Faculdade de Engenharia da Universidade do Porto
Bases de Dados
2018/19

Indústria Cinematográfica

Relatório do Projeto

2ª Entrega

T3G01
Bernardo Manuel Esteves dos Santos
up201706534
David Luís Dias da Silva
up201705373
Luís Pedro Pereira Lopes Mascarenhas Cunha
up201706736

Conteúdos

Contexto da Base de Dados 3
1ª Entrega 4
Contexto 4
Dificuldades 6
Diagrama UML 7
2ª Entrega 8
Revisão do modelo conceptual 8
Diagrama UML Revisto 9
Esquema Relacional, dependências funcionais e chaves 10
Análise das dependências funcionais e das formas normais 14
Restrições 15

Contexto da Base de Dados

A base de dados proposta para este projeto pretende modular a indústria cinematográfica de um modo semelhante ao IMDb, tendo sido desenhada com algumas funcionalidades desta plataforma em mente, estas serão exploradas em mais detalhe mais à frente assim como a constituição do nosso modelo.

A maior utilidade do IMDb é a possibilidade de obter uma imensidão de informação acerca de qualquer pessoa que trabalhe na indústria cinematográfica ou de qualquer conteúdo, seja este um episódio em específico de uma série de televisão, a série como um todo ou um filme. Esta funcionalidade é a base da nossa modulação, sendo a nossa base de dados composta pelos diversos conteúdos e pessoas da indústria. Deste modo o utilizador poderá então obter a mais diversa informação, desde o ano de lançamento de um conteúdo, listado por países, ou elenco do mesmo, até à biografia de um ator ou mesmo prémios ganhos, quer por atores ou conteúdos.

Outra funcionalidade que consideramos vital numa base de dados relacionada com este tema é a capacidade de os nossos utilizadores fazerem críticas a qualquer tipo de conteúdo presente na mesma. Numa crítica é pedido ao utilizador uma avaliação a quantitativa, de 0 a 10, do conteúdo, tornando possível que qualquer conteúdo tenha uma avaliação proveniente da comunidade.

Os utilizadores têm ainda a opção de manter uma *watchlist*, uma lista de filmes ou episódios de séries que pretendem visualizar mais tarde.

1ª Entrega

Contexto

A peça central da modulação proposta é a classe *Content* que representa o conceito dos conteúdos que podem ser visualizados pelas pessoas e sobre o qual se debruça o tema. A classe *Content* serve como a generalização de duas outras classes: *VideoContent* e *TVShow*. A classe guarda, portanto, os atributos que são comuns a ambas as classes, sendo eles o nome do conteúdo, as suas restrições de idade, a sua descrição e o seu score (ver classe *Review*). A classe associa-se também a uma classe *Genre* que representa o género dos conteúdos (ex.: ação e romance). A generalização é completa e disjunta.

A classe *VideoContent* representa as unidades do conteúdo cinematográfico, servindo como generalização das classes *Movie* e *Episode.* Estas unidades têm em comum as informações sobre a sua duração e o seu orçamento. Esta generalização também é completa e disjunta. Um *VideoContent* associa-se a um dado país para guardar a informação acerca da data em que o conteúdo foi lançado (classe *Released*).

Um filme, representado pela classe *Movie* guarda informação sobre o ano em que saiu (elemento que pode ser derivado da sua primeira data de lançamento). O ano serve como identificador desambiguador de um filme, dado a estes serem usualmente identificados através do seu nome e ano de lançamento (ex.: *"King Kong (2005)" e "King Kong (1993)")*.

Um episódio (classe *Episode*), difere de um filme no seu enquadramento, devido a este fazer parte de um todo que é a série. Para tal o episódio guarda o número da temporada (*season) a* que pertence e o do seu número na mesma. Uma série de televisão (classe *TVShow,* derivada da classe *Content*) que está associada aos diversos episódios guarda informação acerca das suas datas de início e fim (caso já tenha terminado) e sobre o seu orçamento. Todos os atributos de *TVShow* são derivados de informações dos episódios que o constituem (datas de início, de fim e o orçamento).

Outro membro importante do modelo são as pessoas da indústria que são representadas pela classe *Person,* sendo guardadas informações sobre o seu nome, biografia, fotografia, género (classe *Gender*), idade e datas de nascimento e morte (caso exista).

As pessoas estabelecem participações em filmes (classe *MovieParticipation*), que é caracterizada por um papel nessa participação (classe Role*,* ex.: realizador, diretor, ator principal, encenador, etc.) e, caso o seu papel seja ator, por uma associação com a personagem que desempenha nesse filme (classe *Cast*).

Da mesma forma, as pessoas podem ter participações numa série, sendo que a diferença é que estas participações estão associadas não só a cada episódio, mas também na série em si (*TVShow*).

Na indústria cinematográfica, os filmes, séries e pessoas podem receber prémios. Para modelar estas situações criamos as classes *MovieAward* e *TVAward* que guardam informação acerca do ano em que foram concedidos, o seu nome (ex.: *Oscar, Emmy, etc.,* ver classe *AwardName*) e sobre a categoria (ex.: melhor ator, melhor realizador, melhor filme. Ver classe *Category*). Os prémios mencionados podem se associar a participações (*MovieParticipation e*

TVParticipation) , no caso de prémios como “melhor ator”, “melhor realizador” , etc., ou a filmes ou séries, no caso de prémios como “melhor filme” e “melhor série dramática”.

Por último, o nosso modelo contempla também utilizadores (classe *User*) sobre os quais se guardam informações sobre o seu nome e e-mail. O papel dos utilizadores é o de efetuar críticas (classe *Review*) aos conteúdos existentes (filmes, séries ou episódios individuais), e é a partir destas que são calculados os valores do atributo “*score*” da classe *Content*.

Dificuldades

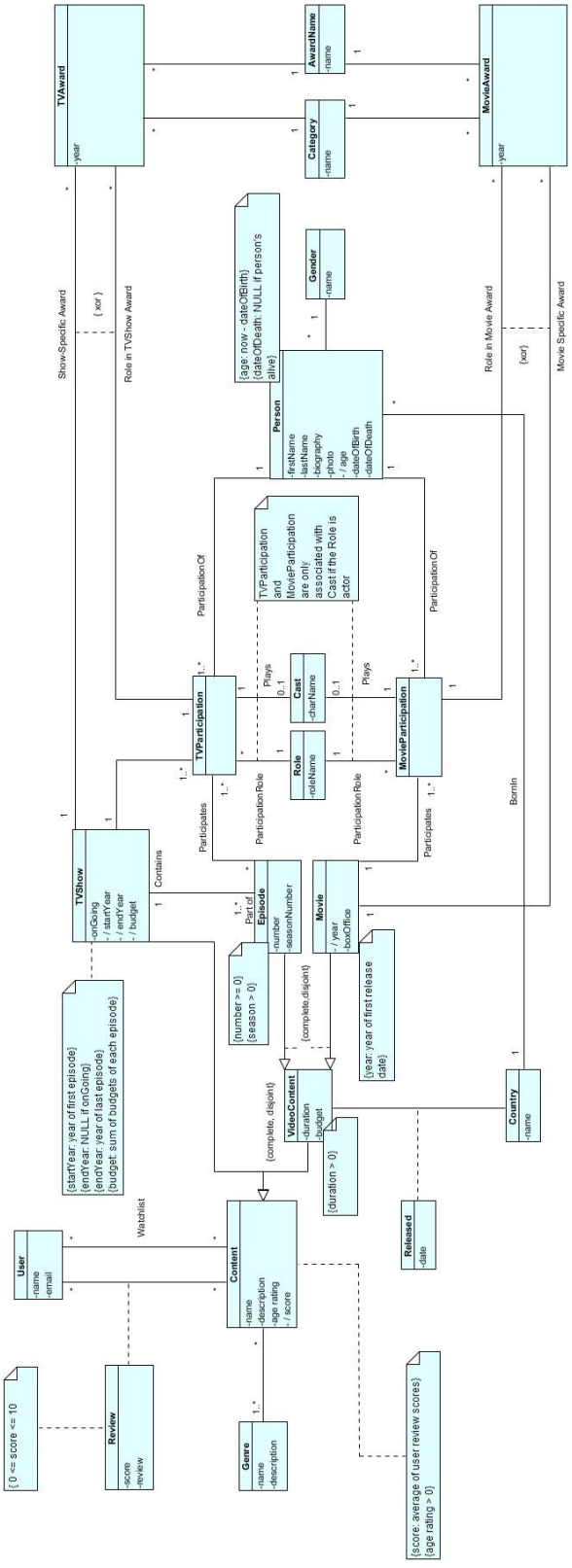
A modelação da situação apresentada apresentou-se como um desafio, na medida em que a procura de uma forma ótima de a representar com simplicidade e com fieldade a situação de mundo real levaram a necessidade de alterar o "caminho" escolhido. De facto, à medida que desenvolvíamos o modelo, tornavam-se mais evidentes as ligações entre os vários elementos, e da forma de como as escolhas de otimização de umas das "peças" do modelo podiam levar a situações menos ótimas noutras partes. Assim, gerou-se uma necessidade de escolher as prioridades do modelo e de gerir o seu desenvolvimento em prol dessas necessidades. Algumas das dificuldades foram agravadas pelos detalhes inerentes ao tema em si, para os quais necessitamos de fazer uma investigação que transcende o conhecimento geral.

O segmento do modelo que se apresentou mais desafiante foi o que constitui os prémios. Esta dificuldade deve-se a este segmento estar numa forte ligação com os principais elementos do modelo, que são os conteúdos televisivos, os filmes e as pessoas que participam nesses conteúdos. Além disto, os prémios apresentam diversas variantes, sendo que podem ser atribuídos a participações em filmes ou séries, como aos filmes e séries. Como um prémio tem várias categorias, que se repetem ao longo de diferentes prémios, tentamos que o nosso modelo evitasse redundâncias, diminuindo a quantidade de informação a guardar.

Inicialmente tivemos dúvidas acerca de qual seria a melhor opção de representação do conteúdo, se considerávamos uma generalização entre filme e episódio, entre filme e série, ou se utilizávamos uma generalização entre série e conteúdo de vídeo, que por sua vez generaliza filme e episódio. Acabamos por escolher a terceira opção pois permitiu obter uma leitura mais fácil e uma ligação mais simples com o segmento dos prémios e das críticas dos utilizadores.

Outra dificuldade prendeu-se ao conhecimento acerca de detalhes da modelação de uma base de dados, por exemplo acerca da necessidade de substituir alguns atributos por uma classe (ex.: datas).

Diagrama UML



2ª Entrega

Revisão do modelo conceptual

Após a análise do modelo conceptual anterior detetámos algumas falhas no mesmo, o que levou a um novo modelo que elimina eventuais fragilidades e não-conformidades com a linguagem UML do anterior e o simplifica, tornando a compreensão do mesmo mais fácil.

Talvez a maior mudança da última entrega para a atual terá sido a forma como modulamos o conteúdo (classe *Content*). Anteriormente este era uma generalização de *TVShow* e de *videoContent*, que por sua vez era uma generalização de Movie e *Episode*. Após a revisão este passou a ser apenas uma generalização das últimas duas pois *TVShow* não era por si só conteúdo, mas sim uma composição de episódios.

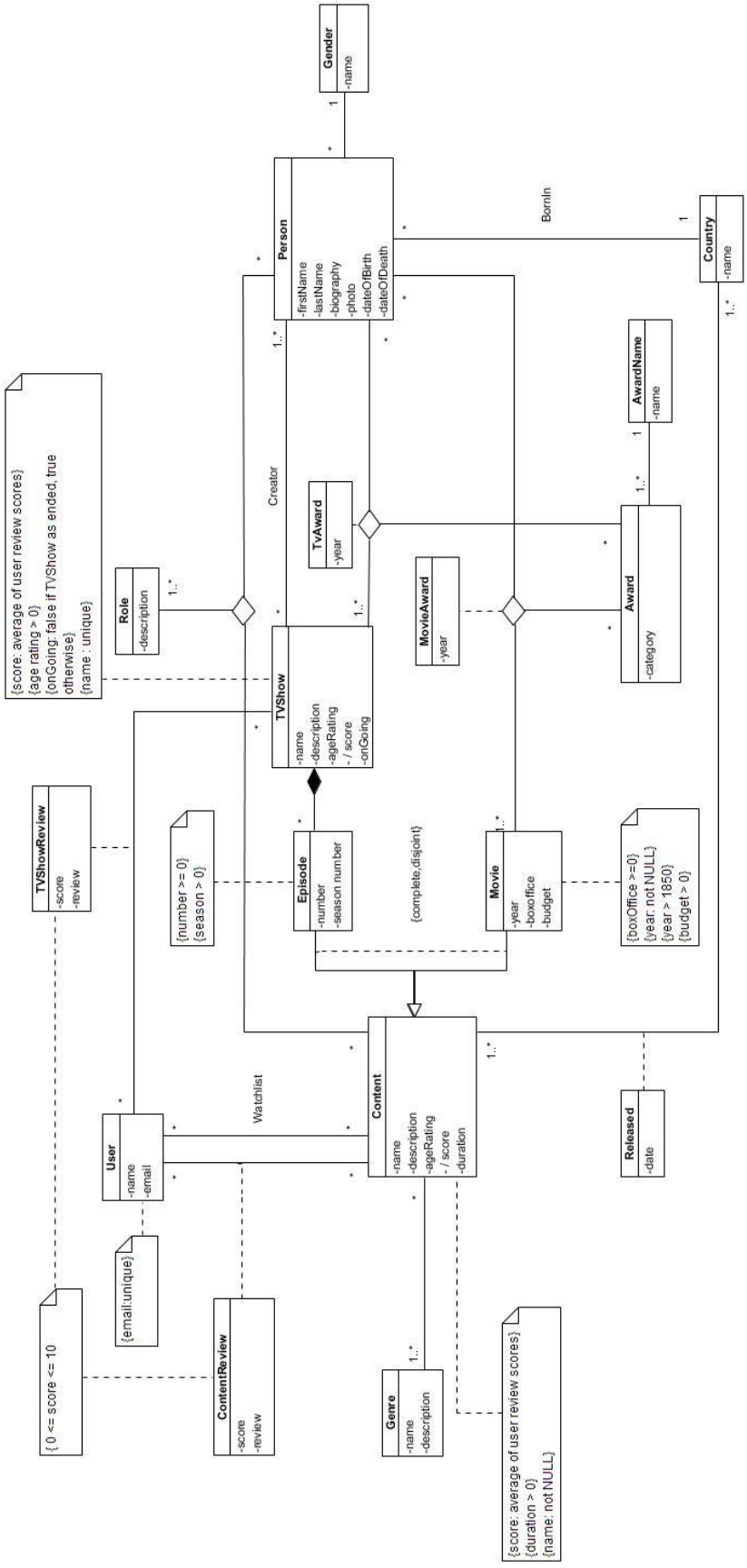
Devido à alteração prévia foi necessário a dividir *Review* em *ContentReview* e *TVShowReview*, mantendo assim a possibilidade de um *User* fazer críticas tanto a séries televisivas como a filmes e a episódios isolados.

Um *TVShow* deixou de ter um género, pois diferentes episódios da mesma podem ter diferentes géneros, por exemplo, podem existir especiais de Halloween que serão classificados como terror embora o género principal da série não o ser.

Foi simplificada também a forma como eram abordadas as participações das pessoas no conteúdo. No modelo revisto estas só têm participações em conteúdos (filmes ou episódios), uma vez que episódios isolados podem ter diferentes elencos ou diferentes realizadores podem trabalhar diferentes episódios de uma mesma série. Embora este aspeto tenha sofrido alterações continua a manter-se a possibilidade da mesma pessoa ter diferentes tipos de participações num mesmo conteúdo (a mesma pessoa pode participar como ator num filme e ao mesmo tempo realizá-lo), por esta mesma razão justifica-se o uso de uma associação ternária.

A última alteração feita ao antigo modelo foi a atribuição dos prémios. Mantém-se a possibilidade de um prémio ser atribuído quer a um filme ou série, como a uma participação num destes dois. No caso dos filmes, *Movie*, *Person* e *Award* estão ligados por uma associação ternária com uma classe de associação *MovieAward*, esta contem apenas o ano em que o prémio foi atribuído ao filme em caso. O facto de a multiplicidade de Movie ser “1 ou mais” e das restantes classes ser “0 ou mais” permite que seja atribuído um prémio apenas ao filme em si e não apenas a participações (as multiplicidades desta associação encontram-se no diagrama UML revisto, que se encontra abaixo representado). No caso das séries de televisão a única diferença é o nome da classe de associação, sendo neste caso *TVAward*, mas esta tem a mesma função que na situação anterior.

Diagrama UML Revisto



Esquema Relacional, dependências funcionais e chaves

Content (<u>id</u>, name, description, ageRating, score, duration)

Dependências Funcionais:

- id → name, description, ageRating, score, duration

Chaves:

- {id}

Episode (<u>id</u> → Content, number, seasonNumber, tvshowID->TVShow)

Dependências Funcionais:

- id → number, seasonNumber, tvShowID
- number, seasonNumber, tvShowID → id

Chaves:

- {id}
- {number, seasonNumber, tvShowID}

Movie (<u>id</u> → Content, year, boxOffice,budget)

Dependências Funcionais:

- id → year, box-office,budget

Chaves:

- {id}

TVShow (<u>id</u>, name, description, ageRating, score)

Dependências Funcionais:

- id → name, description, ageRating, score
- name → id, description, ageRating, score

Chaves:

- {id}
- {name}

Award (<u>id</u>, category, awardNameID → AwardName)

Dependências Funcionais:

- id → category, awardNameID

Chaves:

- {id}

AwardName (<u>id</u>, name)

Dependências Funcionais:

- id → name

- name → ida

Chaves:

- {id}
- {name}

MovieAward (year, awardID → Award, movieID → Movie, personID→Person)

Dependências Funcionais:

- awardID, movieID, personID → year

Chaves:

- {awardID, movieID, personID}

TvAward (year, awardID → Award, tvShowID → TVShow, personID→Person)

Dependências Funcionais:

- awardID, tvShowID, personID → year

Chaves:

- {awardID, tvShowID, personID}

Person (id, firstName, lastName, biography, photo, dateOfBirth, dateOfDeath, country → Country, gender → Gender)

Dependências Funcionais:

- id → firstName, lastName, biography, photo, dateOfBirth, dateOfDeath, country, gender

Chaves:

- {id}

Role (id, description)

Dependências Funcionais:

- id → description

Chaves:

- {id}

RolePersonContent (roleID → Role, personID → Person, contentID → Content)

Dependências Funcionais

- não existem dependências funcionais

Chaves:

- {roleID, personID, contentID}

Creator (personID, *tvshowID*)

Dependências Funcionais:

- não existem dependências funcionais

Chaves:

- {personID, tvshowID}

User (id, name, email)

Dependências Funcionais:

- id → name, email
- email → id, name

Chaves:

- {id}
- {email}

ContentReview (contentID → Content, userID → User, score, review)

Dependências Funcionais:

- contentID, userID → score, review

Chaves:

- {contentID, userID}

TVShowReview (tvshowID → TVShow, userID → User, score, review)

Dependências Funcionais:

- tvShowID, userID → score, review

Chaves:

- {id}

Watchlist (contentID → Content, userID → User)

Dependências Funcionais:

- não exitem dependências funcionais

Chaves:

- {contentID, userID}

Gender (id, name)

Dependências Funcionais:

- id → name
- name → id

Chaves:

- {id}
- {name}

Country (id, name)

Dependências Funcionais:

- id → name
- name → id

Chaves:

- {id}
- {name}

Released (contentID → Content, date, countryID → Country)

Dependências Funcionais:

- contentID, countryID → date

Chaves:

- {contentID, countryID}

Genre (id, name, description)

Dependências Funcionais:

- id → name, description
- name → id, description

Chaves:

- {id}
- {name}

ContentGenre (contentID → Content, genreID → Genre)

Dependências Funcionais:

- não existem dependências funcionais

Chaves:

- {contentID, genreID}

Análise das dependências funcionais e das formas normais

Com base nas dependências funcionais e chaves obtidas no ponto anterior deste relatório podemos afirmar que a base de dados proposta se encontra modelada segundo a forma normal de *Boyce-Codd*. Esta afirmação prende-se ao facto de para todas as dependências funcionais encontradas da forma $\overline{A} \rightarrow \overline{B}$, $\overline{A}$ é uma super-chave. Como consequência, a base de dados também está na terceira forma normal e em todas as anteriores (primeira e segunda).

É de notar que a adição, à grande maioria das relações, de atributos que representam *IDs* inteiros como chaves primárias (a fim de aumentar a eficiência da base de dados*), não introduziu violações das formas normais, pois os atributos que anteriormente serviam de chave primária passaram a ser também eles determinados funcionalmente pelo ID, e vice-versa.

* No caso do SQLite este aspeto não é tão relevante pois é feito automaticamente uma chave primária inteira para todas as tabelas que não o possuam (ROWID).

Restrições

Neste segmento serão mencionadas as restrições impostas ao modelo apresentado e a forma como estas foram impostas no SQLite. Para tal primeiro será feita uma descrição da restrição em linguagem natural seguido pelo método de implementação da mesma.

- Para as classes *ContentReview* e *TVShowReview* é esperado que o atributo *score* seja um valor entre 0 e 10. Em ambos os atributos a restrição foi declarada da seguinte forma:

```
score       INTEGER CHECK (score >= 0 AND score <= 10)
```

- A duração de um conteúdo, seja ele um filme ou um episódio não pode ter uma duração igual a zero e muito menos negativa, para tal o atributo *duration* da classe *Content* possui a restrição de ser maior que zero:

```
duration          INTEGER CHECK (duration > 0)
```

- Da mesma forma que na duração, também o orçamento (*budget*) de um filme terá de ter um valor superior a zero, pelo que:

```
budget            REAL CHECK (budget > 0)
```

- Os valores de bilheteira (dinheiro obtido por venda de bilhetes) de um filme, representados pelo atributo *boxOffice* têm de ser não negativos:

```
boxoffice         REAL CHECK (boxoffice >= 0)
```

- A indústria cinematográfica é relativamente recente à história humana, pelo que o ano dos filmes não pode ser um valor anterior a 1850 *:

```
year              INTEGER CHECK (year > 1850) NOT NULL
```

* Esta restrição poderá ser substituída com a introdução dos *triggers* dado a estre atributo poder ser derivado a partir de elementos da classe *Released.*

- O nome de uma série de televisão deve ser único (de facto o atributo *name* de *TVShow*) é uma chave candidata:

```
name              TEXT UNIQUE NOT NULL
```

- O número de um episódio não pode ser negativo (por vezes pode ser nulo no caso de um episódio piloto), e o número de uma temporada tem de ser positivo:

```
number              INTEGER CHECK (number >= 0) NOT NULL
seasonNumber        INTEGER CHECK (seasonNumber > 0) NOT NULL
```

- O atributo *category* da classe *Award* e *name* da classe *AwardName* não podem ser nulos pois nesse caso as classes perderiam significado:

```
category                TEXT NOT NULL,
name                    TEXT UNIQUE NOT NULL
```

- O mesmo acontece para os atributos *name* de *Gender*, Country e *Genre* que são únicos e que não devem ser nulos para não retirarem o significado às suas classes respetivas:

```
name                    TEXT UNIQUE NOT NULL
```

- A data de lançamento de um conteúdo num dado país representado na classe de associação *Released,* não pode ser nula para não retirar o significado à classe:

```
date                    DATE NOT NULL
```

- Embora não sejam uma chave candidate o nome, sobrenome e data de nascimento de uma pessoa (*firstname,lastname e dateOfBirth* de *Person*) não devem ser nulos para que seja de facto guardado um "mínimo" de informação relevante acerca da mesma:

```
firstName               TEXT NOT NULL,
lastName                TEXT NOT NULL,
dateOfBirth             DATE NOT NULL,
```

- Para além do visto anteriormente, caso uma pessoa já tenha falecido, a sua data de falecimento deve ser de facto posterior à sua data de nascimento:

```
CHECK ((dateOfBirth < dateOfDeath) or (dateOfDeath is NULL))
```

- A descrição do papel de uma pessoa num filme não deve ser nula para não retirar o prepósito desta classe:

```
description              TEXT NOT NULL
```

- Para os utilizadores da base de dados, é necessário saber o email e o nome. O email é identificativo do utilizador (também é uma chave candidate) pelo que deve ser único:

```
email                TEXT UNIQUE NOT NULL,
name                 TEXT NOT NULL
```

- Para finalizar esta secção é importante notar que caso exista uma chave candidata que não seja chave primária, foi-lhe dado a restrição de UNIQUE, como é o caso de *name* em *TVShow*, *name* em AwardName,*Gender , Country* e *Genre* e *email* em *User*.