado nos cinemas. Posteriormente ao lançamento do filme, este poderá ser avaliado por utilizadores, que podem atribuir uma classificação ao filme, baseado na sua satisfação com o mesmo. O conjunto

			de classificações atribuídas será usado no cálculo da nota geral do mesmo pela média do conjunto das classificações referidas. Estes filmes contêm pelo menos um género que o agrupam numa dada categoria consoante o conteúdo que apresentam.
Genero	Termo geral que descreve os géneros possíveis que um filme pode ter	Categoria, Tipo	Um género é um agrupamento de filmes da mesma categoria para efeitos de caracterização de conteúdo. Todos os filmes têm pelo menos um ou mais géneros baseados no conteúdo que reproduzem, podendo combinar os diversos géneros disponíveis.
Ator	Termo geral que descreve os atores presentes na BD	Interveniente	O ator é uma pessoa que protagoniza o papel de uma personagem num determinado filme. No caso de o filme ser de animação, o ator apenas dará voz à personagem, não surgindo diretamente no filme. Todos os atores presentes no sistema entraram em pelo menos um filme, não sendo permitido no sistema atores sem filmes protagonizados. Os filmes protagonizados pelo

			ator vão entrar para a sua biografia, podendo ser consultados posteriormente acedendo a esta mesma.
Realizador	Termo geral que descreve todos os realizadores presentes no sistema	Diretor, Chefe de Realização	O realizador é a pessoa que dirige a realização do filme, ou seja, que coordena todas as etapas de filmagem desde a gestão dos atores, escolha de locais de filmagem e rodagem do mesmo. Todos os filmes têm pelo menos um realizador, sem o qual não haveria coordenação das atividades de gravação do referido. Os realizadores, à semelhança dos atores, também têm uma biografia com os registos dos filmes que realizaram para uma eventual consulta.
Utilizador	Termo geral que descreve todos os utilizadores presentes no sistema	Avaliador	O utilizador é a pessoa que visualiza um filme e que, posteriormente à sua visualização, pode avaliar o filme através da atribuição de uma classificação, que se vai refletir na classificação final do filme, que vai constatar na média das classificações atribuídas pelo conjunto de utilizadores que avaliou o filme

			referido. O utilizador, após visualizar filmes, pode adicionar a uma lista de favoritos, que consiste nos filmes que o utilizador mais gostou. De realçar que esta lista pode estar vazia como o utilizador pode não avaliar nenhum filme, usando apenas a BD para consultar informações acerca do filme.
Personagem	Termo geral que descreve todas as personagens que entram num filme	Papel	Uma personagem é um papel representado por um ator num filme. No caso de um filme de animação, estas personagens são fictícias, sendo que apenas são protagonizadas por alguém que lhes dá voz. Um filme obrigatoriamente tem uma personagem que entra no seu conteúdo. Uma personagem será protagonizada por um e um só ator, não podendo a mesma personagem no mesmo filme ser protagonizada por 2 ou mais atores.

Tabela 1 - Representação das entidades

4.2. Identificação de Tipos de Relacionamentos

Na seguinte tabela desta secção para cada entidade da primeira coluna referenciamos quais os relacionamentos e multiplicidade existentes com as entidades da última coluna.

Nome da Entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da Entidade
Filme	1..m	Tem	1..n	Genero
	1..m	Tem	1..n	Personagem
	1..n	É realizado por	1..m	Realizador
	0..n	É visualizado por	0..m	Utilizador
	0..n	É gostado por	0..m	Utilizador
	0..n	É avaliado por	0..m	Utilizador
Genero	1..m	Possui	1..n	Filme
Personagem	0..n	É protagonizada por	1..m	Ator
	1..n	Entra	1..m	Filme
Ator	1..m	Protagoniza	1..n	Personagem
Realizador	1..m	Realiza	1..n	Filme
Utilizador	0..m	Visualiza	0..n	Filme
	0..m	Gosta	0..n	Filme
	0..m	Avalia	0..n	Filme

Tabela 2 - Representação dos relacionamentos

4.3. Descricao, Domínios e Associação de Atributos com Tipo de Entidades ou Relacionamentos

Entidade	Atributos	Descrição	Tipo e tamanho	Chave Primária	Chave Estrangeira	Nulo	Multivalor	Derivado
Filme	id_filme	Identifica um determinado filme	Inteiro Positivo	Sim	Não	Não	Não	Não
	Titulo	Nome do filme	String tamanho variável com comprimento de 50 caracteres	Não	Não	Não	Não	Não
	Ano_lancamento	Ano de lançamento do filme	Inteiro Positivo	Não	Não	Não	Não	Não
	Duracao	Duração do Filme em minutos	Inteiro Positivo	Não	Não	Não	Não	Não
	Descricao	Pequeno texto sobre o filme	String tamanho variável com comprimento de 250 caracteres	Não	Não	Sim	Não	Não

	Capa	Imagem de representação do filme	Imagem	Não	Não	Sim	Não	Não
	Rating_Geral	Nota de 0 a 10 atribuída ao filme	Float de 0 a 10	Não	Não	Não	Não	Sim
Genero	Nome	Nome do género	String tamanho variável com comprimento de 50 caracteres	Sim	Não	Não	Não	Não
	Descricao	Pequeno texto sobre o género	String tamanho variável com comprimento de 250 caracteres	Não	Não	Sim	Não	Não
Ator	id_ator	Identifica um determinado ator	Inteiro Positivo	Sim	Não	Não	Não	Não
	Nome	Nome do ator	String tamanho variável com comprimento de 50 caracteres	Não	Não	Não	Não	Não
	Data_Nascimento	Data de Nascimento do ator	Data	Não	Não	Não	Não	Não
	Biografia	Contém uma descrição e uma lista de datas referentes a ocasiões em que o ator entrou num filme		Não	Não	Não	Sim	Não
	Descricao	Texto sobre um determinado acontecimento da vida do ator. Este atributo faz parte do atributo multivalor Biografia	String tamanho variável com comprimento de 250 caracteres	Não	Não	Sim	Não	Não
	Data	Data de registo da descrição associada. Este atributo faz parte do atributo multivalor Biografia	Data	Não	Não	Não	Não	Não
Personagem	Id_personagem	Identifica uma determinada personagem	Inteiro Positivo	Sim	Não	Não	Não	Não
	Nome	Nome da personagem	String tamanho variável com comprimento de 50 caracteres	Não	Não	Não	Não	Não
	Descricao	Pequeno Texto sobre a Personagem	String tamanho variável com comprimento de 250 caracteres	Não	Não	Sim	Não	Não
Realizador	id_realizador	Identifica um determinado realizador	Inteiro Positivo	Sim	Não	Não	Não	Não
	Nome	Nome do Realizador	String tamanho variável com	Não	Não	Não	Não	Não

			comprimento de 50 caracteres					
	Data_Nascimento	Data de Nacimento do realizador	Data	Não	Não	Não	Não	Não
	Biografia	Contém uma descrição e uma lista de datas referentes a ocasiões em que o realizador realizou um filme		Não	Não	Não	Sim	Não
	Descricao	Texto sobre um determinado acontecimento da vida do realizador. Este atributo faz parte do atributo multivalor Biografia	String tamanho variável com comprimento de 250 caracteres	Não	Não	Sim	Não	Não
	Data	Data de registo da descrição associada. Este atributo faz parte do atributo multivalor Biografia	Data	Não	Não	Não	Não	Não
Utilizador	Nome	Nome do Utilizador	String tamanho variável com comprimento de 50 caracteres	Não	Não	Não	Não	Não
	Password	Texto alfanumérico encriptado para acesso à conta do utilizador	String tamanho variável com comprimento de 50 caracteres encriptado	Não	Não	Não	Não	Não
	Telefone	Número de contacto telefónico do utilizador	Inteiro Positivo	Não	Não	Sim	Não	Não
	Email	Contacto de Correio Eletrónico que também é o identificador do utilizador	String tamanho variável com comprimento de 50 caracteres	Sim	Não	Não	Não	Não
	Facebook	Nome ou username do utilizador no Facebook	String tamanho variável com comprimento de 50 caracteres	Não	Não	Sim	Não	Não
	Foto	Imagem de identificação do utilizador	Imagem	Não	Não	Sim	Não	Não
	Data_Registo	Data em que o utilizador foi registado na base de dados	Data	Não	Não	Não	Não	Não
Relação Filme Utilizador	Rating_User	Nota que um determinado utilizador atribuiu a um determinado filme	Float de 0 a 10	Não	Não	Não	Não	Não
	Comentario	Pequeno texto que um determinado utilizador critica sobre um	String tamanho variável com comprimento de 250	Não	Não	Sim	Não	Não

		determinado filme	caracteres					

Tabela 3 – Dicionário de dados

4.4. Identificação de Chaves Candidatas e Primárias

4.4.1 Filme

Chaves candidatas: id_Filme, Titulo

No caso da entidade filme, temos como chaves candidatas os atributos id_filme e Titulo. Como o título do filme não identifica unicamente o filme pois existe a possibilidade de existir dois filmes com títulos iguais, este atributo não poderia ser considerado a chave primária.

Discutimos na possibilidade de criar uma chave composta que incluísse os atributos Titulo e Ano_Lancamento, mas como nada nos garantia que não poderiam existir filmes com o mesmo nome e que tivessem sido lançados no mesmo dia, decidimos criar um identificador único para cada filme (id_Filme), este que servirá como chave primária da entidade Filme pois garante - nos a unicidade de cada registo de filme.

4.4.2 Genero

Chaves candidatas: Nome

No caso do género temos como chave candidata o atributo Nome. Podemos garantir que este nome vai ser único, pois não é possível existirem dois tipos de género com nomes iguais sem que acabem por representar o mesmo. Decidimos assim adotar o atributo Nome para chave primária da entidade Genero.

4.4.3 Ator

Chaves candidatas: id_Ator, Nome

No caso da entidade Ator temos como chaves candidatas os atributos id_ator e Nome. No caso do atributo Nome não podemos garantir a unicidade de cada nome pois duas pessoas podem ter nomes iguais. Concluindo, o atributo nome não pode ser adotado como chave primária da entidade Ator.

Ponderámos em usar uma chave composta que combinasse os atributos Nome e Data Nascimento, mas como nada nos podia garantir que dois atores não nascessem no mesmo dia e tivessem o mesmo nome, decidimos criar um identificador único para cada ator (id_Ator), que seria utilizado como chave primária da entidade Ator.

4.4.4 Personagem

Chaves candidatas: id_Personagem, Nome

Para a entidade Personagem considerámos como chaves candidatas os atributos id_personagem e Nome. Para o caso do atributo Nome como não podemos garantir a inexistência de duas personagens com o mesmo nome, decidimos criar um identificador único para cada personagem que, no final, viria a ser a chave primária desta entidade.

4.4.5 Realizador

Chaves candidatas: id_Realizador, Nome

No caso da entidade Realizador tomámos como chaves candidatas os atributos id_realizador e Nome. No caso do atributo Nome como não podemos garantir a unicidade de cada registo, pois podem existir realizadores com o mesmo nome, decidimos não adotar o atributo Nome como chave primária da entidade Realizador.

Considerámos a adoção de uma chave composta pelos atributos Nome e Data Nascimento mas, como desta forma não poderíamos garantir a unicidade de cada registo de realizador, pois não seria possível garantir a inexistência de realizadores que tivessem nascido no mesmo dia e tivessem o mesmo nome, decidimos criar um identificador único para cada registo acabou por ser vista como a chave primária da entidade Realizador.

4.4.6 Utilizador

Chaves candidatas: Nome, Telefone, Email, Facebook

No caso da entidade Utilizador considerámos como chaves candidatas os atributos Nome, Telefone, Email e Facebook. Como nada nos garante a inexistência de utilizadores com o mesmo nome decidimos não utilizar o atributo Nome como chave primária da entidade Utilizador.

No caso do atributo Telefone, também nada nos garante que não surjam utilizadores com o mesmo telefone, logo esta também não poderá ser uma chave primária.

No caso dos atributos Email e Facebook, qualquer um destes atributos identifica unicamente cada utilizador. Optamos assim pela utilização do atributo Email para chave primária da entidade Utilizador, não mostrando obrigação ao utilizador de criar uma conta do Facebook para aceder à nossa plataforma.

4.5. Diagrama de Chen

Esta secção referente à modulação conceptual encontra – se representada por um diagrama na secção dos Anexos (Anexo 1).

4.6. Verificação do Modelo por Redundância

Neste passo examinamos o modelo conceptual com o propósito de identificar e remover qualquer tipo de redundância de informação com vista a uma simplificação do modelo. Isto é, procuramos casos em que, partindo de entidades ou relacionamentos diferentes, conseguimos obter a mesma informação. O que procurar?

4.6.1 Relações 1:1

Caso este tipo de relações exista, verificar se as entidades envolvidas na relação não são apenas sinónimos, pois se isso acontecer, teremos que fundir as entidades numa só escolhendo uma chave primária para a nova entidade e guardar a outra como chave alternativa.

4.6.2 Remover relações redundantes

Verificar a existência de vários caminhos entre as mesmas entidades (Origem e destino), o que pode levar a que estes relacionamentos conduzam o utilizador para a mesma informação variando apenas no acesso a esta.

No modelo conceptual apresentado não existe qualquer tipo de relacionamento do tipo 1:1, sendo que o caso 4.6.1 está fora da nossa margem de análise.

Relativamente ao ponto 4.6.2, analisámos o relacionamento existente entre a entidade Filme e Utilizador como mostra a figura seguinte.

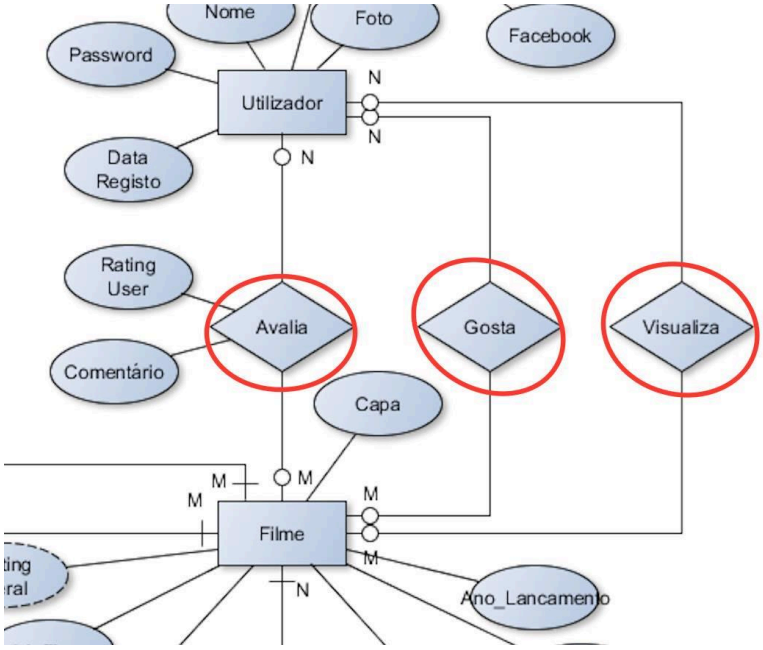


Figura 1 – Relação entre Filme e Utilizador

Este tipo de relação pode apresentar redundância, o que não é o caso. Embora as entidades partilhem três relacionamentos diferentes entre elas são todas do tipo N:M, dando origem a tabelas diferentes que correspondem a dados únicos e necessários, sendo que em nenhum caso, a informação contida em qualquer uma destas relações pode derivar o que está contido nas restantes. Concluímos assim que nenhuma destas relações deveria ser alterada.

Não identificámos mais nenhum caso em que, partindo de uma dada origem, conseguíssemos chegar ao mesmo destino por mais que um caminho distinto.

4.7. Validação do Modelo com Possível Transações de Utilizadores

Nesta secção analisamos a correspondência existente entre o modelo e as solicitações realizadas pelo cliente (Neste caso o próprio grupo de projeto) e verifica-se a resposta deste face a algumas transações. O objetivo é verificar se as transações pedidas ao modelo são passíveis de serem realizadas. Optámos por fazer esta análise de forma puramente textual.

4.7.1 Descrição de transações

Um ator protagoniza uma ou mais personagens, ou seja, cumprem entre si um relacionamento do tipo N:M. Os dados estão contidos na própria relação, sendo que uma personagem entra obrigatoriamente num ou mais filmes e a sua informação fica guardada aqui.

Em suma, uma personagem tem uma relação com um ou mais filmes assim como o ator tem com uma ou mais personagens.

Um filme é realizado por um ou mais realizadores. O realizador condiciona a entidade filme e os dados referentes a esta transação ficam guardados no relacionamento entre estas duas entidades.

Um filme tem também um ou mais géneros que são caracterizados pela sua descrição. Mais uma vez, encontramos uma certa influência entre entidades que fica armazenada na relação que mantêm.

O utilizador pode relacionar - se com o filme de três formas distintas. Uma entidade deste tipo pode visualizar um ou mais filmes, sendo que esta simples transação modifica o estado da relação, e pode ainda gostar ou avaliar um ou mais filmes.

Estas duas transações alteram também o estado das respetivas relações podendo assim dizer que existem três relações possíveis entre estas duas entidades e que todas elas são necessárias de forma a respeitar as transações requeridas pelo utilizador.

5. Modelo Lógico

5.1. Ilustração Modelo Lógico

Para esta etapa do processo de construção de uma base de dados apresentamos o respetivo modelo lógico, que se encontra em anexo neste documento. Para a representação deste modelo recorremos à utilização da ferramenta do MySQL Workbench.

Na tradução do modelo conceptual para o lógico deixamos de nos referir a entidades e relacionamentos passando a apresentar o mesmo conceito através de tabelas.

Nas subsecções que se seguem será tudo explicado de forma mais detalhada, incluindo todas as decisões tomadas na transformação do projeto aqui retratado desde o tratamento de entidades fortes ou fracas, relacionamentos de um para um, um para muitos, muitos para muitos, atributos multivalor, etc.

5.2. Descrição de Cada Relacionamento

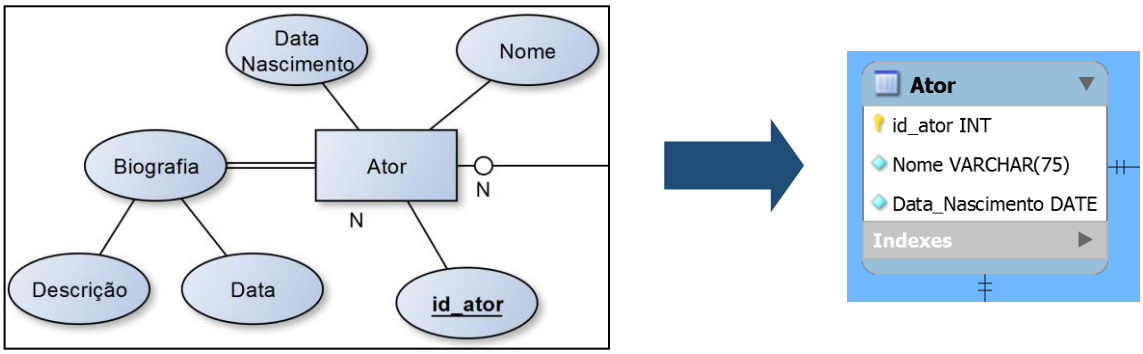


Figura 2 – Conversão da entidade Ator

Ator = {id_ator, Nome, Data_Nascimento}
Chave(s) Primária(s): id_ator

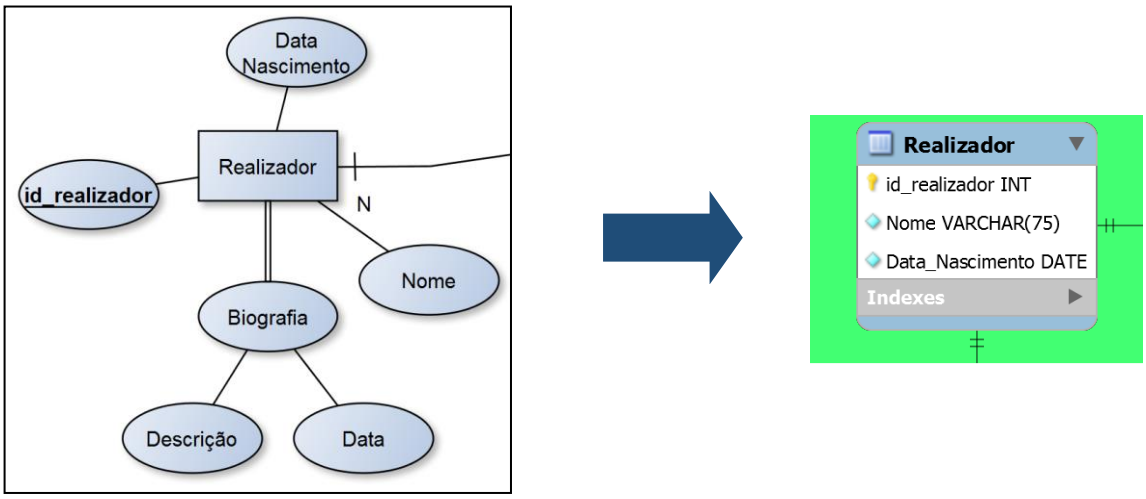


Figura 3 – Conversão da entidade Realizador

Realizador = {id_realizador, Nome, Data_Nascimento}

Chave(s) Primária(s): id_realizador

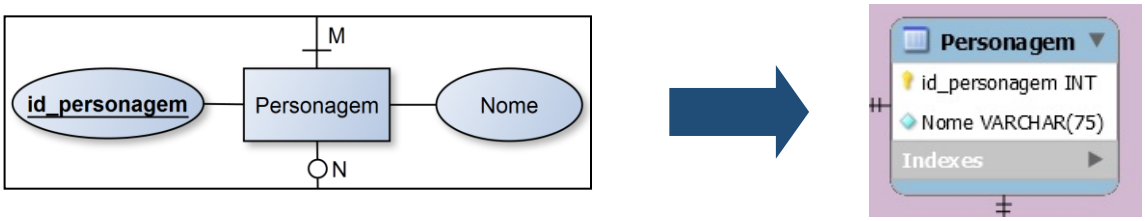


Figura 4 – Conversão da entidade Personagem

Personagem = {id_personagem, Nome}

Chave(s) Primária(s): id_personagem

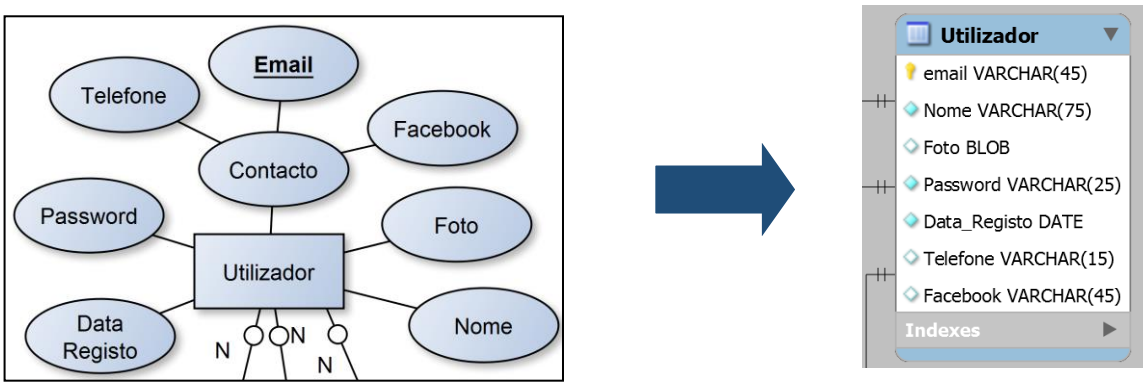


Figura 5 – Conversão da entidade Utilizador

Utilizador = {email, Nome, Foto, Password, Data_Registo, Telefone, Facebook}

Chave(s) Primária(s): email

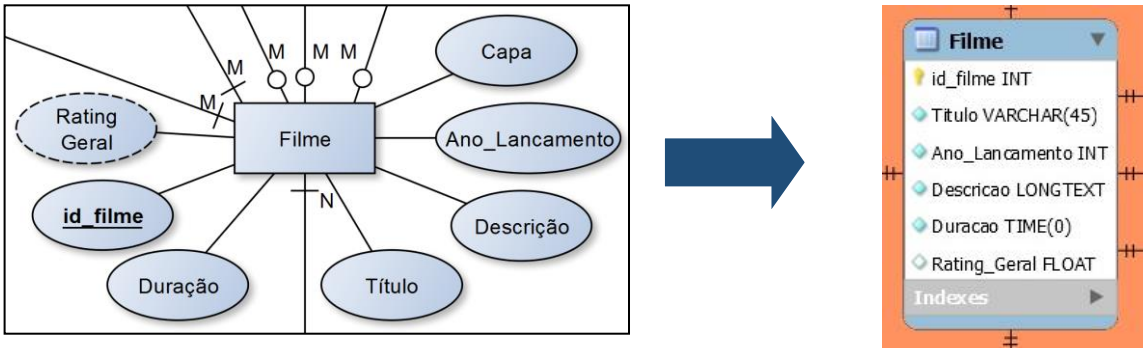


Figura 6 – Conversão da entidade Filme

Filme = {id_filme, Titulo, Ano_Lancamento, Descricao, Duracao, Rating_Geral}
Chave(s) Primária(s): id_filme
Atributo(s) Derivado(s): Rating_Geral (**Avg (Rating_User)** - Referência a **Utilizador_Avalia_Filme**)

5.2.1 Relacionamentos de 1 para 1 (1:1)

No projeto em questão não se encontram presentes relacionamentos que apresentem uma definição de um para um.

5.2.2 Relacionamentos de 1 para Muitos (1:n)

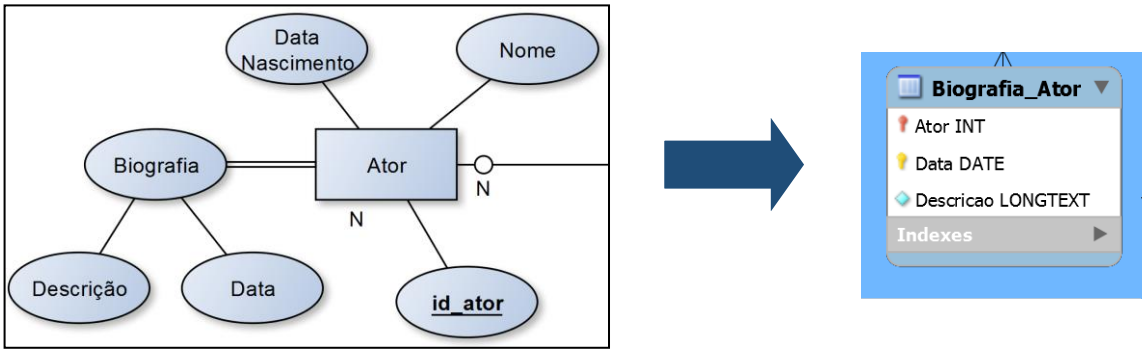


Figura 7 – Conversão do atributo multivalor Biografia de um Ator

Biografia_Ator = {Ator, Data, Descricao}
Chave(s) Primária(s): Ator, Data
Chave(s) Estrangeira(s): Ator - Referência a **Ator (id_ator)**

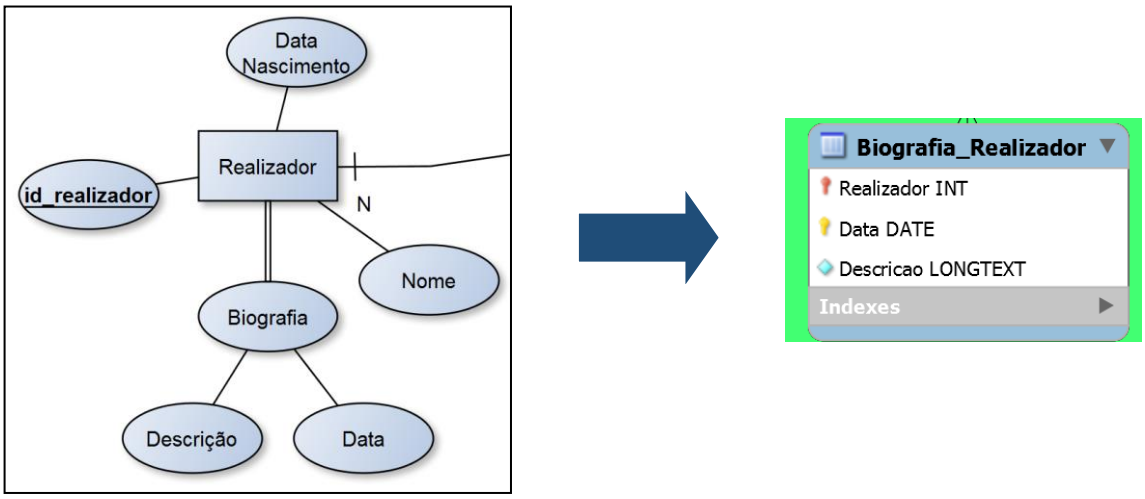


Figura 8 – Conversão do atributo multivalor Biografia de um Realizador

Biografia_Realizador = {Realizador, Data, Descricao}

Chave(s) Primária(s): Realizador, Data

Chave(s) Estrangeira(s): Realizador - Referência a **Realizador (id_realizador)**

5.2.3 Relacionamentos de Muitos para Muitos (n:n)

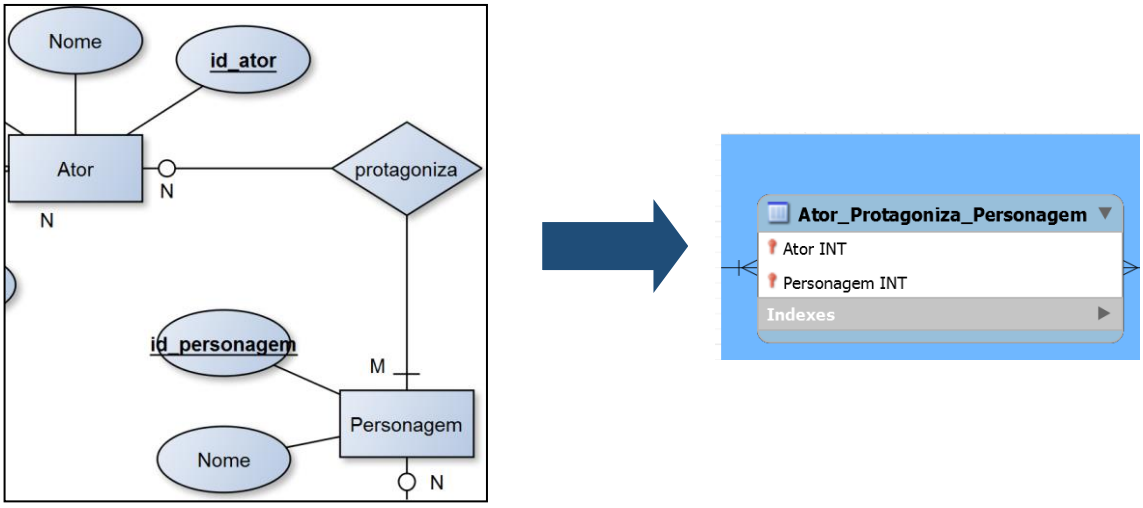


Figura 9 – Conversão da relação Ator protagoniza uma Personagem

Ator_Protagoniza_Personagem = {Ator, Personagem}

Chave(s) Primária(s): Ator, Personagem

Chave(s) Estrangeira(s): Ator - Referência a **Ator (id_ator)**

Personagem - Referência a **Personagem (id_personagem)**

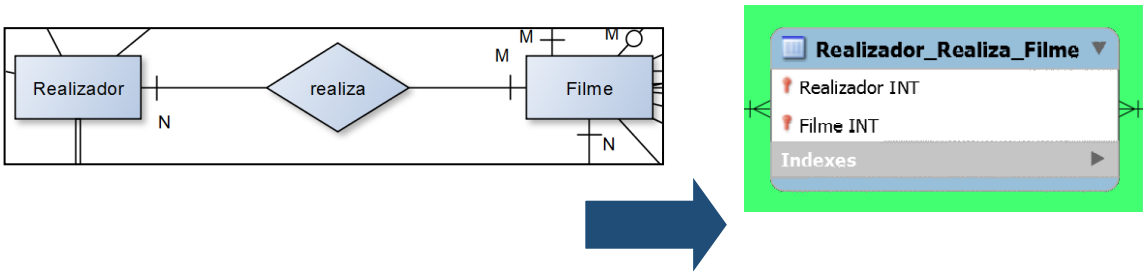


Figura 10 – Conversão da relação Realizador realiza Filme

Realizador_Realiza_Filme = {Realizador, Filme}

Chave(s) Primária(s): Realizador, Filme

Chave(s) Estrangeira(s): Realizador - Referência a **Realizador (id_realizador)**

Filme - Referência a **Filme (id_filme)**

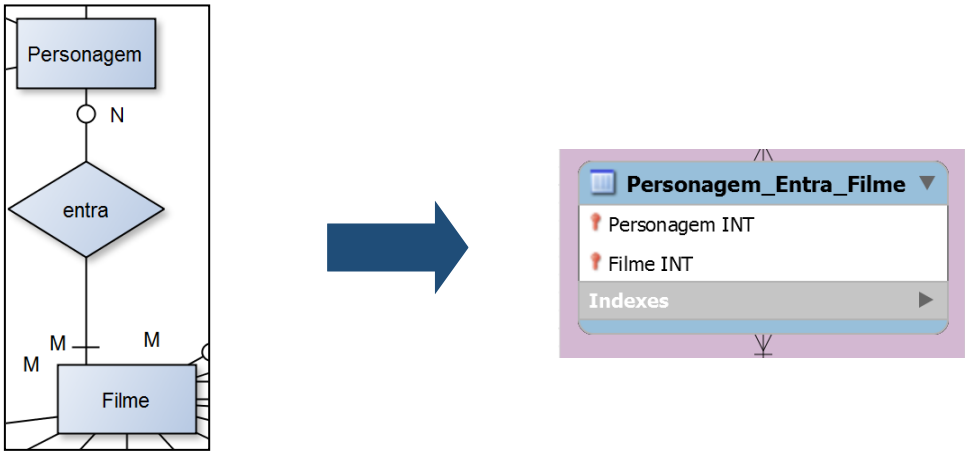


Figura 11 – Conversão da relação Personagem entra num Filme

Personagem_Entra_Filme = {Personagem, Filme}

Chave(s) Primária(s): Personagem, Filme

Chave(s) Estrangeira(s): Personagem - Referência a **Personagem (id_personagem)**

Filme - Referência a **Filme (id_filme)**

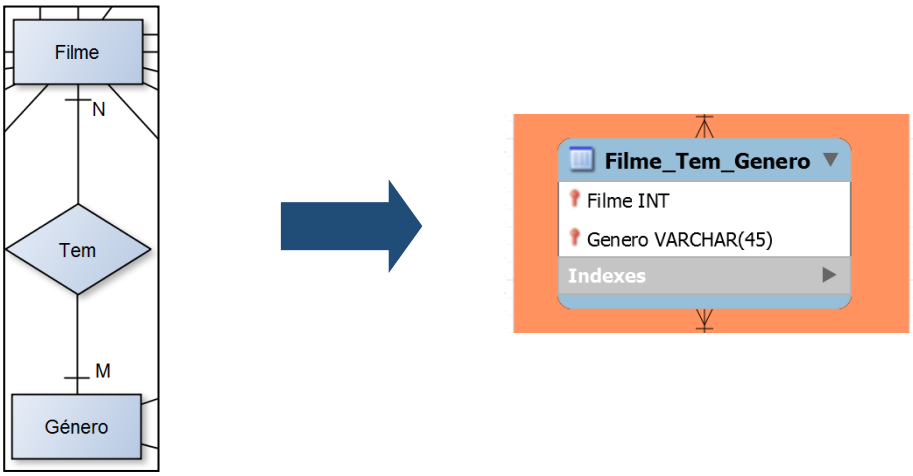


Figura 12 – Conversão da relação Filme tem Género

Filme_Tem_Genero = {Filme, Genero}

Chave(s) Primária(s): Filme, Genero

Chave(s) Estrangeira(s): Filme - Referência a **Filme (id_filme)**

Genero - Referência a **Genero (Nome)**

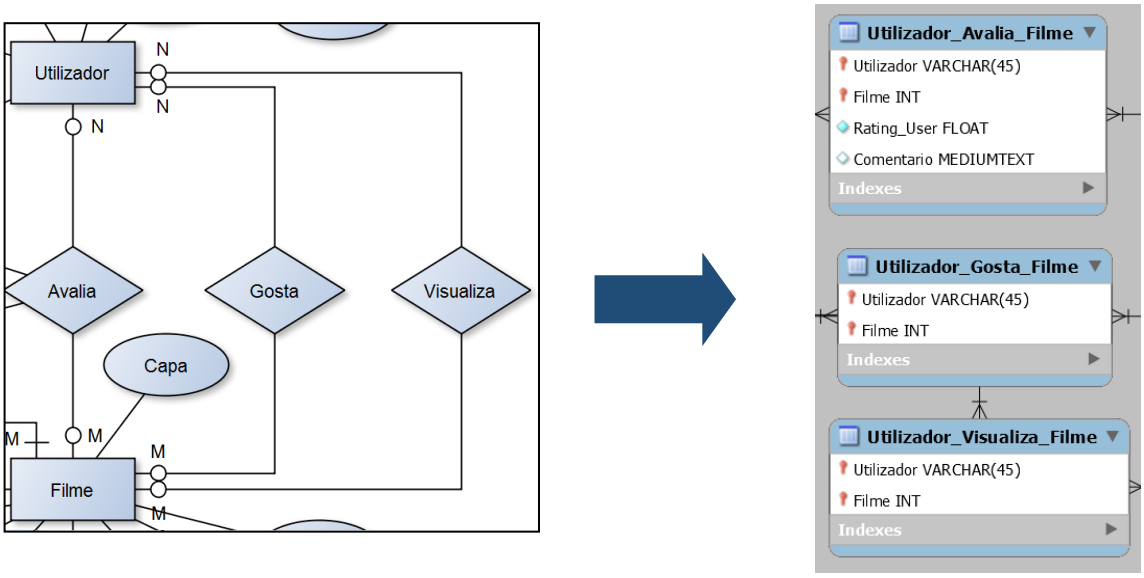


Figura 13 – Conversão das relações entre Utilizador e Filme

Utilizador_Avalia_Filme = {Utilizador, Filme, Rating_User, Comentario}

Chave(s) Primária(s): Utilizador, Filme

Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**

Filme - Referência a **Filme (id_filme)**

Utilizador_Gosta_Filme = {Utilizador, Filme}
Chave(s) Primária(s): Utilizador, Filme
Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**
Filme - Referência a **Filme (id_filme)**

Utilizador_Visualiza_Filme = {Utilizador, Filme}
Chave(s) Primária(s): Utilizador, Filme
Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**
Filme - Referência a **Filme (id_filme)**

5.3. Validação das Relações recorrendo à Teoria da Normalização

De forma a validar as relações impostas pelo modelo lógico no nosso projeto, foi necessário recorrer à normalização de forma a garantir que o nosso modelo não apresente problemas futuros na sua prática.

A utilização da normalização teve como principal objetivo identificar as relações com base nas suas chaves e dependências entre os atributos e identificar relações com redundância de dados, impedindo assim futuras anomalias na utilização e atualização da base de dados da nossa plataforma e garantir a integridade e flexibilidade dos dados.

Assim, aplicaremos a normalização até à Terceira Forma Normal validando o modelo lógico realizado para a nossa Biblioteca de Filmes.

5.3.1 Primeira Forma Normal (1FN)

Esta forma de normalização é a primeira a ser aplicada e tem como objetivo a identificação de possíveis elementos de informação repetidos nas tabelas, ou seja, se todos os valores dos correspondentes atributos são atómicos, não sendo possível a sua decomposição.

A estratégia de validação aplicada consiste em pesquisar entre tabelas informação que se encontre repetida e permitir um grau de facilitismo na identificação de uma chave-primária por tabela.

Aplicando este paradigma ao nosso modelo, verificamos que este já se encontra normalizado para a Primeira Forma Normal.

5.3.2 Segunda Forma Normal (2FN)

Nesta forma de normalização, tendo já a Primeira Forma Normal sido aplicada, teremos que verificar se todos os atributos que não são chaves candidatas estão dependentes de qualquer uma chave candidata, ou seja, não podemos permitir que existiam dependências parciais.

A estratégia de validação aplicada consiste principalmente em verificar as tabelas com uma chave primária composta, pois as tabelas com uma só chave primária encontram – se já na Segunda Forma Normal.

Sendo assim, através deste arquétipo partimos para a análise de cada atributo chave e verificamos a sua importância na tabela, identificando também os atributos que não são funcionalmente dependentes da chave-primária da tabela.

Aplicando-a ao nosso modelo, verificamos que este já se encontra normalizado para a Segunda Forma Normal.

5.3.3 Terceira Forma Normal (3FN)

Nesta forma de normalização, após aplicação da Segunda Forma Normal, também teremos que verificar se todos os atributos que não são chaves candidatas serão alcançadas entre relações apenas de uma forma, ou seja, verificar as dependências transitivas.

A estratégia aplicada consiste em identificar os atributos que são funcionalmente dependentes de outros atributos que não são chaves – estes não poderão ser aceites na tabela.

Aplicando-a ao nosso modelo verificamos que este já se encontra normalizado também para a Terceira Forma Normal.

Após este processo de normalização, concluímos que o modelo lógico realizado encontra-se normalizado (até à Terceira Forma Normal), o que implica a inexistência de qualquer tipo de redundância de informação ou inconsistência na base de dados que coloque em perigo o seu funcionamento como sistema fidedigno e robusto.

5.4. Validação das Relações a partir das Transações

Partimos agora para a validação das transações no modelo Lógico que aqui pretendemos reportar, ou seja, averiguar se os requisitos impostos na Secção 2 deste trabalho se encontram aqui satisfeitos por este modelo.

De seguida apresentamos e analisamos a transação mais significativa/relevante da nossa plataforma - a inserção de um filme no sistema. Desta forma confirmamos as relações regidas pelo modelo Lógico.

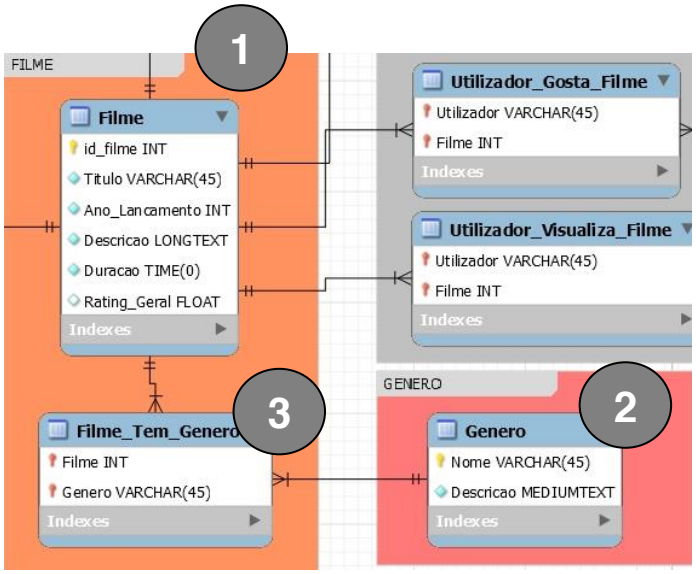


Figura 14 – Inserção de um Filme e correspondentes Géneros

Figura 15 – Inserção das Personagens e correspondentes Atores

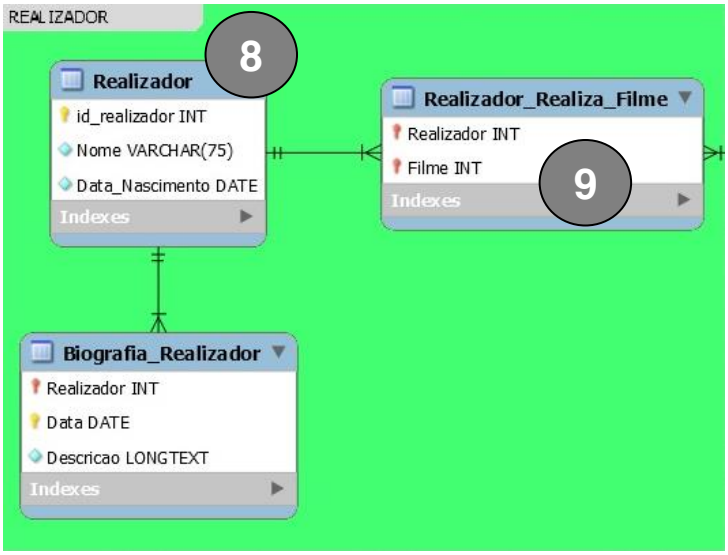


Figura 16 – Inserção dos Realizadores

Na inserção de um novo Filme na nossa biblioteca de filmes encontramos necessidade de, após inserir os dados relativos ao título, ano de lançamento, descrição e duração (1), proceder à introdução dos géneros relativos ao filme (2) e respetivas relações entre estes (3).

Esta complexa e necessária transação não termina por aqui e continua com a inserção das personagens (4), atores (5), relações entre estas duas entidades (6) e relação das referidas personagens com o filme que se pretende inserir no sistema (7). Terminamos assim adicionando os realizadores do filme (8) e devidas relações entre eles (9).

Todo este processo encontra – se numerado e descrito nas Figuras 14,15 e 16 desta secção.

Após esta curta análise através do esquema anteriormente apresentado (de notar que não utilizámos vistas e apenas utilizamos um processo de forma a organizar melhor a estrutura do modelo) verificamos que esta transação foi estabelecida sem quebrar os conceitos apresentados pelo modelo Conceptual.

Concluímos assim que o nosso modelo Lógico satisfaz os requisitos apresentados e que as relações encontram – se validadas.

5.5. Verificar Restrições de Integridade

Nesta secção tratamos de referir o modo como foram tratadas as operações de remoção e/ou atualização dos dados existentes na nossa BD e as restrições a que cada atributo das variadas tabelas estão sujeitos.

Ator = {id_ator, Nome, Data_Nascimento}
Chave(s) Primária(s): id_ator (**NOT NULL**, **AUTO_INCREMENT**)

Realizador = {id_realizador, Nome, Data_Nascimento}
Chave(s) Primária(s): id_realizador (**NOT NULL**, **AUTO_INCREMENT**)

Personagem = {id_personagem, Nome}
Chave(s) Primária(s): id_personagem (**NOT NULL**, **AUTO_INCREMENT**)

Filme = {id_filme, Titulo, Ano_Lancamento, Descricao, Duracao, Rating_Geral}
Chave(s) Primária(s): id_filme (**NOT NULL**, **AUTO_INCREMENT**)
Atributo(s) Derivado(s): Rating_Geral (**Avg (Rating_User)** - Referência a **Utilizador_Avalia_Filme**) (**NOT NULL**, **DEFAULT 0**)

Utilizador = {email, Nome, Foto, Password, Data_Registo, Telefone, Facebook}
Chave(s) Primária(s): email (**NOT NULL**)

Biografia_Ator = {Ator, Data, Descricao}
Chave(s) Primária(s): Ator (**NOT NULL**), Data (**NOT NULL**)
Chave(s) Estrangeira(s): Ator - Referência a **Ator (id_ator)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Biografia_Realizador = {Realizador, Data, Descricao}
Chave(s) Primária(s): Realizador (**NOT NULL**), Data (**NOT NULL**)
Chave(s) Estrangeira(s): Realizador - Referência a **Realizador (id_realizador)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Ator_Protagoniza_Personagem = {Ator, Personagem}
Chave(s) Primária(s): Ator (**NOT NULL**), Personagem (**NOT NULL**)
Chave(s) Estrangeira(s): Ator - Referência a **Ator (id_ator)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Personagem - Referência a **Personagem (id_personagem)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Realizador_Realiza_Filme = {Realizador, Filme}
Chave(s) Primária(s): Realizador (**NOT NULL**), Filme (**NOT NULL**)
Chave(s) Estrangeira(s): Realizador - Referência a **Realizador (id_realizador)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Personagem_Entra_Filme = {Personagem, Filme}

Chave(s) Primária(s): Personagem (**NOT NULL)**, Filme (**NOT NULL)**

Chave(s) Estrangeira(s): Personagem - Referência a **Personagem (id_personagem)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Filme_Tem_Genero = {Filme, Genero}

Chave(s) Primária(s): Filme (**NOT NULL)**, Genero (**NOT NULL)**

Chave(s) Estrangeira(s): Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Genero - Referência a **Genero (Nome)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Utilizador_Avalia_Filme = {Utilizador, Filme, Rating_User, Comentario}

Chave(s) Primária(s): Utilizador (**NOT NULL)**, Filme (**NOT NULL)**

Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Utilizador_Gosta_Filme = {Utilizador, Filme}

Chave(s) Primária(s): Utilizador (**NOT NULL)**, Filme (**NOT NULL)**

Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Utilizador_Visualiza_Filme = {Utilizador, Filme}

Chave(s) Primária(s): Utilizador (**NOT NULL)**, Filme (**NOT NULL)**

Chave(s) Estrangeira(s): Utilizador - Referência a **Utilizador (email)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)
Filme - Referência a **Filme (id_filme)**
(**NOT NULL, ON UPDATE NO ACTION, ON DELETE NO ACTION**)

Assim, concluímos:

1 – As restrições de integridade de entidade são cumpridas, pois nenhuma das chaves primárias do modelo permitem valores nulos;

2 – As restrições de integridade referencial são cumpridas, pois não são permitidas chaves estrangeiras numa tabela sem que a respetiva chave primária exista noutra tabela, e estas mesmas também não poderão ser nulas;

3 – As restrições de integridade de domínio são cumpridas, pois de acordo com os domínios e tipos definidos nos requisitos, estes foram restringidos diretamente no modelo lógico;

4 – A multiplicidade manteve-se na conversão do modelo Conceptual para o Lógico;

5.6. Verificar Modelo Lógico com o Utilizador

Tendo em conta a natureza do trabalho e a inexistência de um utilizador concreto para possível discussão, esta etapa do projeto não foi considerada.

5.7. Verificar Expansão Futura do Modelo

Neste desfecho da secção em que retratamos todas as considerações e reflexões debruçadas sobre o modelo lógico da nossa plataforma é importante relembrar e não deixar de referir a possível capacidade que o nosso sistema poderá ter no que toca a futuras alterações e poder de extensibilidade através de, por exemplo, adição de novas funcionalidades à nossa Biblioteca de Filmes.

O grupo de projeto terá tido em conta todos os requisitos necessários para um SGBD que permitisse de uma forma genérica albergar as funcionalidades mais básicas de outras plataformas já existentes (Ex: Popcorn Time, Kodi), pelo que consideramos que os modelos aqui apresentados irão permitir a existência de um sistema confiável, estável, duradouro e “atómico”, no sentido que se comporte como a “semente” para a futura possibilidade de incorporação de novas utilidades na plataforma.

6. Modelo Físico

Chegamos à fase final deste trabalho, ponto o qual desvendamos a nossa plataforma, mas antes disso necessitamos de recorrer a um SGDB, a um motor de bases de dados que nos permita construir esta realidade.

Utilizamos o MySQL Workbench, desenvolvido pela Oracle, ferramenta muito pouco exigente de uma máquina nos seus requisitos e demonstra grandes expetativas na sua *performance*.

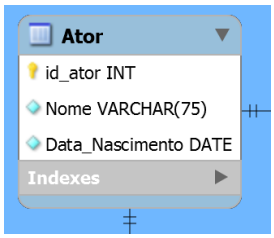
Tendo já escolhido o SGDB partimos para a construção das tabelas apresentadas anteriormente pelo modelo Lógico, pequenas *queries* (Procedimentos, Funções, etc.) e transações muito úteis para o funcionamento da biblioteca de filmes e medição de dados proficientes para a resposta a certas questões encontradas nesta secção.

6.1. Implementação e Conversão do Modelo Lógico para um SGBD

Para esta parte do trabalho consideramos a tradução do nosso modelo Lógico para o modelo Físico através do SGBD do MySQL Workbench.

Esta tarefa revelou - se bastante fácil, pois esta ferramenta apresenta um mecanismo denominado *Forward Enginner* que, a partir da nossa representação do modelo Lógico (apresentado na secção anterior), gera o código SQL que permite criar as tabelas correspondentes respeitando os tipos de dados que foram indicados e todas as referências entre tabelas (chaves estrangeiras).

6.1.1 Criação de Relações




```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Ator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Ator` (
  `id_ator` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(75) NOT NULL,
  `Data_Nascimento` DATE NOT NULL,
  PRIMARY KEY (`id_ator`))
ENGINE = InnoDB;
```

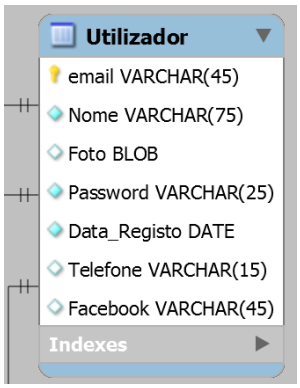
Figura 17 – Tabela Ator


```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Personagem`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Personagem` (
  `id_personagem` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(75) NOT NULL,
  PRIMARY KEY (`id_personagem`))
ENGINE = InnoDB;
```

Figura 18 – Tabela Personagem

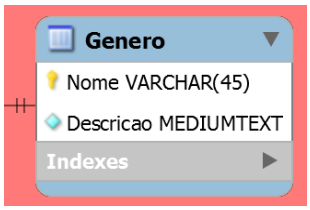

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Filme` (
  `id_filme` INT NOT NULL AUTO_INCREMENT,
  `Titulo` VARCHAR(45) NOT NULL,
  `Ano_Lancamento` INT NOT NULL,
  `Descricao` LONGTEXT NOT NULL,
  `Duracao` TIME(0) NOT NULL,
  `Rating_Geral` FLOAT NULL DEFAULT 0,
  PRIMARY KEY (`id_filme`))
ENGINE = InnoDB;
```

Figura 19 – Tabela Filme



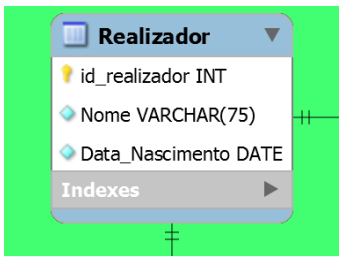

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Utilizador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Utilizador` (
  `email` VARCHAR(45) NOT NULL,
  `Nome` VARCHAR(75) NOT NULL,
  `Foto` BLOB NULL,
  `Password` VARCHAR(25) NOT NULL,
  `Data_Registo` DATE NOT NULL,
  `Telefone` VARCHAR(15) NULL,
  `Facebook` VARCHAR(45) NULL,
  PRIMARY KEY (`email`))
ENGINE = InnoDB;
```

Figura 20 – Tabela Utilizador



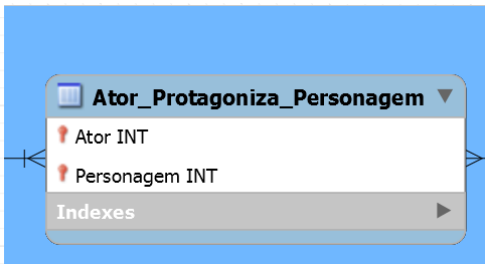

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Genero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Genero` (
  `Nome` VARCHAR(45) NOT NULL,
  `Descricao` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`Nome`))
ENGINE = InnoDB;
```

Figura 21 – Tabela Genero



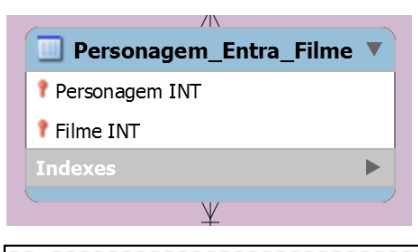

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Realizador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Realizador` (
  `id_realizador` INT NOT NULL AUTO_INCREMENT,
  `Nome` VARCHAR(75) NOT NULL,
  `Data_Nascimento` DATE NOT NULL,
  PRIMARY KEY (`id_realizador`))
ENGINE = InnoDB;
```

Figura 22 – Tabela Realizador



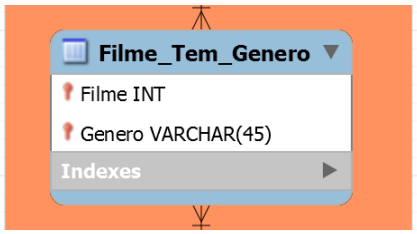

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Ator_Protagoniza_Personagem`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Ator_Protagoniza_Personagem` (
  `Ator` INT NOT NULL,
  `Personagem` INT NOT NULL,
  PRIMARY KEY (`Ator`, `Personagem`),
  INDEX `fk_Ator_has_Personagem_Personagem1_idx` (`Personagem` ASC),
  INDEX `fk_Ator_has_Personagem_Ator_idx` (`Ator` ASC),
  CONSTRAINT `fk_Ator_has_Personagem_Ator`
    FOREIGN KEY (`Ator`)
    REFERENCES `BiblioFilmes`.`Ator` (`id_ator`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Ator_has_Personagem_Personagem1`
    FOREIGN KEY (`Personagem`)
    REFERENCES `BiblioFilmes`.`Personagem` (`id_personagem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 23 – Tabela Ator_Protagoniza_Personagem



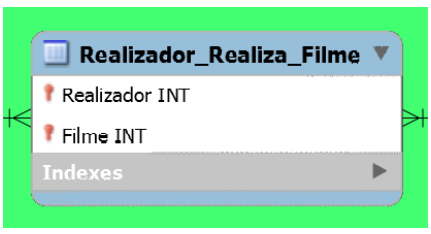

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Personagem_Entra_Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Personagem_Entra_Filme` (
  `Personagem` INT NOT NULL,
  `Filme` INT NOT NULL,
  PRIMARY KEY (`Personagem`, `Filme`),
  INDEX `fk_Personagem_has_Filme_Filme1_idx` (`Filme` ASC),
  INDEX `fk_Personagem_has_Filme_Personagem1_idx` (`Personagem` ASC),
  CONSTRAINT `fk_Personagem_has_Filme_Personagem1`
    FOREIGN KEY (`Personagem`)
    REFERENCES `BiblioFilmes`.`Personagem` (`id_personagem`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Personagem_has_Filme_Filme1`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 24 – Tabela Personagem_Entra_Filme



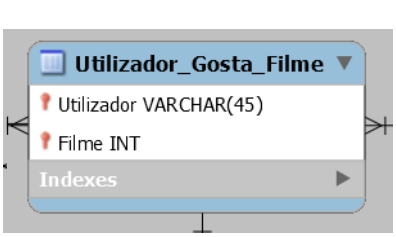

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Filme_Tem_Genero`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Filme_Tem_Genero` (
  `Filme` INT NOT NULL,
  `Genero` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Filme`, `Genero`),
  INDEX `fk_Genero_has_Filme_Filme1_idx` (`Filme` ASC),
  INDEX `fk_Genero_has_Filme_Genero1_idx` (`Genero` ASC),
  CONSTRAINT `fk_Genero_has_Filme_Genero1`
    FOREIGN KEY (`Genero`)
    REFERENCES `BiblioFilmes`.`Genero` (`Nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Genero_has_Filme_Filme1`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 25 – Tabela Filme_Tem_Genero



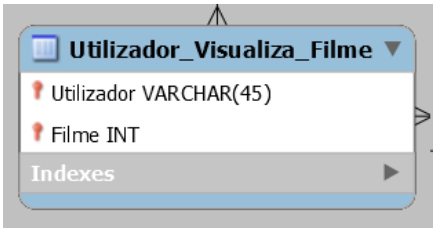

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Realizador_Realiza_Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Realizador_Realiza_Filme` (
  `Realizador` INT NOT NULL,
  `Filme` INT NOT NULL,
  PRIMARY KEY (`Realizador`, `Filme`),
  INDEX `fk_Realizador_has_Filme_Filme1_idx` (`Filme` ASC),
  INDEX `fk_Realizador_has_Filme_Realizador1_idx` (`Realizador` ASC),
  CONSTRAINT `fk_Realizador_has_Filme_Realizador1`
    FOREIGN KEY (`Realizador`)
    REFERENCES `BiblioFilmes`.`Realizador` (`id_realizador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Realizador_has_Filme_Filme1`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 26 – Tabela Realizador_Realiza_Filme



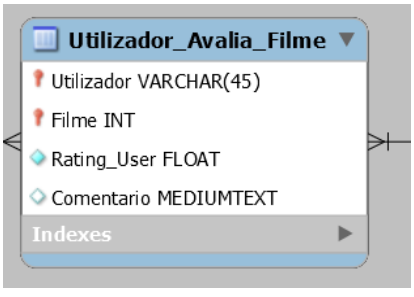

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Utilizador_Gosta_Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Utilizador_Gosta_Filme` (
  `Utilizador` VARCHAR(45) NOT NULL,
  `Filme` INT NOT NULL,
  PRIMARY KEY (`Utilizador`, `Filme`),
  INDEX `fk_Filme_has_Utilizador_Utilizador2_idx` (`Utilizador` ASC),
  INDEX `fk_Filme_has_Utilizador_Filme2_idx` (`Filme` ASC),
  CONSTRAINT `fk_Filme_has_Utilizador_Filme2`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Utilizador_Utilizador2`
    FOREIGN KEY (`Utilizador`)
    REFERENCES `BiblioFilmes`.`Utilizador` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 27 – Tabela Utilizador_Gosta_Filme



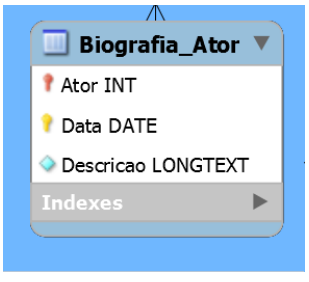

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Utilizador_Visualiza_Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Utilizador_Visualiza_Filme` (
  `Utilizador` VARCHAR(45) NOT NULL,
  `Filme` INT NOT NULL,
  PRIMARY KEY (`Utilizador`, `Filme`),
  INDEX `fk_Filme_has_Utilizador_Utilizador3_idx` (`Utilizador` ASC),
  INDEX `fk_Filme_has_Utilizador_Filme3_idx` (`Filme` ASC),
  CONSTRAINT `fk_Filme_has_Utilizador_Filme3`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Utilizador_Utilizador3`
    FOREIGN KEY (`Utilizador`)
    REFERENCES `BiblioFilmes`.`Utilizador` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 28 – Tabela Utilizador_Visualiza_Filme



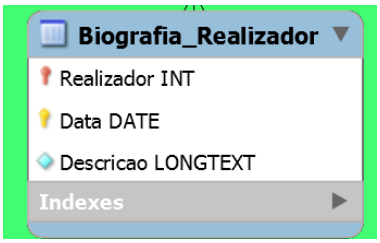

```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Utilizador_Avalia_Filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Utilizador_Avalia_Filme` (
  `Utilizador` VARCHAR(45) NOT NULL,
  `Filme` INT NOT NULL,
  `Rating_User` FLOAT NOT NULL,
  `Comentario` MEDIUMTEXT NULL,
  PRIMARY KEY (`Utilizador`, `Filme`),
  INDEX `fk_Filme_has_Utilizador_Utilizador1_idx` (`Utilizador` ASC),
  INDEX `fk_Filme_has_Utilizador_Filme1_idx` (`Filme` ASC),
  CONSTRAINT `fk_Filme_has_Utilizador_Filme1`
    FOREIGN KEY (`Filme`)
    REFERENCES `BiblioFilmes`.`Filme` (`id_filme`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Filme_has_Utilizador_Utilizador1`
    FOREIGN KEY (`Utilizador`)
    REFERENCES `BiblioFilmes`.`Utilizador` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 29 – Tabela Utilizador_Avalia_Filme




```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Biografia_Ator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Biografia_Ator` (
  `Ator` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Descricao` LONGTEXT NOT NULL,
  PRIMARY KEY (`Data`, `Ator`),
  INDEX `fk_Biografia_Ator1_idx` (`Ator` ASC),
  CONSTRAINT `fk_Biografia_Ator1`
    FOREIGN KEY (`Ator`)
    REFERENCES `BiblioFilmes`.`Ator` (`id_ator`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 30 – Tabela Biografia_Ator




```
-- -----------------------------------------------------
-- Tabela `BiblioFilmes`.`Biografia_Realizador`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `BiblioFilmes`.`Biografia_Realizador` (
  `Realizador` INT NOT NULL,
  `Data` DATE NOT NULL,
  `Descricao` LONGTEXT NOT NULL,
  PRIMARY KEY (`Realizador`, `Data`),
  INDEX `fk_Realizador_Bio_Realizador1_idx` (`Realizador` ASC),
  CONSTRAINT `fk_Realizador_Bio_Realizador1`
    FOREIGN KEY (`Realizador`)
    REFERENCES `BiblioFilmes`.`Realizador` (`id_realizador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 31 – Tabela Biografia_Realizador

6.1.2 Relações Base

Para o desenho das relações base, para além de utilizarmos as descrições de tabelas e atributos definidos aquando da realização do modelo Conceptual do nosso projeto, respeitámos ainda os tipos de dados que se encontravam definidos no momento da elaboração do dicionário de dados, o que nos permitiu determinar qual o domínio dos atributos de cada tabela, para além dos valores por defeito e chaves primárias/estrangeiras.

6.1.3 Representação de Dados Derivados

Passamos agora para o tratamento dos atributos derivados na nossa implementação. No nosso caso apenas encontramos um atributo derivado que se designa por Rating Geral, atributo que se encontra num Filme.

Quando é inserida uma nova avaliação de um Utilizador, o Rating Geral necessita de ser atualizado para o novo valor, que consiste sempre na média de todas as classificações atribuídas pelos utilizadores até ao momento.

Quanto à implementação, decidimos implementar três *Triggers*, um para inserção, um para remoção e outro para a atualização de forma a tratar este caso.

Quando um valor for inserido/atualizado/removido, os *Triggers* vão atuar de modo a atualizar o valor do Rating Geral do Filme afetado.

Como alternativa, estes *Triggers* poderiam ter sido substituídos por um *Stored Procedure*, que seria invocado de cada vez que um registo fosse inserido, atualizado ou removido.

Depois de analisarmos os dois casos, considerámos que os *Triggers* seriam a melhor opção pois não envolvem intervenção direta de nenhum utilizador da BD, o que poderia levar à existência de inconsistências nas atualizações e consequente incoerência no funcionamento do nosso sistema.

```
-- ------------------------------------------------------
-- Triggers de Atualização do Rating Geral
-- ------------------------------------------------------
DELIMITER |
CREATE TRIGGER Atualiza_Rating_Insert
AFTER INSERT ON Utilizador_Avalia_Filme
FOR EACH ROW
BEGIN
  UPDATE Filme AS F
    SET F.Rating_Geral =
      (SELECT AVG(UAF.Rating_User)
        FROM Utilizador_Avalia_Filme AS UAF
        WHERE UAF.Filme = NEW.Filme)
    WHERE id_filme = NEW.Filme;
END
|
DELIMITER ;

DELIMITER |
CREATE TRIGGER Atualiza_Rating_Update
AFTER UPDATE ON Utilizador_Avalia_Filme
FOR EACH ROW
BEGIN
  UPDATE Filme AS F
    SET F.Rating_Geral =
      (SELECT AVG(UAF.Rating_User)
        FROM Utilizador_Avalia_Filme AS UAF
        WHERE UAF.Filme = NEW.Filme)
    WHERE id_filme = NEW.Filme;
END
|
DELIMITER ;

DELIMITER |
CREATE TRIGGER Atualiza_Rating_Delete
AFTER DELETE ON Utilizador_Avalia_Filme
FOR EACH ROW
BEGIN
  UPDATE Filme AS F
    SET F.Rating_Geral =
      (SELECT AVG(UAF.Rating_User)
        FROM Utilizador_Avalia_Filme AS UAF
        WHERE UAF.Filme = OLD.Filme)
    WHERE id_filme = OLD.Filme;
END
|
DELIMITER ;
```

Figura 32 – Triggers Rating_Geral

6.1.4 Restrições Gerais

Para este trabalho foi apenas necessário criar uma restrição geral. Esta restrição indica que as avaliações inseridas pelos respetivos utilizadores devem encontrar – se no intervalo entre 0.0 e 10.0. De forma a definir este limite, desenvolvemos o seguinte código SQL:

```
-- ------------------------------------------------------
-- Restrição de verificação do valor da avaliação do utilizador
-- ------------------------------------------------------
DELIMITER |
CREATE TRIGGER Avaliacao_Entre_0_e_10_Insert
BEFORE INSERT ON Utilizador_Avalia_Filme
FOR EACH ROW
BEGIN
  IF (NEW.Rating_User < 0.0 OR NEW.Rating_User > 10.0)
  THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'A avaliação só pode ser entre 0 e 10';
  END IF;
END
|
DELIMITER ;

DELIMITER |
CREATE TRIGGER Avaliacao_Entre_0_e_10_Update
BEFORE UPDATE ON Utilizador_Avalia_Filme
FOR EACH ROW
BEGIN
  IF (NEW.Rating_User < 0.0 OR NEW.Rating_User > 10.0)
  THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'A avaliação só pode ser entre 0 e 10';
  END IF;
END
|
DELIMITER ;
```

Figura 33 – Restrição Rating_Geral

6.1.5 Triggers

Como já foi abordado, os *Triggers* criados foram aqueles que nos permitissem atualizar o Rating_Geral de um filme após uma inserção, atualização ou remoção de uma classificação por parte de um utilizador e os responsáveis pela análise do valor do Rating de um Utilizador antes da sua inserção na BD.

6.2. Analisar Transações

Nesta secção analisamos as transações e procedimentos criados na nossa BD, com vista a uma possível otimização de performance.

6.2.1 Transações Críticas à Operacionalidade do Negócio

No caso da nossa Biblioteca de Filmes, e dada a análise das transações que realizámos, não encontrámos transações demasiado complexas que justificassem a criação de barreiras na execução de qualquer *query* do nosso sistema.

6.2.2 Períodos de Maior Utilização da Base de Dados

Assumimos que o fluxo de acesso à nossa plataforma surgirá sempre entre limites considerados normais, pela que esta não terá períodos de maior utilização definidos.

6.2.3 Mapa dos Caminhos Transacionais nas Relações

Nesta secção mostramos o mapa de utilização de tabelas por parte das seguintes transações:

Nota: De notar que o código SQL desenvolvido para estes procedimentos encontra – se em anexo neste documento.

(1) – Identificar filmes de um dado ano;
(2) – Identificar atores que participaram em um dado filme;
(3) – Identificar filmes de um dado género;
(4) – Identificar top X filmes em termos de *rating*;
(5) – Identificar filmes em que um dado ator entrou;
(6) – Identificar filmes de um dado realizador;
(7) – Identificar filmes por intervalos de duração;
(8) – Inserir um filme na base de dados;
(9) – Identificar atores que protagonizam uma dada personagem;
(10) – Inserir um acontecimento na biografia de um ator;
(11) – Inserir um acontecimento na biografia de um realizador;
(12) – Atualizar a informação de um dado utilizador;
(13) – Avaliar um filme;
(14) – Adicionar um filme aos favoritos de um dado utilizador;
(15) – Indicar que um filme foi visualizado por um utilizador;

	(1)				(2)				(3)				(4)				(5)				(6)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Ator						X																		
Biografia_Ator																								
Ator_Protagoniza_Personagem						X												X						
Personagem						X																		
Personagem_Entra_Filme						X												X						
Realizador																								
Biografia_Realizador																								
Realizador_Realiza_Filme																						X		
Filme		X								X				X				X				X		
Filme_Tem_Genero										X														
Genero																								
Utilizador																								
Utilizador_Avalia_Filme																								
Utilizador_Gosta_Filme																								
Utilizador_Visualiza_Filme																								

	(7)				(8)				(9)				(10)				(11)				(12)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Ator					X					X														
Biografia_Ator													X											
Ator_Protagoniza_Personagem					X					X														
Personagem					X																			
Personagem_Entra_Filme					X																			
Realizador					X																			
Biografia_Realizador																		X						
Realizador_Realiza_Filme					X																			
Filme		X			X																			
Filme_Tem_Genero					X																			
Genero					X																			
Utilizador																							X	
Utilizador_Avalia_Filme																								
Utilizador_Gosta_Filme																								
Utilizador_Visualiza_Filme																								

	(13)				(14)				(15)			
	I	R	U	D	I	R	U	D	I	R	U	D
Ator												
Biografia_Ator												
Ator_Protagoniza_Personagem												
Personagem												
Personagem_Entra_Filme												
Realizador												
Biografia_Realizador												
Realizador_Realiza_Filme												
Filme												
Filme_Tem_Genero												
Genero												
Utilizador												

Utilizador_Avalia_Filme	X											
Utilizador_Gosta_Filme					X							
Utilizador_Visualiza_Filme									X			

LEGENDA:

- I – *Insert* R – *Read* U – *Update* D – *Delete*

Tabela 4 – Caminhos transacionais nas relações

6.2.4 Frequência de Informação

Tendo em conta que as tabelas de Filmes, Utilizadores, Personagens, Atores e Realizadores poderão conter mais informação futuramente considerámos desenvolver procedimentos e transações que fossem o mais eficiente possível de modo a tornar todas as operações efetuadas nas respetivas tabelas mais rápidas em tempo de execução.

6.3. Representação de Transações

Neste tópico apresenta - se a implementação em SQL das transações abordadas na Secção 5.4.

```
DELIMITER |
CREATE PROCEDURE Insere_Filme
  (IN titulo VARCHAR(45), IN Ano INT ,
   IN descricao LONGTEXT, IN duracao TIME,
   IN generos VARCHAR(255),
   IN personagens VARCHAR(1000),
   IN atores VARCHAR(1000),
   IN datas VARCHAR(1000),
   IN realizadores VARCHAR(255),
   IN datas2 VARCHAR(1000))
BEGIN
  DECLARE lastIDFilme INT;
  DECLARE Erro BOOL DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;
  START TRANSACTION;
    INSERT INTO Filme
      (Titulo, Ano_Lancamento, Descricao, Duracao)
      VALUES
      (titulo, Ano, descricao, duracao);

    SET lastIDFilme = LAST_INSERT_ID();

    CALL Insere_Generos(generos, lastIdFilme);
    CALL Insere_Atores_Personagens(personagens, atores, datas, lastIdFilme);
    CALL Insere_Realizadores(realizadores, datas2, lastIdFilme);

    IF Erro THEN
      ROLLBACK;
    ELSE
      COMMIT;
    END IF;
END
|
DELIMITER ;
```

Figura 34 – Transação Insere_Filme

```
DELIMITER |
CREATE PROCEDURE Insere_Generos
  (IN generos VARCHAR(255),
   IN filme INT)
BEGIN
  DECLARE strLen INT DEFAULT 0;
  DECLARE SubStrLen INT DEFAULT 0;
  IF generos IS NULL
  THEN
    SET generos = '';
  END IF;

  Generos:LOOP
    SET strLen = CHAR_LENGTH(generos);

    INSERT INTO Genero
      VALUES
      (SUBSTRING_INDEX(generos, ',', 1), 'Descrição do Género')
      ON DUPLICATE KEY UPDATE Nome=Nome;

    INSERT INTO Filme_Tem_Genero
      VALUES
      (filme, SUBSTRING_INDEX(generos, ',', 1));

    SET SubStrLen = CHAR_LENGTH(SUBSTRING_INDEX(generos, ',', 1)) + 2;
    SET generos = MID(generos, SubStrLen, strLen);

    IF generos = ''
    THEN
      LEAVE Generos;
    END IF;
  END LOOP Generos;
END
|
DELIMITER ;
```

Figura 35 – Componente Insere_Generos

```
DELIMITER |
CREATE PROCEDURE Insere_Atores_Personagens
  (IN personagens VARCHAR(1000),
   IN atores VARCHAR(1000),
   IN datas VARCHAR(1000),
   IN filme INT)
BEGIN
  DECLARE strLen INT DEFAULT 0;
  DECLARE strLen2 INT DEFAULT 0;
  DECLARE strLen3 INT DEFAULT 0;
  DECLARE SubStrLen INT DEFAULT 0;
  DECLARE SubStrLen2 INT DEFAULT 0;
  DECLARE SubStrLen3 INT DEFAULT 0;

  IF personagens IS NULL THEN
    SET personagens = '';
  END IF;
  IF atores IS NULL THEN
    SET atores = '';
  END IF;
  IF datas IS NULL THEN
    SET datas = '';
  END IF;
```

Figura 36 – Componente Insere_Atores_Personagens (1ªParte)

```
  Personagens_Atores:LOOP
    SET strLen = CHAR_LENGTH(personagens);
    SET strLen2 = CHAR_LENGTH(atores);
    SET strLen3 = CHAR_LENGTH(datas);

  IF (NOT EXISTS (SELECT * FROM Personagem
                    WHERE Nome=SUBSTRING_INDEX(personagens, ',', 1)))
  THEN
    INSERT INTO Personagem
      (Nome)
      VALUES
      (SUBSTRING_INDEX(personagens, ',', 1));

    SET @lastIDPersonagem = LAST_INSERT_ID();
  ELSE
    SET @lastIDPersonagem = (SELECT id_personagem FROM Personagem
      WHERE Nome=SUBSTRING_INDEX(personagens, ',', 1));
  END IF;

  IF (NOT EXISTS (SELECT *
                    FROM Ator
                    WHERE Nome=SUBSTRING_INDEX(atores, ',', 1)
                      AND Data_Nascimento=SUBSTRING_INDEX(datas, ',', 1)))
  THEN
    INSERT INTO Ator
      (Nome, Data_Nascimento)
      VALUES
      (SUBSTRING_INDEX(atores, ',', 1), SUBSTRING_INDEX(datas, ',', 1));

    SET @lastIDAtor = LAST_INSERT_ID();
  ELSE
    SET @lastIDAtor = (SELECT id_ator
                         FROM Ator
                         WHERE Nome=SUBSTRING_INDEX(atores, ',', 1)
                           AND Data_Nascimento=SUBSTRING_INDEX(datas, ',', 1));
  END IF;

  INSERT INTO Personagem_Entra_Filme
    VALUES
    (@lastIDPersonagem, filme)
    ON DUPLICATE KEY UPDATE Personagem=Personagem;

    INSERT INTO Ator_Protagoniza_Personagem
      VALUES
      (@lastIDAtor, @lastIDPersonagem)
      ON DUPLICATE KEY UPDATE Ator=Ator;

    SET SubStrLen = CHAR_LENGTH(SUBSTRING_INDEX(personagens, ',', 1)) + 2;
    SET SubStrLen2 = CHAR_LENGTH(SUBSTRING_INDEX(atores, ',', 1)) + 2;
    SET SubStrLen3 = CHAR_LENGTH(SUBSTRING_INDEX(datas, ',', 1)) + 2;
    SET personagens = MID(personagens, SubStrLen, strLen);
    SET atores = MID(atores, SubStrLen2, strLen2);
    SET datas = MID(datas, SubStrLen3, strLen3);

    IF personagens = '' AND atores = '' AND datas = '' THEN
      LEAVE Personagens_Atores;
    END IF;
  END LOOP Personagens_Atores;
END
|
DELIMITER ;
```

Figura 37 – Componenete Insere_Atores_Personagens (2ªParte)

```
DELIMITER |
CREATE PROCEDURE Insere_Realizadores
  (IN realizadores VARCHAR(255),
   IN datas VARCHAR(1000),
   IN filme INT)
BEGIN
  DECLARE strLen INT DEFAULT 0;
  DECLARE strLen2 INT DEFAULT 0;
  DECLARE SubStrLen INT DEFAULT 0;
  DECLARE SubStrLen2 INT DEFAULT 0;

  IF realizadores IS NULL
  THEN
    SET realizadores = '';
  END IF;
  IF datas IS NULL
  THEN
    SET datas = '';
  END IF;

  Realizadores:LOOP
    SET strLen = CHAR_LENGTH(realizadores);
    SET strLen2 = CHAR_LENGTH(datas);

    IF (NOT EXISTS (SELECT *
                    FROM Realizador
                    WHERE Nome=SUBSTRING_INDEX(realizadores, ',', 1)
                      AND Data_Nascimento=SUBSTRING_INDEX(datas, ',', 1)))
    THEN
      INSERT INTO Realizador
        (Nome, Data_Nascimento)
        VALUES
        (SUBSTRING_INDEX(realizadores, ',', 1), SUBSTRING_INDEX(datas, ',', 1));

      SET @lastIDRealizador = LAST_INSERT_ID();
    ELSE
      SET @lastIDRealizador = (SELECT id_realizador
                               FROM Realizador
                               WHERE Nome=SUBSTRING_INDEX(realizadores, ',', 1)
                                 AND Data_Nascimento=SUBSTRING_INDEX(datas, ',', 1));
    END IF;

    INSERT INTO Realizador_Realiza_Filme
      VALUES
      (@lastIDRealizador, filme);

    SET SubStrLen = CHAR_LENGTH(SUBSTRING_INDEX(realizadores, ',', 1)) + 2;
    SET SubStrLen2 = CHAR_LENGTH(SUBSTRING_INDEX(datas, ',', 1)) + 2;
    SET realizadores = MID(realizadores, SubStrLen, strLen);
    SET datas = MID(datas, SubStrLen, strLen);

    IF realizadores = '' AND datas = ''
    THEN
      LEAVE Realizadores;
    END IF;
  END LOOP Realizadores;
END
|
DELIMITER ;
```

Figura 38 – Componente Insere_Realizadores

6.4. Escolha de Organização de Ficheiros

Neste caso, a organização dos ficheiros na BD não foi alterada, pelo que foi utilizada a configuração por defeito do MySQL, a configuração em B-Tree.

6.5. Escolha de Índices

Neste caso não foram criados índices adicionais para além dos que foram criados no momento da geração do código SQL utilizando a ferramenta MySQL Workbench.

6.6. Definição de Vistas de Utilizadores

Decidimos não criar nenhuma vista específica para utilizadores, pois as operações mais comuns neste sistema encontram - se já implementadas pelos procedimentos apresentados.

6.7. Política de Segurança

Nesta secção tivemos em conta a segurança da base de dados e os perfis de utilização referidos durante a elaboração do modelo Conceptual da nossa base de dados.

6.7.1 Regras de Acesso

Criando dois tipos de utilizadores, "*admin*" e "*user"*, cobrimos os perfis de utilização referidos durante a análise dos requisitos da nossa plataforma, que se encontra detalhada na Secção 2 deste documento.

6.7.2 Dados

Em relação à segurança nos dados, decidimos atribuir permissões distintas aos dois tipos de utilizadores criados nesta BD.

O utilizador "*admin*" tem acesso a tudo e poderá fazer as modificações que sejam necessárias na BD, desde criar uma tabela, selecionar registos e até apagar a base de dados inteira.

O utilizador "*user*" será o utilizador normal da base de dados, o qual terá permissão de realizar a seleção, a inserção e a atualização de registos nas tabelas Utilizador_Avalia_Filme, Utilizador_Visualiza_Filme, Utilizador_Gosta_Filme e Utilizador.

Este último tem ainda a possibilidade de fazer seleções em todas as tabelas da BD. Quanto a operações de criação, atualização e remoção de valores nas restantes tabelas da BD, estas são reservadas ao administrador, não sendo assim possível ao utilizador comum fazer as alterações.

```
-- -----------------------------------------------------
-- Criação de Utilizadores e Atribuição de Privilégios
-- -----------------------------------------------------
CREATE USER 'admin'@'localhost'
  IDENTIFIED BY '123';
GRANT ALL PRIVILEGES ON BiblioFilmes.*
  TO 'admin'@'localhost';

CREATE USER 'user'@'localhost'
  IDENTIFIED BY '1234';
GRANT SELECT ON BiblioFilmes.*
  TO 'user'@'localhost';
GRANT INSERT, UPDATE ON BiblioFilmes.Utilizador
  TO 'user'@'localhost';
GRANT INSERT, UPDATE ON BiblioFilmes.Utilizador_Avalia_Filme
  TO 'user'@'localhost';
GRANT INSERT, UPDATE ON BiblioFilmes.Utilizador_Gosta_Filme
  TO 'user'@'localhost';
GRANT INSERT, UPDATE ON BiblioFilmes.Utilizador_Visualiza_Filme
  TO 'user'@'localhost';
```

Figura 39 – Utilizadores e atribuição de privilégios

Em termos de *backups*, uma vez que o número de entrada de novos filmes no sistema não é tão elevado por mês, será apenas necessário fazer um backup de todos os dados mensalmente.

Pelo menos considerámos que será necessária a verificação dos *backups* dos últimos dois meses. Estes *backups* poderão ser guardados num servidor de dados e este terá que ser distribuído em caso de avaria de alguma máquina.

6.8. Povoamento da BD

Para o povoamento da nossa BD, recolhemos informações acerca de alguns filmes e utilizámos as transações criadas para adicionar toda a informação destes filmes na nossa plataforma.

Para além disso, criámos perfis de Utilizador para cada um dos elementos do grupo e avaliámos os filmes existentes, usando para isso a transação criada para o efeito.

```
INSERT INTO Utilizador (email,Nome,Password,Data_Registo)
  VALUES
  ('a71184@alunos.uminho.pt','João Pedro Fontes','123','2016-01-19'),
  ('a67847@alunos.uminho.pt','Bruno Renato Carvalho','123','2016-01-19'),
  ('a64309@alunos.uminho.pt','Pedro Vieira Fortes','123','2016-01-19'),
  ('a70377@alunos.uminho.pt','André Filipe Silva','123','2016-01-19');
```

Figura 40 – Povoamento com perfis de Utilizador

```
CALL Insere_Filme('The Shawshank Redemption', 1994,
  'Two imprisoned men bond over a number of years,
  finding solace and eventual redemption through acts of common decency.',
  '2:22:00', 'Drama','Andy Dufresne,Red','Tim Robbins,Morgan Freeman','1958-10-16,1937-06-01',
  'Frank Darabont','1958-01-29');
CALL Insere_Filme('The Godfather', 1972,
  'The aging patriarch of an organized crime dynasty transfers
  control of his clandestine empire to his reluctant son.',
  '2:55:00', 'Drama,Crime','Don Vito Corleone,Michael Corleone,Tom Hagen',
  'Marlon Brando,Al Pacino,Robert Duvall','1924-04-03,1940-04-25,1931-01-05',
  'Francis Ford Coppola','1939-04-07');
CALL Insere_Filme('The Godfather: Part II', 1974,
  'The early life and career of Vito Corleone in 1920s New York is portrayed
  while his son, Michael, expands and tightens his grip on his crime syndicate
  stretching from Lake Tahoe, Nevada to pre-revolution 1958 Cuba.',
  '3:22:00', 'Drama,Crime','Don Vito Corleone,Michael Corleone,Tom Hagen',
  'Robert De Niro,Al Pacino,Robert Duvall','1943-08-17,1940-04-25,1931-01-05',
  'Francis Ford Coppola','1939-04-07');
CALL Insere_Filme('The Dark Knight', 2008,
  'When the menace known as the Joker wreaks havoc and chaos on the people
  of Gotham, the caped crusader must come to terms with one of the greatest
  psychological tests of his ability to fight injustice.',
  '1:32:00', 'Action','Bruce Wayne,Joker','Christian Bale,Heath Ledger','1974-01-30,1979-04-04',
  'Christopher Nolan','1970-07-30');
CALL Insere_Filme('Pulp Fiction', 1994,
  'The lives of two mob hit men, a boxer, a gangster s
  wife, and a pair of diner bandits intertwine in four
  tales of violence and redemption.',
  '1:36:00', 'Drama,Crime','Vincent Vega,Jules Winnfield',
  'John Travolta,Samuel L. Jackson','1954-02-18,1948-12-21','Quentin Tarantino','1963-03-27');
```

Figura 41 – Povoamento com Filmes

```
CALL Avalia_Filme('a71184@alunos.uminho.pt',1,9.0);
CALL Avalia_Filme('a71184@alunos.uminho.pt',2,7.0);
CALL Avalia_Filme('a71184@alunos.uminho.pt',3,7.0);
CALL Avalia_Filme('a71184@alunos.uminho.pt',4,10.0);
CALL Avalia_Filme('a71184@alunos.uminho.pt',5,8.0);
CALL Avalia_Filme('a67847@alunos.uminho.pt',5,10.0);
```

Figura 42 – Povoamento com Avaliação de Filmes

6.9. Estimar Uso de Disco para os Dados Referidos e Estimar Futuro Crescimento

Para realizar uma estimativa dos requisitos em termos de espaço da nossa BD, corremos o seguinte código SQL:

```
-- ------------------------------------------------------
-- Calcular o tamanho da BD em bytes
-- ------------------------------------------------------
DELIMITER ;
SELECT table_name AS 'Tabela', (data_length+index_length) AS 'Tamanho em Bytes'
  FROM information_schema.tables
  WHERE table_schema='BiblioFilmes';
```

Figura 43 – Requisitos de espaço da BD

Com este código conseguimos obter o tamanho da nossa BD no momento (em bytes). Apresentamos agora os resultados que adquirimos:

Tabela	Tamanho em Bytes
Ator	16384
Biografia_Ator	32768
Ator_Protagoniza_Personagem	49152
Personagem	16384
Personagem_Entra_Filme	49152
Realizador	16384
Biografia_Realizador	32768
Realizador_Realiza_Filme	49152
Filme	16384
Filme_Tem_Genero	49152
Genero	16384
Utilizador	16384
Utilizador_Avalia_Filme	49152
Utilizador_Gosta_Filme	49152
Utilizador_Visualiza_Filme	49152

Tabela 5 – Resultados do tamanho da BD em bytes

6.10. Validação de Transações com o Utilizador

Quanto às transações que tínhamos anteriormente definido no período de desenvolvimento do modelo Conceptual da nossa BD acabámos for criar esses mesmos elementos durante esta fase da elaboração do modelo Físico, sendo que o nosso sistema tem a capacidade de fazer tudo o que o utilizador comum da BD precisará de fazer com ela, desde consultar o top de filmes até adicionar um filme novo, com personagens, atores, realizadores e géneros.

7. Conclusão e Análise Crítica

Concluído o projeto é possível rever alguns pontos importantes na implementação desta BD desde a dificuldade na escolha de um tema suficientemente original e complexo, que iria influenciar o desenvolvimento do problema incluindo a listagem de requisitos, escolha de entidades e seus derivados (Atributos, chaves e relações), até à potencial extensibilidade da plataforma idealizada neste trabalho.

Inserido nas temáticas da unidade curricular de Bases de Dados encontrámos algumas dificuldades na comparação de entidades com entidades fracas que, consoante o caso, deveriam ser consideradas atributos. Este foi o caso da entidade Biografia, que no término, acabou por ser considerada atributo multivalor por validação com o próprio docente.

Outra grande preocupação e, de nossa opinião, um dos pontos mais fortes do nosso trabalho cingiu – se na introdução de entidades que nos permitissem generalizar o problema em questão e ao mesmo tempo concedessem a possibilidade de incluir não só em quantidade mas também em detalhe toda a informação referente aos variados tipos e géneros de filmes existentes no mercado (Animação, documentários, natureza …).

Problemas que surgiram durante o desenvolvimento deste projeto apontam para o facto de que inicialmente considerámos a utilização de uma entidade para referir a existência de Personagens na nossa BD, facto que no fim veio a introduzir problemas quando, por exemplo, necessitávamos de saber quais os atores que entraram num dado filme. Utilizando a nossa presente implementação, não é possível obter de forma correta a lista de atores que entraram num filme através das suas respetivas personagens.

Após verificação com o docente, concluímos que uma solução para este problema seria a introdução de uma relação ternária entre Filme, Ator e Personagem, algo que não se verifica no momento, pois Ator relaciona-se com Personagem e Filme também se relaciona com Personagem, mas nada relaciona a entidade Ator com Filme.

Desta forma já seria possível listar os atores que entraram num filme sem nenhum inconveniente. Acabámos por não implementar este caso pois implicaria alterações profundas no projeto, o que não seria viável com o tempo disponível para a conclusão do mesmo no momento em que este erro foi descoberto.

Por último, acabamos por mencionar a extensibilidade que a nossa plataforma poderia introduzir com um desenvolvimento futuro. Por exemplo, a implementação complementar de uma rede social, que seria permitida devido à existência desde já da entidade Utilizador e possibilidade de este mesmo ter um Perfil. A própria existência destes últimos elementos permitiria uma partilha de informação pessoal em rede juntamente com a possível troca de opiniões disponibilizada pelas classificações, visualizações e comentários deixados pelos utilizadores.

Terminamos finalmente referindo a análise positiva que este grupo atribuiu a este projeto, tendo sempre em conta a possível alteração de pequenos detalhes que permitiria encaixar novas possibilidades de desenvolvimento e corrigir alguns erros de idealização e mesmo de modulação.

Bibliografia

[1] Thomas M. Connolly and Carolyn E. Begg, *Database Systems: A practical approach to Design, Implementation and Management*, 6th Ed. New York, USA: Pearson, 2014.

Lista de Siglas e Acrónimos

BD Bases de Dados

SGBD Sistema de Gestão de Base de Dados

SQL Structured Query Language

Anexos

I. Diagrama de Chen

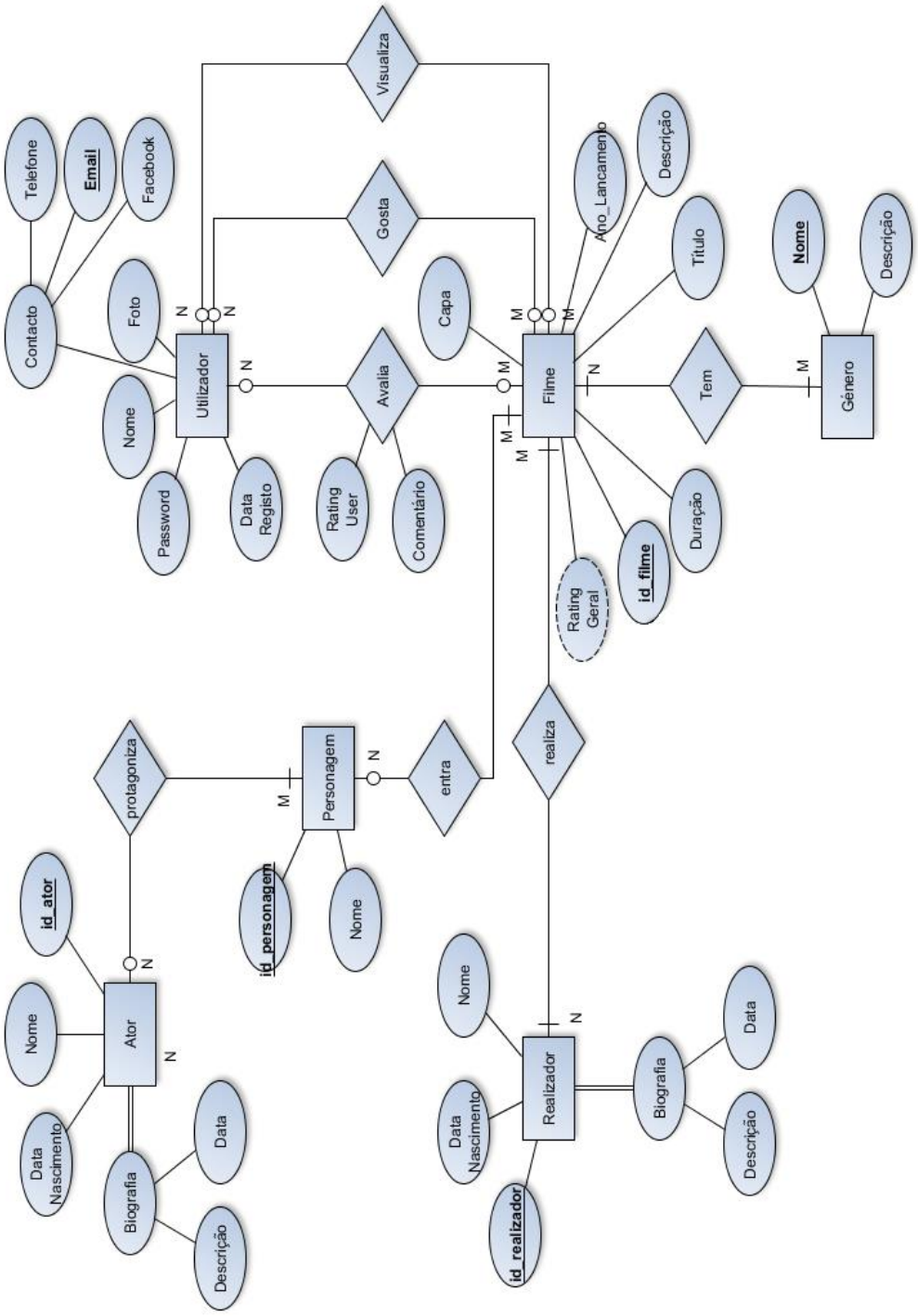


Figura 44 – Modelo Conceptual – Diagrama de Chen

II. Modelo Lógico

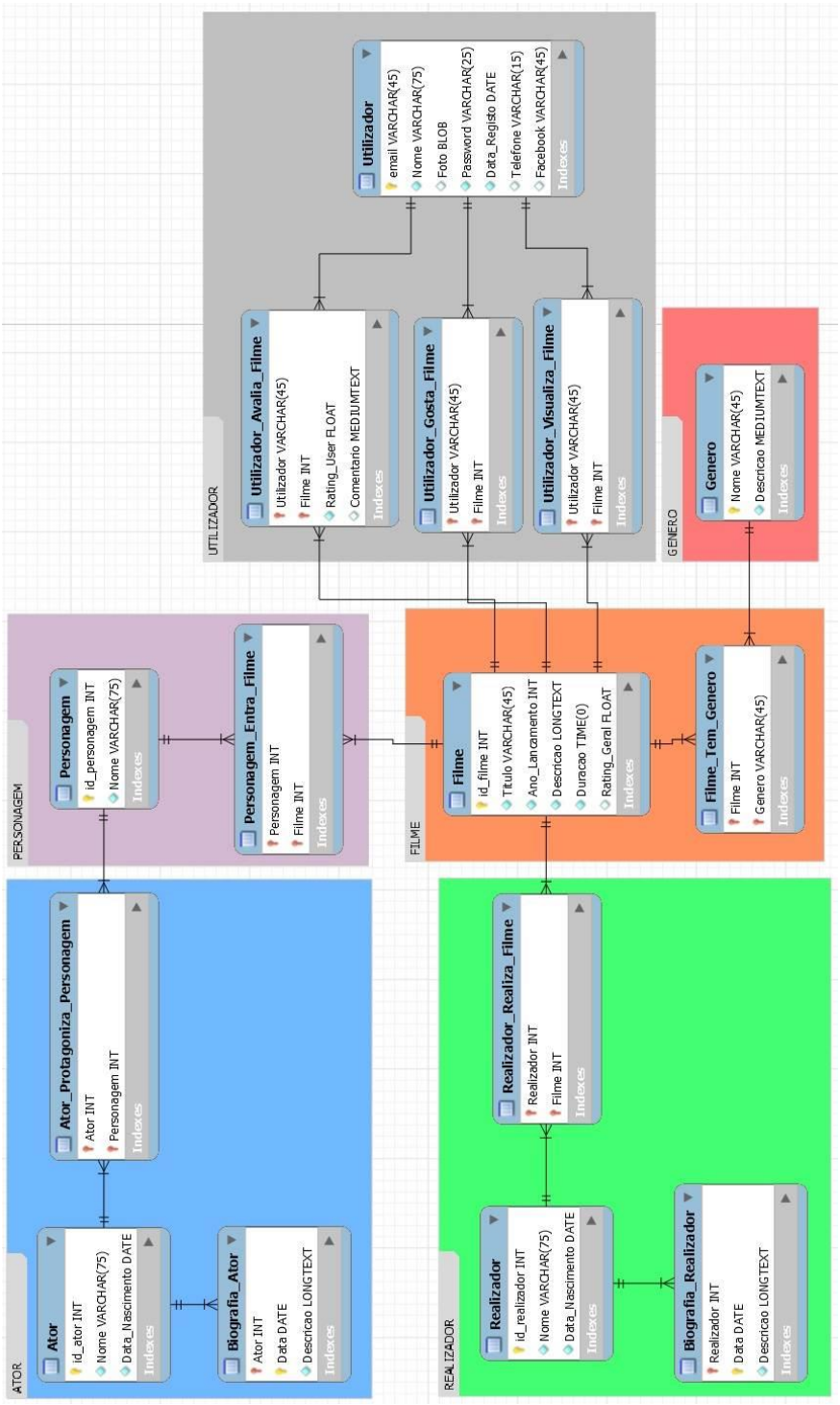


Figura 45 – Modelo Lógico

III. Procedimentos SQL

```
-- -----------------------------------------------------
-- Identificar filmes de um dado ano
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Filtra_Ano(IN Ano INT)
BEGIN
  SELECT Titulo
    FROM Filme
    WHERE Ano_Lancamento = Ano;
END
|
DELIMITER ;
```

Figura 46 – Identificar Filmes de um dado ano

```
-- -----------------------------------------------------
-- Identificar actores que participaram em um dado filme
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Atores_No_Filme(IN filme INT)
BEGIN
  SELECT A.Nome AS 'Nome'
    FROM Ator AS A
      INNER JOIN Ator_Protagoniza_Personagem AS APP
        ON A.id_ator = APP.Ator
      INNER JOIN Personagem AS P
        ON P.id_personagem = APP.Personagem
      INNER JOIN Personagem_Entra_Filme AS PEF
        ON PEF.Personagem = P.id_personagem
    WHERE PEF.Filme = filme;
END
|
DELIMITER ;
```

Figura 47 – Identificar atores que participaram num dado filme

```
-- -----------------------------------------------------
-- Identificar filmes de um dado género
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Filmes_De_Genero(IN genero VARCHAR(45))
BEGIN
  SELECT F.Titulo AS 'Titulo'
    FROM Filme AS F
      INNER JOIN Filme_Tem_Genero AS FTG
        ON FTG.Filme = F.id_filme
    WHERE FTG.Genero = genero;
END
|
DELIMITER ;
```

Figura 48 – Identificar filmes de um dado género

```
-- -----------------------------------------------------
-- Identificar top X filmes em termos de rating
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Top_Filmes(IN top INT)
BEGIN
  SET @Ranking = 0;

  SELECT @Ranking := @Ranking +1 AS 'Posição' , Q.Titulo, Q.Rating_Geral
    FROM (SELECT Titulo, Rating_Geral
            FROM Filme
            ORDER BY Rating_Geral DESC
            LIMIT top) AS Q;
END
|
DELIMITER ;
```

Figura 49 – Identificar top X filmes em termos de *rating*

```
-- -----------------------------------------------------
-- Identificar filmes em que um dado ator entrou
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Filmes_Ator(IN ator INT)
BEGIN
  SELECT F.Titulo AS 'Titulo'
    FROM Ator_Protagoniza_Personagem AS APP
      INNER JOIN Personagem_Entra_Filme AS PEF
        ON PEF.Personagem = APP.Personagem
      INNER JOIN Filme AS F
        ON F.id_filme = PEF.Filme
    WHERE APP.Ator = ator;
END
|
DELIMITER ;
```

Figura 50 – Identificar filmes em que um dado ator entrou

```
-- ------------------------------------------------------
-- Identificar filmes de um dado realizador
-- ------------------------------------------------------
DELIMITER |
CREATE PROCEDURE Filmes_Realizador(IN realizador INT)
BEGIN
  SELECT F.Titulo AS 'Título'
    FROM Filme AS F
      INNER JOIN Realizador_Realiza_Filme AS RRF
        ON RRF.Filme = F.id_filme
    WHERE RRF.Realizador = realizador;
END
|
DELIMITER ;
```

Figura 51 – Identificar filmes de um dado realizador

```
-- ------------------------------------------------------
-- Identificar filmes por intervalos de duração
-- ------------------------------------------------------
DELIMITER |
CREATE PROCEDURE Filmes_Duracao(IN duracaoIni TIME, IN duracaoFim TIME)
BEGIN
  SELECT Titulo
    FROM Filme
    WHERE TIME_TO_SEC(Duracao) > TIME_TO_SEC(duracaoIni)
      AND TIME_TO_SEC(Duracao) < TIME_TO_SEC(duracaoFim);
END
|
DELIMITER ;
```

Figura 52 – Identificar filmes por intervalos de duração

```
-- ------------------------------------------------------
-- Identificar atores que protagonizam uma dada personagem
-- ------------------------------------------------------
DELIMITER |
CREATE PROCEDURE Atores_Personagem(IN personagem INT)
BEGIN
  SELECT A.Nome
    FROM Ator AS A
      INNER JOIN Ator_Protagoniza_Personagem AS APP
        ON A.id_ator = APP.Ator
    WHERE APP.Personagem = personagem;
END
|
DELIMITER ;
```

Figura 53 – Identificar atores que protagonizam uma dada personagem

```
-- -----------------------------------------------------
-- Atualizar a informação de um Utiizador
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Atualiza_Utilizador
    (IN mail VARCHAR(45),IN nome VARCHAR(75),
    IN pic BLOB,IN pass VARCHAR(25),IN pNumber VARCHAR(15),IN fb VARCHAR(45))
BEGIN
  UPDATE Utilizador AS U
    SET U.Nome = nome,
        U.Foto = pic,
        U.Password = pass,
        U.Telefone = pNumber,
        U.Facebook = fb
    WHERE U.email = mail;
END
|
DELIMITER ;
```

Figura 54 – Atualizar a informação de um Utilizador

```
-- -----------------------------------------------------
-- Inserir um acontecimento na biografia de um ator
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Acontecimento_Ator(IN actor INT,IN quando DATE,IN descr LONGTEXT)
BEGIN
  INSERT INTO Biografia_Ator (Ator,Data,Descricao)
    VALUES
    (actor,quando,descr);
END
|
DELIMITER ;
```

Figura 55 – Inserir um acontecimento na biografia de um ator

```
-- -----------------------------------------------------
-- Inserir um acontecimento na biografia de um realizador
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Acontecimento_Realizador(IN realizador INT,IN quando DATE,IN descr LONGTEXT)
BEGIN
  INSERT INTO Biografia_Realizador(Realizador,Data,Descricao)
    VALUES
    (realizador,quando,descr);
END
|
DELIMITER ;
```

Figura 56 – Inserir um acontecimento na biografia de um realizador

```
-- -----------------------------------------------------
-- Avaliar filme
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Avalia_Filme(IN utilizador VARCHAR(45), IN filme INT, IN avaliacao FLOAT)
BEGIN
  INSERT INTO Utilizador_Avalia_Filme (Utilizador,Filme,Rating_User)
    VALUES
    (utilizador,filme,avaliacao)
    ON DUPLICATE KEY
      UPDATE Rating_User = avaliacao;
END
|
DELIMITER ;
```

Figura 57 – Avaliar filme

```
-- -----------------------------------------------------
-- Adicionar um filme aos favoritos
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Adiciona_Favorito(IN utilizador VARCHAR(45), IN filme INT)
BEGIN
  INSERT INTO Utilizador_Gosta_Filme (Utilizador,Filme)
    VALUES
    (utilizador,filme)
    ON DUPLICATE KEY
      UPDATE Utilizador = utilizador;
END
|
DELIMITER ;
```

Figura 58 – Adicionar um filme aos favoritos

```
-- -----------------------------------------------------
-- Indicar que o filme foi visualizado
-- -----------------------------------------------------
DELIMITER |
CREATE PROCEDURE Adiciona_WList(IN utilizador VARCHAR(45),IN filme INT)
BEGIN
  INSERT INTO Utilizador_Visualiza_Filme (Utilizador,Filme)
    VALUES
    (utilizador,filme)
    ON DUPLICATE KEY
      UPDATE Utilizador = utilizador;
END
|
DELIMITER ;
```

Figura 59 – Indicar que o filme foi visualizado