Universidade do Minho

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Unidade Curricular de

Bases de Dados

Ano Lectivo de 2015/2016

Base de Dados Cinematográfica

Adriano Dias Teixeira – a67663

João Simões Farinha – a69302

Stéphane Fernandes – a67681

Vítor Hugo de Castro Rodrigues – a67703

Janeiro, 2016

Data de Recepção	
Responsável	
Avaliação	
Observações	

Base de Dados Cinematográfica

Adriano Dias Teixeira – a67663

João Simões Farinha – a69302

Stéphane Fernandes – a67681

Vítor Hugo de Castro Rodrigues – a67703

Janeiro, 2016

Resumo

Este projeto consiste na elaboração de uma base de dados que permita ao utilizador consultar informação principalmente sobre filmes e séries. Esta base de dados terá informação sobre atores/realizadores de filmes ou séries, assim como os prémios que cada série, filme, ator, realizador recebeu. Decidimos realizar este trabalho sobre este tema porque é uma área que nos interessa.

Este trabalho será baseado num site bastante conhecido “IMDb”, porque achamos que é um site bastante organizado e usado por pessoas com o intuito de aprender mais sobre filmes e séries, e essa é também uma das nossas motivações na realização deste projeto.

Nesta fase inicial do projeto, este relatório contém vários passos, desde a análise de requisitos até à modelação conceptual, no final, quando estiver concluído o projeto, este relatório irá conter ainda a modelação lógica assim como a física.

Neste relatório representaremos as várias etapas do desenvolvimento deste projeto incluindo modulação conceptual, lógica e física e a análise da base de dados desenvolvida.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados.

Palavras-Chave: Modelação conceptual, filmes, séries.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Neste capítulo é feita uma introdução ao projeto a realizar, sendo apresentada a contextualização do problema, o caso de estudo, a motivação e objetivos e por fim a estrutura do relatório.

1.1. Contextualização

Pense em todos os filmes, séries, atores, realizadores e outros intervenientes do mundo da sétima arte e pense no quão bom seria ter um sistema que permitisse reunir, organizar e providenciar todas estas informações, foi com este pensamento que sites como o "IMDb" e "Metascore" foram criados, partindo deste mesmo pensamento tem-se como objetivo criar um sistema de base de dados idênticos.

1.2. Apresentação do Caso de Estudo

Atualmente, a indústria do cinema e televisão é das principais líderes a nível do entretenimento, por isso existem muitos serviços que funcionam em torno deste setor, como por exemplo os serviços de divulgação de conteúdo como o Netflix e websites como Hulu. Por isso surge a necessidade de armazenar uma grande quantidade de dados e organizá-los de maneira a torná-los rapidamente acessíveis e de fácil gestão.

Cabe à equipa deste projeto desenvolver uma base de dados capaz de satisfazer algumas das necessidades de armazenamento e gestão de dados apresentadas por plataformas como o IMDb, que atualmente é das mais utilizadas para pesquisa de informação deste domínio.

1.3. Motivação e Objetivos

Dado o tamanho e variedade de filmes e series existentes torna-se necessário encontrar uma forma eficiente e estável de guardar e aceder a toda esta informação. Para isso pretendemos criar uma base de dados consistente capaz de guardar não só os dados referentes a séries e filmes, como outros a estes associados como atores, preenchendo para isso tantos os requisitos expostos neste relatório como também aqueles exigidos pela unidade curricular em que este projeto se integra.

1.4. Estrutura do Relatório

O Capítulo 1 traz-nos uma introdução ao problema, indicando a sua contextualização bem como a apresentação do caso de estudo.

No Capítulo 2 é apresentada a análise de requisitos realizada para a elaboração deste trabalho.

No Capítulo 3 desenvolvemos o modelo conceptual da base de dados.
No Capítulo 4 desenvolvemos o modelo lógico da base de dados.
No Capítulo 5 desenvolvemos o modelo físico da base de dados.

Por fim é realizada uma apreciação crítica sobre o trabalho, dando especial destaque aos pontos fortes e fracos do mesmo.

2. Análise de Requisitos

2.1. Objetivos da base de dados

Manter (inserir, atualizar, apagar) informação sobre filmes.
Manter (inserir, atualizar, apagar) informação sobre séries.
Manter (inserir, atualizar, apagar) informação sobre géneros

Manter (inserir, atualizar, apagar) informação sobre pessoas.
Manter (inserir, atualizar, apagar) informação sobre prémios.
Manter (inserir, atualizar, apagar) informação sobre utilizadores.
Manter (inserir, atualizar, apagar) informação sobre críticas.

Efetuar pesquisas de filmes.
Efetuar pesquisas de séries.
Efetuar pesquisas de géneros.
Efetuar pesquisas de pessoas.
Efetuar pesquisas de prémios.
Efetuar pesquisas de utilizadores.
Efetuar pesquisas de críticas.

2.2. Requisitos de Dados

Nota: Dado que só existe um tipo de utilizador (Administrador), foi criada uma única vista de utilizado.

Requisitos de dados

- A base de dados deve manter para cada filme o seu título, duração, data de estreita, faixa etária a que é destinado, o seu género(s) e a lista de prémios que ganhou, sendo para cada prémio registado o seu nome e ano de atribuição.

- Com cada filme estão envolvidas pessoas, que podem ser atores, realizadores ou simultaneamente atores e realizadores. É por isso, necessário que a base de dados mantenha registado para cada pessoa o seu nome, data de nascimento, idade, lista de prémios que ganhou, um valor que indica se a pessoa é um realizador(a) e outro que indica se a pessoa é um ator/atriz.

- Um filme pode ter associado a si uma ou várias críticas feitas por utilizadores.

- A cada crítica feita deve estar associado o seu conteúdo e o número de pontos, de 1 a 10, atribuídos à obra

- Um utilizador pode fazer uma ou várias críticas e para este, a base de dados deverá ter registado um nome de utilizador e uma palavra-passe.

- Além de filmes a base de dados deve também manter registos de séries, e estas devem ter a si associadas um nome, a faixa etária a que é indicada, o seu género(s), a lista de prémios que recebeu, o número de episódios e temporadas e as suas datas de início e fim (se for o caso).
- Da mesma maneira que os filmes, cada série tem a si associada a um conjunto de críticas feitas por utilizadores e um conjunto de pessoas nelas envolvidas.

3. Modelo Conceptual

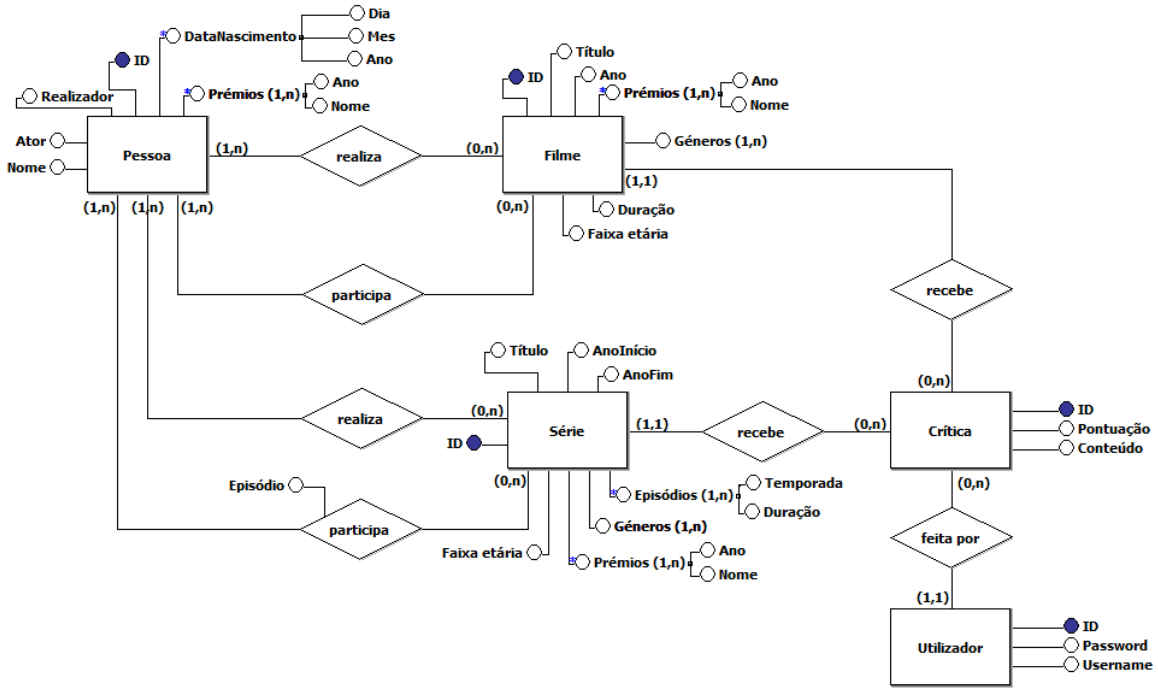

Figura 1 - Modelo Conceptual

O modelo conceptual do sistema aparece depois da análise de requisitos e pode ser considerada a primeira etapa de construção do sistema, a nível de modelo.

Este modelo serve principalmente para identificar os vários relacionamentos entre as várias entidades do sistema, assim como os atributos associados.

3.1 Identificação e Caraterização de Entidades

A primeira etapa no desenvolvimento do modelo conceptual é identificar os principais objetos que têm interesse para os utilizadores, esses objetos serão as entidades do nosso modelo.

A seguinte tabela corresponde ao dicionário de dados de entidades.

Tabela 1 - Dicionário de dados de entidades

Nome	Descrição	Sinónimos	Ocorrência
Filme	Termo usado para descrever os filmes mantidos pela base de dados	Obra cinematográfica	Cada filme é realizado e/ou protagonizado por pessoas
Serie	Termo usado para descrever as séries mantidas pela base de dados	Obra televisiva	Cada série é realizada e/ou protagonizada por pessoas
Pessoa	Termo usado para descrever os intervenientes nos filmes e séries	Ator / Realizador	Uma pessoa realiza ou participa num filme ou série
Utilizador	Termo utilizado para descrever quem usufrui dos serviços prestados pelo sistema	Utente	Um utilizador avalia um filme ou uma série
Critica	Termo usado para descrever a opinião que um utilizador tem de um filme ou série	Opinião	Um utilizador faz uma crítica a um filme ou uma série

3.2 Identificação e Caracterização de Relacionamentos

Terminado o processo de identificação de entidades, o passo seguinte consiste em identificar todos os relacionamentos entre as entidades identificadas.

Foram determinados os seguintes relacionamentos entre entidades:

Pessoa e Filme:

- Uma Pessoa pode ser um ator, realizador ou ambos.
- Um ator participa em um ou vários Filmes, e num Filme participam um ou vários atores. Isto traduz-se num relacionamento entre Pessoa e Filme, designado por “participa”, com cardinalidade de N (opcional) para N (opcional).
- Um realizador realiza um ou vários Filmes, e um Filme é realizado por um ou vários realizadores. Isto traduz-se num relacionamento entre Pessoa e filme, designado por “realiza”, com cardinalidade de N (opcional) para N (opcional).
- Existem assim, dois relacionamentos entre as entidades Pessoa e Filme.

Pessoa e Série:

- Os relacionamentos entre estas entidades são iguais aos relacionamentos entre as entidades Pessoa e Filme.

Filme e Crítica:

- Um Filme pode ser avaliado por várias Criticas, e uma Critica classifica um Filme. Isto traduz-se num relacionamento entre Filme e Critica, designado por “avaliado por”, com cardinalidade de 1 (obrigatório) para N (opcional).

Série e Crítica:

- O relacionamento entre estas entidades é igual ao relacionamento entre as entidades Filme e Crítica.

Utilizador e Crítica:

- Um Utilizador pode fazer várias Criticas, e uma Critica é feita por um Utilizador. Isto traduz-se num relacionamento entre Utilizador e Critica, designado por “faz”, com uma cardinalidade de 1 (obrigatório) para N (opcional).

Toda esta informação encontra-se resumida no seguinte dicionário de dados de relacionamentos.

Tabela 2 - Dicionário de dados de relacionamentos de entidades

Relacionamento	Entidade	Multiplicidade	Entidade
feita por	Crítica	(0,n)--(1,1)	Utilizador
tem	Serie	(1,1)--(0,n)	Crítica
tem	Filme	(1,1)--(0,n)	Crítica
realiza	Pessoa	(1,n)--(0,n)	Filme
participa	Pessoa	(1,n)--(0,n)	Série
realiza	Pessoa	(1,n)--(0,n)	Filme
participa	Pessoa	(1,n)--(0,n)	Filme

3.3 Identificação e Caracterização dos atributos das entidades

Terminada a identificação e caracterização de entidades, o passo seguinte é identificar e caracterizar os atributos que as descrevem.

Essa informação é apresentada no seguinte dicionário de dados, que corresponde ao dicionário de dados de atributos de entidades.

Tabela 3 - Dicionário de dados de atributos de entidades

Entidade	Atributo	Descrição	Domínio	Nulo	Multi-Valor	Derivado
Filme	ID		Inteiro	Não	Não	Não
	Título	Titulo da série	String	Não	Não	Não
	Data	Data do filme	Data	Não	Não	Não
	Duração	Duração do filme	Inteiro	Não	Não	Não
	Género	Género do filme	String	Sim	Sim	Não
	Faixa etária	Faixa etária do filme	String	Sim	Não	Não
	Prémio	Prémio que o filme poderá ter recebido		Sim	Sim	Não
	Nome	Nome do prémio	String	Não	Não	Não
	Ano	Ano da entrega do prémio	Inteiro	Não	Não	Não
Serie	ID		Inteiro	Não	Não	Não
	Nome	Nome da série	String	Não	Não	Não
	Género	Género da série	String	Sim	Sim	Não
	Faixa etária	Faixa etária da série	String	Sim	Não	Não
	Episódios	Conjunto de episódios da série		Não	Não	Não
	Temporada	Teporada a que o episódio pertence	Inteiro	Não	Não	Não
	Duração	Duração do episódio	Inteiro	Não	Não	Não
	AnoInício	Ano em que a série estreou	Inteiro	Não	Não	Não
	AnoFim	Ano em que a série terminou	Inteiro	Sim	Não	Não

	Prémio	Prémio que a série poderá ter recebido		Sim	Sim	Não
	Nome	Nome do prémio	String	Não	Não	Não
	Ano	Ano da entrega do prémio	Inteiro	Não	Não	Não

Pessoa	ID		Inteiro	Não	Não	Não
	Nome	Nome do ator/realizador	String	Não	Não	Não
	Data Nasc	Data de nascimento	Data	Sim	Não	Não
	Realizador	Booleano que indica se a pessoa é realizador	Booleano	Não	Não	Não
	Ator	Booleano que indica se a pessoa é realizador	Booleano	Não	Não	Não
Utilizador	ID		Inteiro			
	Username	Nome de utilizador do sistema	String	Não	Não	Não
	Password	Senha do utilizador do sistema	String	Não	Não	Não
Critica	Conteúdo	Corpo da crítica	Texto	Sim	Não	Não
	Pontuação	Valor no intervalo 0-10	Inteiro (1-5)	Não	Não	Não

3.4 Definição de Chaves Candidatas, Primárias e Alternadas

Depois de identificar para cada entidade os atributos que as descrevem, é necessário identificar quais desses atributos são chaves candidatas, alternadas e qual delas é a primária.

Uma chave candidata é o atributo, ou menor conjunto possível de atributos que identificam inequivocamente a ocorrência de uma entidade, dessas chaves é escolhida uma que se designa por chave primária. As chaves candidatas que não são escolhidas para chave primárias são chamadas de chaves alternadas.

A seguir indicam-se, para cada entidade, quais as suas chaves candidatas, qual a chave primária e a razão para a sua escolha.

Filme

Chaves candidatas: ID

Chave primária: A chave primária escolhida é a chave candidata ID uma vez que é a única chave candidata.

Serie

Chaves candidatas: ID
Chave primária: A chave primária escolhida é a chave candidata ID uma vez que é a única chave candidata.

Pessoa

Chaves candidatas: ID

Chave primária: A chave primária escolhida é a chave candidata ID uma vez que é a única chave candidata.

Utilizador

Chaves candidatas: Username

Chave primária: A chave primária escolhida é a chave candidata Username uma vez que é a única chave candidata.

Crítica

Chaves candidatas: ID

Chave primária: A chave primária escolhida é a chave candidata ID uma vez que é a única chave candidata.

4. Modelo Lógico

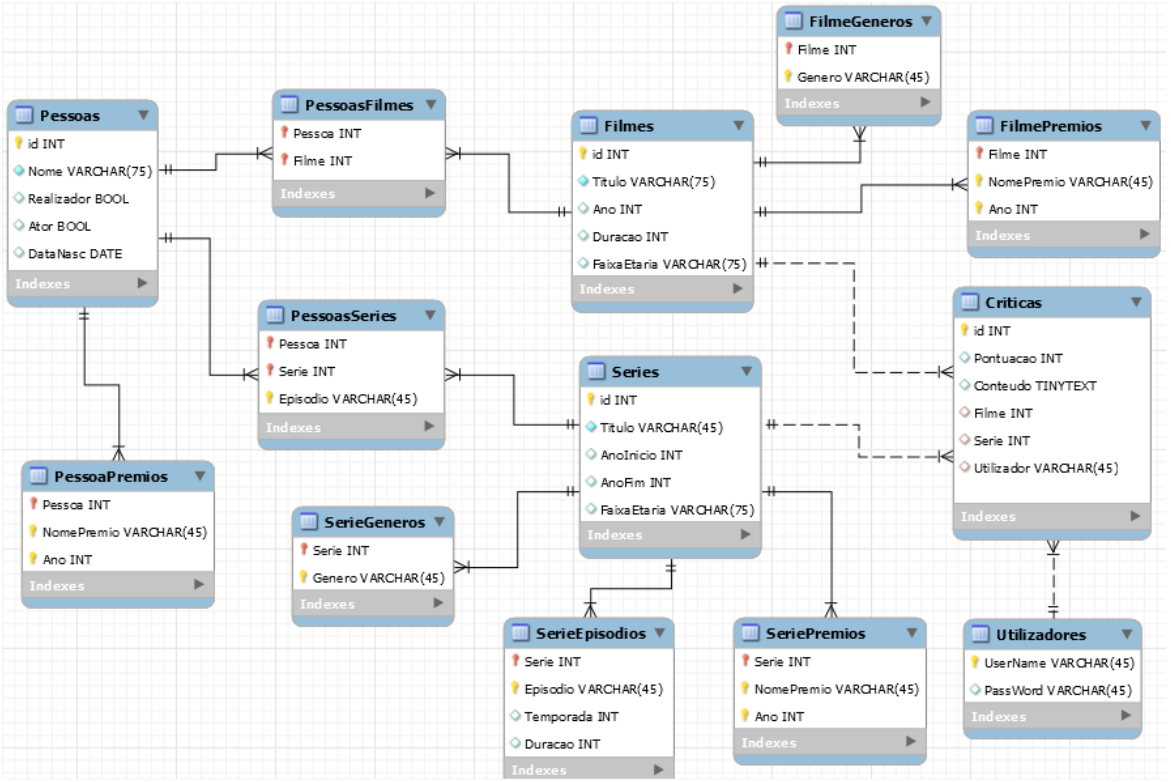

Figura 2 - Modelo Lógico

Terminada a fase de modelação conceptual, passamos à fase de modelação lógica, que tem como objetivo traduzir o modelo conceptual anteriormente desenvolvido para um modelo lógico e posteriormente validá-lo de maneira a verificar que o modelo está estruturalmente correto e que seja capaz de suportar os requisitos definidos.

Isto será conseguido através da execução de cada um dos passos a seguir apresentados.

4.1. Derivação das relações para o modelo lógico

Neste passo o objetivo é derivar, para o modelo lógico, as relações que representam as entidades, relacionamentos e atributos do modelo conceptual. Foi utilizada uma Database Definition Language para descrever a composição das relações considerando as seguintes estruturas que podem ocorrer num modelo conceptual:

Entidades fortes
Entidades fracas
Relacionamentos binários um para muitos (1:*)

Relacionamentos binários um para um (1:1)
Relacionamentos recursivos um para um (1:1)
Relacionamentos subclasse/superclasse
Relacionamentos muitos para muitos (*:*)
Relacionamentos complexos
Atributos multivalor

Entidades fortes

Pessoa (ID, Nome, Realizador, Ator, Dia, Mês, Ano, PrémioNome, PrémioAno)
Chave Primária: ID

Filme (ID, Titulo, Ano, Duração, FaixaEtária, PrémioNome, PrémioAno)
Chave Primária: ID

Série (ID, Título, AnoInício, AnoFim, FaixaEtária, PrémioNome, PrémioAno)
Chave Primária: ID

Crítica (ID, Pontuação, Conteúdo)
Chave Primária: ID

Utilizador (Username, Password)
Chave Primária: Username

Entidades fracas

Este tipo de entidade é inexistente no modelo desenvolvido uma vez que todas as entidades presentes podem ser unicamente identificadas usando apenas atributos próprios.

Relacionamentos binários um-para-muitos

Neste tipo de relacionamento, a entidade com multiplicidade um é designada por entidade pai, e a entidade com multiplicidade “muitos” por entidade filho.

Para representar este tipo de relacionamento, adiciona-se à entidade filho uma cópia da chave primária da entidade pai, que se designa por chave estrangeira na entidade filho.

Crítica (ID, Pontuação, Conteúdo, Filme, Série, Utilizador)
Chave Primária: ID
Chave Estrangeira: Utilizador referente a Utilizador (ID)
Chave Estrangeira: Filme referente a Filme (ID)
Chave Estrangeira: Série referente a Série (ID)

Relacionamentos binários um-para-um

O modelo conceptual desenvolvido não contém nenhum relacionamento deste tipo.

Relacionamentos recursivos um-para-um

O modelo conceptual desenvolvido não contém nenhum relacionamento deste tipo.

Relacionamentos subclasse/superclasse

O modelo conceptual desenvolvido não contém nenhum relacionamento deste tipo.

Relacionamentos binários muitos-para-muitos

Para representar este tipo de relacionamento é criada uma nova tabela que contém todos os atributos desse relacionamento e uma cópia da chave primária de cada entidade envolvida no relacionamento.

A chave primária desta relação será formada pelo seu par de chaves estrangeiras e possivelmente em conjunto com um ou mais atributos do relacionamento.

Pessoa - Filme
Origina:
PessoasFilmes (Pessoa, Filme)
Chave Primária: (Pessoa, Filme)
Chave Estrangeira: Pessoa referente a Pessoa (ID)
Chave Estrangeira: Filme referente a Filme (ID)

Pessoa - Série
Origina:
PessoasSéries (Pessoa, Série, Episódio)
Chave Primária: (Pessoa, Série, Episódio)
Chave Estrangeira: Pessoa referente a Pessoa (ID)
Chave Estrangeira: Série referente a Série (ID)

Relacionamentos complexos

O modelo conceptual desenvolvido não contém nenhum relacionamento deste tipo.

Atributos multi-valor

Para representar estes atributos é criada, para cada um deles, uma tabela que irá conter o atributo e a chave primária da entidade correspondente para ser usada como chave estrangeira.

O atributo multi-valor **género** em **Filme** origina:

FilmeGéneros (Género, Filme)
Chave Primária: (Género, Filme)
Chave Estrangeira: Filme referente a Filme (ID)

O atributo multi-valor **prémios** em **Filme** origina:

FilmePrémios (NomePrémio, Ano, Filme)
Chave Primária: (NomePrémio, Ano, Filme)
Chave Estrangeira: Filme referente a Filme (ID)

O atributo multi-valor **género** em **Série** origina:

SérieGéneros (Género, Série)
Chave Primária: (Género, Série)
Chave Estrangeira: Série referente a Série (ID)

O atributo multi-valor **prémios** em **Série** origina:

SériePrémios (NomePrémio, Ano, Série)
Chave Primária: (NomePrémio, Ano, Série)
Chave Estrangeira: Série referente a Série (ID)

O atributo multi-valor **espisódios** em **Série** origina:

SérieEpisódios (Episódio, Temporada, Duração, Série)
Chave Primária: (Episódio, Temporada, Série)
Chave Estrangeira: Série referente a Série (ID)

O atributo multi-valor **prémios** em **Pessoa** origina:

PessoaPrémios (NomePrémio, Ano, Pessoa)
Chave Primária: (NomePrémio, Ano, Pessoa)
Chave Estrangeira: Pessoa referente a Pessoa (ID)

4.2. Validação das relações através de normalização

No passo anterior derivámos um conjunto de relações para representar o modelo conceptual desenvolvido anteriormente. Neste passo validamos os grupos de atributos em cada relação através das regras de normalização.

A normalização é uma técnica formal de análise de relações baseada nas suas chaves primárias e dependências funcionais. Essa técnica envolve uma série de regras que podem ser usadas para testar relações individuais para que a base de dados possa ser normalizada até um dado grau.

O processo de normalização está dividido em cinco fases/graus de normalização designadas de formas normais (1FN, 2FN, 3FN, 4FN e 5FN).

Quando alguma regra de uma das formas normais não é respeitada, a relação que viola essa regra deve ser decomposta em relações que individualmente a respeitem.

Uma vez que para que para que sejam evitadas anomalias de atualização, é recomendado que se proceda pelo menos até à terceira forma normal (3FN) e dado que as duas últimas formas normais lidam com situações que são muito raras, vamos descrever apenas as primeiras três formas normais.

Começando por explicar o que é uma dependência funcional, considere-se seguinte relação:

Filme (ID, Titulo, Ano, Duração, FaixaEtária)

Nesta relação, se for conhecido o valor do atributo ID, é possível determinar o valor do atributo Ano, uma vez que para cada valor do atributo ID existe um e um só valor para o atributo Ano, e por essa razão Ano é funcionalmente dependente de ID, sendo **ID -> Ano** a notação utilizada para representar a dependência.

1FN - 1ª Forma Normal

Exemplo de uma relação que não está na 1FN:

Filme (ID, Titulo, Ano, PrémioAno, PrémioNome, Género1, Género2, Género3, Duração, FaixaEtária)

Por exemplo, (1, X-Men Apocalypse, 2016, 2017, Melhor filme, Ação, Aventura, Fantasia, 132, PG-13)

Neste exemplo o atributo **Géneros** contém mais que um valor, neste caso três (Ação, Aventura, Fantasia), por esse motivo a relação não se encontra na 1FN, encontra-se desnormalizada.

Para colocar a relação na 1FN é necessário remover dela o atributo (**Géneros**) que se repete e para isso é necessária a criação de uma nova tabela **FilmeGéneros (Género, Filme)** onde o atributo Filme é a chave estrangeira que referencia a tabela Filme, assim quando um Filme tem N géneros, são inseridos nesta tabela N registos correspondentes a esse filme. Assim ambas as relações passam a estar na 1FN.

O mesmo processo foi feito para passar as relações **Pessoa** e **Filme** para a 1FN sendo que as restantes tabelas já se encontravam na 1FN.

2FN - 2ª Forma Normal

Considerando a seguinte relação e dependências funcionais:

R = {a1, a2, a3, a4, a5, a6}

a1, a2 -> a4, a5
a2 -> a5

Chave primária: (a1, a2)

Neste exemplo verifica-se que o atributo a5 depende apenas de um dos atributos da chave primária, o a2, e por essa razão diz-se que a5 é parcialmente dependente da chave primária (a1,a2).

Para passar esta relação para 2FN é necessário remover dela o atributo a5, o que depende parcialmente da chave primária, e colocá-lo numa nova relação, ficando a relação R com a seguinte forma R = {a1, a2, a3, a4, a6} e a nova relação S = {a2, a5}. Desta forma ambas as relações ficam na 2FN.

Ao transformar as relações do nosso modelo na 1FN estas passaram diretamente a estar na 2FN.

3FN - 3ª Forma Normal

Considerando a seguinte relação e dependências funcionais:

R = {a1, a2, a3, a4, a5}

a1 -> a3, a4
a3 -> a5

Chave primária: a1

Neste exemplo verifica-se que o atributo a5 depende de a3, que por sua vez depende da chave primária, e por essa razão a5 é transitivamente dependente de a3.

Para transformar esta relação na 3FN, é necessário remover dela a dependência transitiva. Isto consegue-se removendo da relação o atributo transitivamente dependente (a5) e colocando-o numa nova relação.

Assim a relação R fica da forma R = {a1, a2, a3, a4} e a nova relação S = {a3, a5}. Desta maneira ambas as relações ficam na 3FN.

Ao transformar as relações do nosso modelo na 2FN estas passaram diretamente a estar na 3FN.

4.3. Validação das relações através das transações dos utilizadores

Terminada a derivação do Modelo Lógico, é importante que a partir deste modelo seja possível obter a resposta a qualquer pergunta que um utilizador possa colocar. Foi então testada se as transações requisitadas pelo utilizador são suportadas pelo Modelo Lógico.

a) Listar os nomes de filmes que estrearam num dado ano (por exemplo: 2014).
b) Listar o nome do ator/atores com o maior número de prémios.
c) Listar os nomes dos atores que participaram num dado filme.
d) Determinar idade de um dado autor.
e) Determinar se um determinado ator participou num dado episódio de uma série.
f) Listar os episódios de uma série em que determinado ator participou.

Descrição das transações:

a) Listar os nomes de filmes que estrearam num dado ano (por exemplo: 2014).

A partir da tabela Filme começamos por selecionar todos os filmes tal que o valor da coluna Ano seja igual ao ano em questão e por fim identificamos os filmes selecionados através da coluna Titulo.

b) Listar o nome do ator/atores com o maior número de prémios.

Começamos por juntar as tabelas Pessoa e PessoaPrémios e, a partir da tabela resultante, cria-se uma nova tabela que resulta de selecionar Pessoa.ID, Pessoa.Nome e fazer uma contagem do número de linhas agrupada por Pessoa.ID tal que Pessoa.Ator tenha o valor TRUE e atribui-se aos resultados das contagens uma coluna chamada Totais, por fim desta última tabela seleciona-se Pessoa.Nome e o valor máximo da coluna Totais, colocando esse valor a uma coluna designada por NrPrémios

c) Listar os nomes dos atores que participaram num dado filme.

Começamos por juntar a tabela Pessoa à tabela PessoasFilmes, e a tabela resultante dessa junção, à tabela Filmes, e por fim, da tabela resultante das duas junções, selecionamos Pessoa.Nome tal que Pessoa.Ator tenho o valor TRUE e Filme.Título tenha o valor igual ao título do filme em questão.

d) Determinar idade de um dado ator.

A partir da tabela Pessoa começamos por selecionar a pessoa tal que Nome seja igual ao fornecido e o valor da coluna Ator seja igual a TRUE, e por fim, a partir da mesma tabela, selecionamos o Nome e o valor da diferença entre data atual e a DataNascimento desse ator, atribuindo esse valor a uma coluna designada por Idade.

e) Determinar se um determinado ator participou num dado episódio de uma série.

Começamos por juntar a tabela Pessoa à tabela PessoasSéries, e a tabela resultante dessa junção, à tabela Séries, em seguida, sobre a tabela resultante das duas junções, selecionamos o registo tal que Pessoas.Ator tenha valor TRUE e Pessoas.Nome, Séries.Título e AtoresSéries.Episódio tenham valores iguais aos fornecidos, e por fim, sobre a tabela que resulta dessa querie faz-se uma contagem do seu número de linhas, se esse valor for 1 devolve-se TRUE e se for 0 devolve-se FALSE.

f) Listar as temporadas de uma série em que determinado ator participou.

Começamos por juntar a tabela Pessoas à tabela PessoasSéries, à tabela resultante junta-se a tabela Séries, e à resultante dessas duas junções junta-se a tabela SérieEpisódios tal que PessoasSéries.Episódio seja igual a SérieEpisódios.Episódio, por fim, sobre a tabela resultante destas três junções selecionamos as distintas temporadas (SérieEpisódios.Temporada) tal que Série.Título e Pessoas.Nome tenham valores iguais aos fornecidos e Pessoas.Ator tenha o valor TRUE.

4.4. Análise do crescimento futuro

Esta fase do desenvolvimento do Modelo Lógico serve para estudar a potencialidade de expansão de futuros desenvolvimentos. Assim, depois de todos os passos para realizar o Modelo Conceptual seguinte do Modelo Lógico e verificado cada um dos passos, conclui-se que este modelo pode ser facilmente extensível, de forma a suportar novos requisitos que possam surgir.

5. Modelo Físico

Nesta etapa do projeto é efetuada a tradução do Modelo Lógico para o Modelo Físico implementado num SGBD.

5.1 Tradução do modelo lógico para o SGBD escolhido

5.1.1 Desenho das relações base

Neste passo é apresentada a maneira de como o modelo lógico desenvolvido é traduzido para um SGBD específico, neste caso o MySQL Server. E para isso é apresentado o script na linguagem SQL utilizado para criar a estrutura da base de dados, onde se pode ver para cada relação, o domínio dos seus atributos, os seus valores por defeito, valores nulos e as suas chaves primárias e chaves estrangeiras.

```
CREATE SCHEMA IF NOT EXISTS cinemadb DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci;
USE cinemadb;
```

Relação Pessoas

```
CREATE TABLE IF NOT EXISTS cinemadb.Pessoas (
  id  INT NOT NULL AUTO_INCREMENT,
  Nome  VARCHAR(75) NOT NULL,
  Realizador  TINYINT(1) NULL,
  Ator  TINYINT(1) NULL,
  DataNasc  DATE NULL,
  PRIMARY  KEY ( id )
);
```

Relação Séries

```
CREATE TABLE IF NOT EXISTS cinemadb.Series (
  id   INT NOT NULL,
  Titulo  VARCHAR(45) NOT NULL,
  AnoInicio  INT NULL,
  AnoFim  INT NULL,
  FaixaEtaria  VARCHAR(75) NULL,
  PRIMARY KEY ( id )
);
```

Relação Filmes

```
CREATE TABLE IF NOT EXISTS cinemadb.Filmes (
   id  INT NOT NULL,
  Titulo   VARCHAR(75) NOT NULL,
  Ano  INT NULL,
  Duracao   INT NULL,
  FaixaEtaria  VARCHAR(75) NULL,
  PRIMARY KEY ( id )
);
```

Relação Utilizadores

```
CREATE TABLE IF NOT EXISTS cinemadb.Utilizadores (
  UserName  VARCHAR(45) NOT NULL,
  PassWord  VARCHAR(45) NULL,
  PRIMARY KEY ( UserName )
);
```

Relação Críticas

```
CREATE TABLE IF NOT EXISTS cinemadb.Criticas (
  id   INT NOT NULL,
  Pontuacao  INT NULL,
  Conteudo  TINYTEXT NULL,
  Filme  INT NULL,
  Serie  INT NULL,
  Utilizador` VARCHAR(45) NULL,
  PRIMARY KEY ( id ),
  INDEX Filme_idx ( Filme  ASC),
  INDEX Serie_idx ( Serie  ASC),
  INDEX Utilizador_idx ( Utilizador ASC),
  FOREIGN KEY ( Filme )
    REFERENCES  cinemadb.Filmes ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  FOREIGN KEY ( Serie )
    REFERENCES  cinemadb.Series ( id )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  FOREIGN KEY ( Utilizador )
    REFERENCES  cinemadb.Utilizadores  ( UserName )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação PessoasFilmes

```
CREATE TABLE IF NOT EXISTS cinemadb.PessoasFilmes (
  Pessoa INT NOT NULL,
  Filme INT NOT NULL,
  PRIMARY KEY ( Pessoa ,  Filme ),
  INDEX Filme_idx ( Filme  ASC),
  FOREIGN KEY ( Pessoa )
    REFERENCES  cinemadb.Pessoas ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  FOREIGN KEY ( Filme )
    REFERENCES  cinemadb.Filmes ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação PessoasSéries

```
CREATE TABLE IF NOT EXISTS cinemadb.PessoasSeries (
  Pessoa INT NOT NULL,
  Serie INT NOT NULL,
  Episodio VARCHAR(45) NOT NULL,
  PRIMARY KEY (Serie, Pessoa, Episodio),
  INDEX Pessoa_idx (Pessoa ASC),
  FOREIGN KEY (Pessoa)
    REFERENCES cinemadb.Pessoas ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
 FOREIGN KEY ( Serie )
    REFERENCES cinemadb.Series ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação PessoaPrémios

```
CREATE TABLE IF NOT EXISTS cinemadb.PessoaPremios (
  Pessoa INT NOT NULL,
  NomePremio VARCHAR(45) NOT NULL,
  Ano INT NOT NULL,
  PRIMARY KEY (NomePremio, Ano, Pessoa),
  INDEX Pessoa_idx (Pessoa ASC),
  FOREIGN KEY ( Pessoa )
```

```
    REFERENCES cinemadb.Pessoas ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação FilmeGéneros

```
CREATE TABLE IF NOT EXISTS cinemadb.FilmeGeneros (
  Filme  INT NOT NULL,
  Genero  VARCHAR(45) NOT NULL,
  PRIMARY KEY ( Genero ,  Filme ),
  INDEX Filme_idx ( Filme ASC),
  FOREIGN KEY ( Filme )
    REFERENCES cinemadb.Filmes ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação FilmePrémios

```
CREATE TABLE IF NOT EXISTS cinemadb.FilmePremios (
  Filme INT NOT NULL,
  NomePremio VARCHAR(45) NOT NULL,
  Ano INT NOT NULL,
  PRIMARY KEY (Filme, NomePremio, Ano),
  FOREIGN KEY ( Filme )
    REFERENCES cinemadb.Filmes ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação SérieGéneros

```
CREATE TABLE IF NOT EXISTS cinemadb.SerieGeneros (
  Serie  INT NOT NULL,
  Genero  VARCHAR(45) NOT NULL,
  PRIMARY KEY ( Serie ,  Genero),
  FOREIGN KEY ( Serie )
    REFERENCES  cinemadb.Series  ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
);
```

Relação SériePrémios

```
CREATE TABLE IF NOT EXISTS cinemadb.SeriePremios (
  Serie INT NOT NULL,
  NomePremio VARCHAR(45) NOT NULL,
  Ano INT NOT NULL,
  PRIMARY KEY (Serie, NomePremio, Ano),
  FOREIGN KEY ( Serie )
    REFERENCES cinemadb.Series ( id )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
);
```

Relação SérieEpisódios

```
CREATE TABLE IF NOT EXISTS cinemadb.SerieEpisodios (
  Serie INT NOT NULL,
  Episodio VARCHAR(45) NOT NULL,
  Temporada INT NULL,
  Duracao INT NULL,
  PRIMARY KEY ( Serie , Episodio )
  FOREIGN KEY ( Serie )
    REFERENCES  cinemadb.Series ( id )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
);
```

5.1.2 Desenho das restrições gerais

Nesta secção são definidas as restrições que garantem a coerência dos dados. Por exemplo, não faz sentido existir uma série cuja data de início seja superior à data em que foi finalizada. Estas restrições garantem que casos como estes não ocorrem.

Relação Pessoas

- O ano em que uma pessoa recebe um prémio não pode ser anterior ao ano em que essa pessoa nasceu.

```
CREATE FUNCTION anoPremio
      (IN pessoaID INT, anoPremio INT)
      RETURNS BOOLEAN
BEGIN
          DECLARE anoNascimento INT
          SELECT YEAR(DataNasc) INTO anoNascimento
          FROM Pessoas
          WHERE id = pessoaID;

          IF (anoPremio >= anoNascimento) THEN RETURN TRUE;
          ELSE RETURN FALSE;
END;

ALTER TABLE PessoaPremios
          ADD CONSTRAINT AnoDoPremio
                    CHECK (anoPremio(Pessoa, Ano) == TRUE);
```

Relação Filmes

- O ano em que um filme recebe um prémio não pode ser anterior ao ano em que esse filme estreou.

```
CREATE FUNCTION anoPremioFilme
    (IN filmeID INT, anoPremio INT)
    RETURNS BOOLEAN
BEGIN
        DECLARE anoEstreia INT
        SELECT Ano INTO anoEstreia
        FROM Filmes
        WHERE id = FilmeID;

        IF (anoPremio >= anoEstreia) THEN RETURN TRUE;
        ELSE RETURN FALSE;
END;

ALTER TABLE FilmePremios
        ADD CONSTRAINT AnoDoPremioFilme
                CHECK (anoPremioFilme(Filme, Ano) == TRUE);
```

Relação Séries

- O ano em que uma série terminou não pode ser anterior ao ano em que essa série estreou.

```
ALTER TABLE Series
        ADD CONSTRAINT AnoInicioAnoFim
                CHECK (AnoInicio <= AnoFim);
```

- O ano em que uma série recebe um prémio não pode ser anterior ao ano em que essa série estreou.

```
CREATE FUNCTION anoPremioSerie
    (IN serieID INT, anoPremio INT)
    RETURNS BOOLEAN
BEGIN
        DECLARE anoEstreia INT
        SELECT AnoInicio INTO anoEstreia
        FROM Series
        WHERE id = SerieID;

        IF (anoPremio >= anoEstreia) THEN RETURN TRUE;
        ELSE RETURN FALSE;
END;

ALTER TABLE SeriePremios
        ADD CONSTRAINT AnoDoPremioSerie
                CHECK (anoPremioSerie(Serie, Ano) == TRUE);
```

5.2 Organização de ficheiros e índices

Nesta secção é analisada a forma como são guardados os ficheiros que constituem o sistema de base de dados. É também analisada a necessidade de criação de índices no sentido de aumentar a performance do sistema.

5.2.1 Análise de transações

De maneira a melhorar a performance da base de dados é necessário analisar as transações mais importantes que nela ocorrem, isto é, as transações que são executadas com mais frequência e que terão um impacto significativo no desempenho da base de dados.

Para cada delas uma deve-se conhecer a altura do dia em que são executadas, se os intervalos de tempo em que elas executam não se sobrepõem, isto é, se executam em exclusão mútua, então o risco de problemas de performance é reduzido, no entanto, se os seus padrões de operação entram em conflito então torna-se necessário examinar essas transações mais detalhadamente de maneira a determinar as relações que são mais frequentemente por elas acedidas, e se possível, efetuar alterações na estrutura dessas relações de maneira a tornar o seu acesso o mais rápido possível, melhorando assim a performance da base de dados.

Assim, após uma análise do modelo, percebemos que as transações mais importantes são:

1) Inserir os detalhes de um novo filme.
2) Inserir os detalhes de um ator de um filme.
3) Inserir os detalhes de uma crítica de um filme.

A seguir é apresentado o código para as transações anteriores relativas a filmes.

Transação 1)

```
CREATE PROCEDURE insereFilme(IN titulo VARCHAR(75), IN ano INT, IN duracao INT, IN
faixaEt VARCHAR(75), IN genero VARCHAR(45), IN premio VARCHAR(45), IN anoPremio INT)
BEGIN

DECLARE Erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

START TRANSACTION;

INSERT INTO Filmes(Titulo, Ano, Duracao, FaixaEtaria)
VALUES (titulo, ano, duracao, faixaEt);

INSERT INTO FilmeGeneros(Genero)
VALUES (genero);

INSERT INTO FilmePremios(NomePremio, Ano)
VALUES (premio, anoPremio);

IF Erro THEN
```

```
  ROLLBACK;
ELSE
  COMMIT;
END IF;
END;
```

Transação 2)

```
CREATE PROCEDURE insereAtorFilme(IN nome VARCHAR(75), IN data DATE, IN filme INT)
BEGIN

DECLARE atorID INT;
DECLARE Erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

START TRANSACTION

INSERT INTO Pessoas(Nome, Realizador, Ator, DataNasc)
VALUES (nome, FALSE, TRUE, data);

SELECT id INTO atorID
FROM Pessoas
WHERE Nome = nome AND DataNasc = data;

INSERT INTO PessoasFilmes(Pessoa, Filme)
VALUES (atorID, filme);

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

Transação 3)

```
CREATE PROCEDURE insereCriticaFilme(IN pontuacao INT, IN conteudo TINYTEXT, IN filme
INT, IN utilizador VARCHAR(45))
BEGIN

DECLARE Erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

START TRANSACTION;

INSERT INTO Criticas(Pontuacao, Conteudo, Filme, Serie, Utilizador)
VALUES (pontuacao, conteudo, filme, NULL, utilizador);

IF Erro THEN
  ROLLBACK;
ELSE
  COMMIT;
```

```
END IF;
END;
```

5.2.2 Mapa de transações

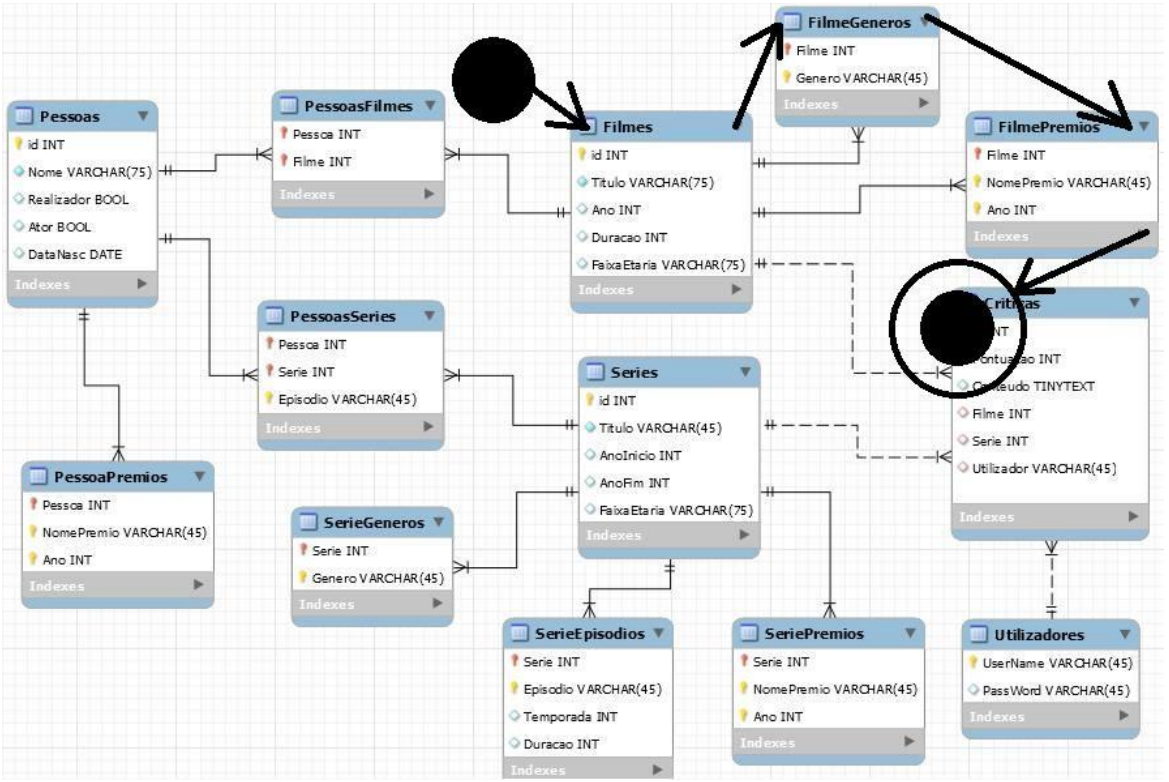

Figura 3 - Mapa da transação 1

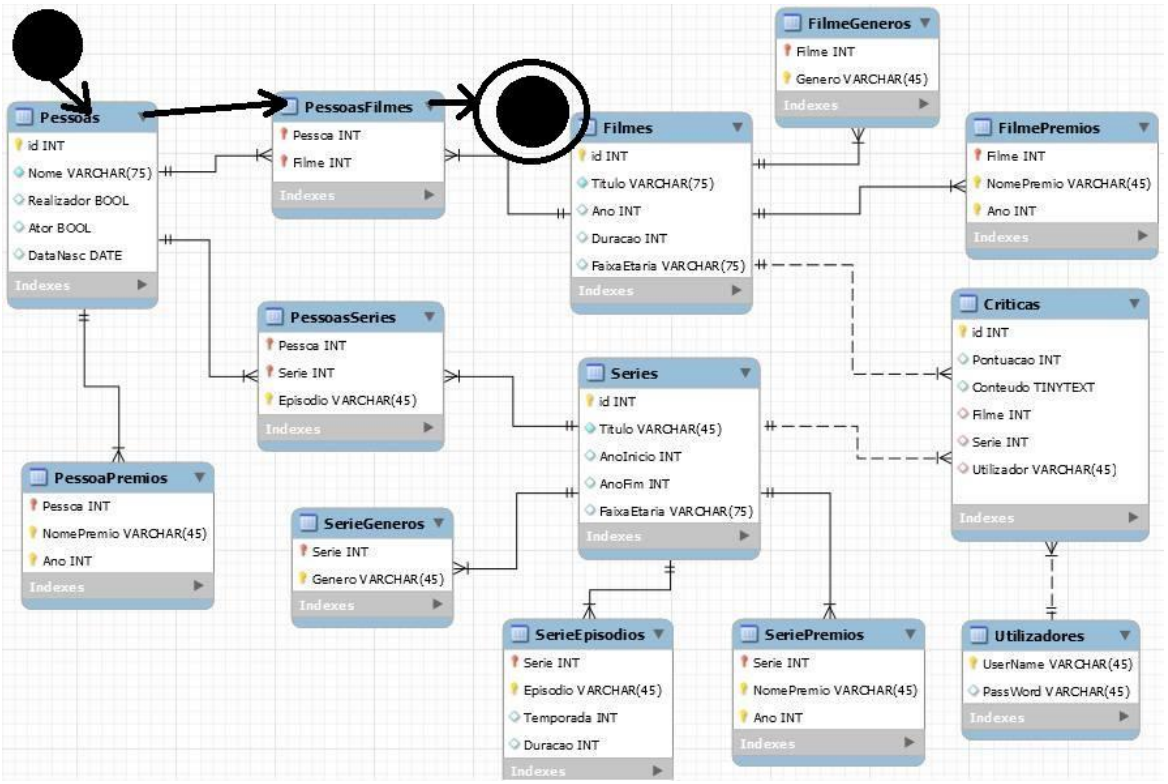

Figura 4 - Mapa da transação 2

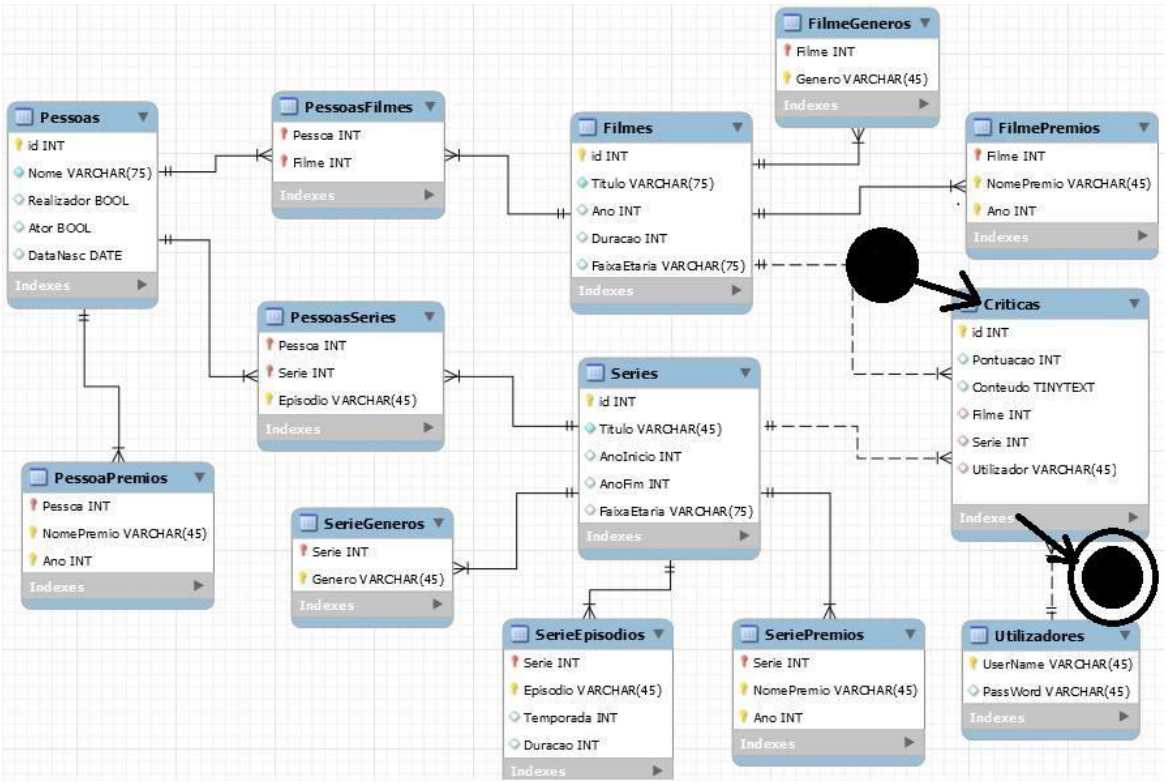

Figura 5 - Mapa da transação 3

5.2.3 Estimativa de espaço em disco necessário

Para se fazer esta estimativa é necessário saber o espaço necessário para cada tipo de domínio que pretendemos usar na base de dados. No InnoDB os tamanhos de cada domínio são os seguintes:

- Integer 4 bytes
- Date 3 bytes
- Varchar (N) 1+N bytes
- TinyText 255 bytes
- Boolean 2 bytes

Após alguns cálculos, concluímos que o espaço máximo necessário para um registo em cada uma das tabelas são os seguintes:

- Pessoas 87 bytes
- PessoasFilmes 8 bytes
- PessoasSeries 54 bytes
- PessoasPremios 54 bytes
- Filmes 164 bytes
- FilmeGeneros 50 bytes

- FilmePremios 54 bytes
- Series 134 bytes
- SerieGeneros 50 bytes
- SerieEpisodios 68 bytes
- SeriePremios 54 bytes
- Criticas 317 bytes
- Utilizadores 92 bytes

Dado não termos qualquer informação acerca do tamanho inicial da base de dados, vamos supor alguns valores iniciais.

- Existem cerca de 40 atores e 10 realizadores envolvidos nos filmes e séries.
- Metade dos atores participa em filmes e a outra metade em séries.
- Metade dos realizadores faz filmes e a outra metade faz séries.
- Cada ator participou em cerca de 6 filmes.
- Cada realizador realizou cerca de 3 filmes.
- Cada ator participou em cerca de 2 séries.
- Cada realizador realizou cerca de 2 séries.
- Cada pessoa começa com 2 prémios.
- Existem cerca de 30 filmes.
- Cada filme encaixa-se em cerca de 3 géneros.
- Cada filme ganhou cerca de 2 prémios.
- Existem cerca de 20 séries.
- Cada série ganhou cerca de 2 prémios.
- Cada série encaixa-se em cerca de 3 géneros.
- Cada série tem cerca de 100 episódios.
- Existem cerca de 30 utilizadores.
- Cada utilizador fez cerca de 5 críticas.

Tamanho inicial estimado de cada tabela:

Pessoas - 50*87
PessoasFilmes - (20*6 + 5*3) * 8
PessoasSeries - (20*2 + 5*2) * 54
PessoasPremios - 50*2*54
Filmes - 30*164
FilmeGeneros - 30*3*50
FilmePremios - 30*2*54
Series - 20*134
SerieGeneros - 20*3*50
SerieEpisodios - 20*100*68
SeriePremios - 20*2*54
Criticas - 30*5*317
Utilizadores - 30*92

Somando os tamanhos estimados de cada tabela, concluímos que o tamanho inicial da base de dados será de cerca de 216269 bytes (216 KB).

5.3 Mecanismos de segurança

Esta base de dados prevê dois tipos de utilizadores: Utilizadores e Administradores. Estes utilizadores têm permissões diferentes mas a mesma vista.

Utilizador

Estes Utilizadores têm permissão para consulta de todas a tabelas, inserção e alteração da tabela Utilizadores e inserção, alteração e remoção da tabela Critica.

Permissões para Utilizadores:

```
GRANT SELECT ON Pessoas TO Utilizador
GRANT SELECT ON PessoasFilmes TO Utilizador
GRANT SELECT ON PessoasSeries TO Utilizador
GRANT SELECT ON PessoasPremios TO Utilizador
GRANT SELECT ON Filmes TO Utilizador
GRANT SELECT ON FilmeGeneros TO Utilizador
GRANT SELECT ON FilmePremios TO Utilizador
GRANT SELECT ON Series TO Utilizador
GRANT SELECT ON SerieGeneros TO Utilizador
GRANT SELECT ON SerieEpisodios TO Utilizador
GRANT SELECT ON SeriePremios TO Utilizador
GRANT SELECT ON Criticas TO Utilizador
GRANT SELECT ON Utilizadores TO Utilizador

DENY INSERT ON Pessoas TO Utilizador
DENY INSERT ON PessoasFilmes TO Utilizador
DENY INSERT ON PessoasSeries TO Utilizador
DENY INSERT ON PessoasPremios TO Utilizador
DENY INSERT ON Filmes TO Utilizador
DENY INSERT ON FilmeGeneros TO Utilizador
DENY INSERT ON FilmePremios TO Utilizador
DENY INSERT ON Series TO Utilizador
DENY INSERT ON SerieGeneros TO Utilizador
DENY INSERT ON SerieEpisodios TO Utilizador
DENY INSERT ON SeriePremios TO Utilizador
GRANT INSERT ON Criticas TO Utilizador
GRANT INSERT ON Utilizadores TO Utilizador

DENY UPDATE ON Pessoas TO Utilizador
DENY UPDATE ON PessoasFilmes TO Utilizador
DENY UPDATE ON PessoasSeries TO Utilizador
DENY UPDATE ON PessoasPremios TO Utilizador
```

```
DENY UPDATE ON Filmes TO Utilizador
DENY UPDATE ON FilmeGeneros TO Utilizador
DENY UPDATE ON FilmePremios TO Utilizador
DENY UPDATE ON Series TO Utilizador
DENY UPDATE ON SerieGeneros TO Utilizador
DENY UPDATE ON SerieEpisodios TO Utilizador
DENY UPDATE ON SeriePremios TO Utilizador
GRANT UPDATE ON Criticas TO Utilizador
GRANT UPDATE ON Utilizadores TO Utilizador
```

```
DENY DELETE ON Pessoas TO Utilizador
DENY DELETE ON PessoasFilmes TO Utilizador
DENY DELETE ON PessoasSeries TO Utilizador
DENY DELETE ON PessoasPremios TO Utilizador
DENY DELETE ON Filmes TO Utilizador
DENY DELETE ON FilmeGeneros TO Utilizador
DENY DELETE ON FilmePremios TO Utilizador
DENY DELETE ON Series TO Utilizador
DENY DELETE ON SerieGeneros TO Utilizador
DENY DELETE ON SerieEpisodios TO Utilizador
DENY DELETE ON SeriePremios TO Utilizador
GRANT DELETE ON Criticas TO Utilizador
DENY DELETE ON Utilizadores TO Utilizador
```

Administrador

O administrador têm permissão para consulta, inserir, alterar e remover de todas a tabelas.

Permissões para Administrador: ALL

```
GRANT ALL ON Pessoas TO Administrador
GRANT ALL ON PessoasFilmes TO Administrador
GRANT ALL ON PessoasSeries TO Administrador
GRANT ALL ON PessoasPremios TO Administrador
GRANT ALL ON Filmes TO Administrador
GRANT ALL ON FilmeGeneros TO Administrador
GRANT ALL ON FilmePremios TO Administrador
GRANT ALL ON Series TO Administrador
GRANT ALL ON SerieGeneros TO Administrador
GRANT ALL ON SerieEpisodios TO Administrador
GRANT ALL ON SeriePremios TO Administrador
GRANT ALL ON Criticas TO Administrador
```

GRANT ALL ON Utilizadores TO Administrador

6. Ferramentas utilizadas

Como auxílio ao desenvolvimento deste projeto, utilizamos as seguintes ferramentas:

- Microsoft Word 2013: Elaboração deste documento.
- BrModelo: Modelo Conceptual.
- MySQL Workbench: Desenvolvimento do modelo lógico e físico.

7. Conclusões e trabalho futuro

Tendo como objetivo tornar a base de dados resultante do nosso trabalho segura, fiável e fácil de aceder aos utilizadores, é necessário que o trabalho de preparação/implementação da base de dados siga os passos de desenvolvimento adquiridos nas aulas.

A criação de uma Base de Dados não é um processo de uma fase apenas e tendo isso em mente verificamos que seguir um plano de desenvolvimento pode ser difícil, mas torna mais fácil prever todos os cenários de utilização e assim criar uma aplicação melhor.

Este processo envolve todo um conjunto de passos a serem efetuados sequencialmente, começando pela análise detalhada do que o utilizador necessita (analise de requisitos), definindo bem as entidades necessárias e estabelecendo relacionamentos entre elas de modo a assegurar um sistema que permita o armazenamento e a possibilidade de gerir os dados pretendidos.

Qualquer base de dados precisa de manutenção periódica para garantir o seu correto funcionamento ao longo do tempo, se tal não acontecer, a sua performance diminuirá, deixando de ser funcional, como tal, para trabalho futuro, a manutenção e atualização da base de dados são os aspetos mais importantes.

Após uma profunda reflexão chegamos a conclusão que os requisitos funcionais definidos no início do projeto foram cumpridos com sucesso.

8. Bibliografia

Connolly, T. M. & Begg, C.E., 2005. Database Systems - A Practical Approach to Design, Implementation and Management Fourth Edition, S.I.: Addison-Wesley