Projecto de Bases de Dados (CC2005) - parte 1

1. Elementos do grupo

Grupo n.º 14

N.º mecanográfico	Nome
202203947	Gonçalo Martins Esteves
202206195	Nuno Santiago Ribeiro Gomes

2. Universo considerado e modelo de classes UML

Atualmente, a Netflix é uma das maiores plataformas de streaming mundial. Esta permite um rápido e cómodo acesso a uma variedade de filmes e séries à distância de alguns *clicks*. Dada a importância desta plataforma no quotidiano de muitas pessoas, decidiu realizar-se uma pequena implementação da forma como a *Netflix* é capaz de, não só organizar, bem como armazenar os seus dados (através de bases de dados).

Como os dados são imprescindíveis a qualquer base de dados procurou encontrar-se um dataset capaz de representar de forma precisa o nosso universo. Deste modo, o dataset utilizado foi retirado e adaptado do seguinte website: **Netflix Movies and TV Shows (kaggle.com)**.

Desta forma, é necessário ter em consideração que podem existir 2 tipos de *Shows* na plataforma: Filmes (*Movies*) e Séries (*TV Shows*). Mais ainda, é necessário ter em atenção que cada *show* apresenta a si associado não só um conjunto de géneros (descritos através do nome) no qual se enquadra, bem como um rating (constituído por um acrónimo e a sua respectiva descrição) que permite definir a sua audiência alvo e um conjunto de países (descritos através do nome) onde este foi produzido.

Consequentemente, para que cada *show* possa ser produzido é necessário ter em consideração as pessoas (definidas através do seu nome e cargo (ator ou realizador) no respetivo *show*) envolvidas no projeto, isto é, o seu *staff.*

Como o dataset contém múltiplos filmes e séries que são descritos através de: título, descrição, duração, rating, géneros, países de produção e staff envolvido é necessário organizar todas estas informações. Para tal, criaram-se 6 classes:

→ “Show” (responsável por armazenar a maior parte dos atributos dos filmes e séries - Ex: título e descrição);
→ “Country” (reflete todos os países onde foram realizadas e produzidas as filmagens);
→ “Type” (permite distinguir entre séries e filmes);
→ “Rating” (representa a audiência à qual o filme/série se destina - Ex: filmes R-rated destinam-se a uma audiência madura com +18 anos);
→ “Genre” (apresenta os géneros associados ao *show* - Ex: comédia, terror)
→ “Person” (representa as pessoas que estiveram envolvidas no filme/série)
→ “Job” (representa os cargos existentes em qualquer produção)

A classe *Show* trata-se da classe central do nosso modelo UML, tendo cada objecto o seu título, descrição e duração (minutos para os filmes e temporadas no caso das séries). Mais ainda, estes apresentam o respetivo ano de publicação, bem como o ano de adição à base de dados.

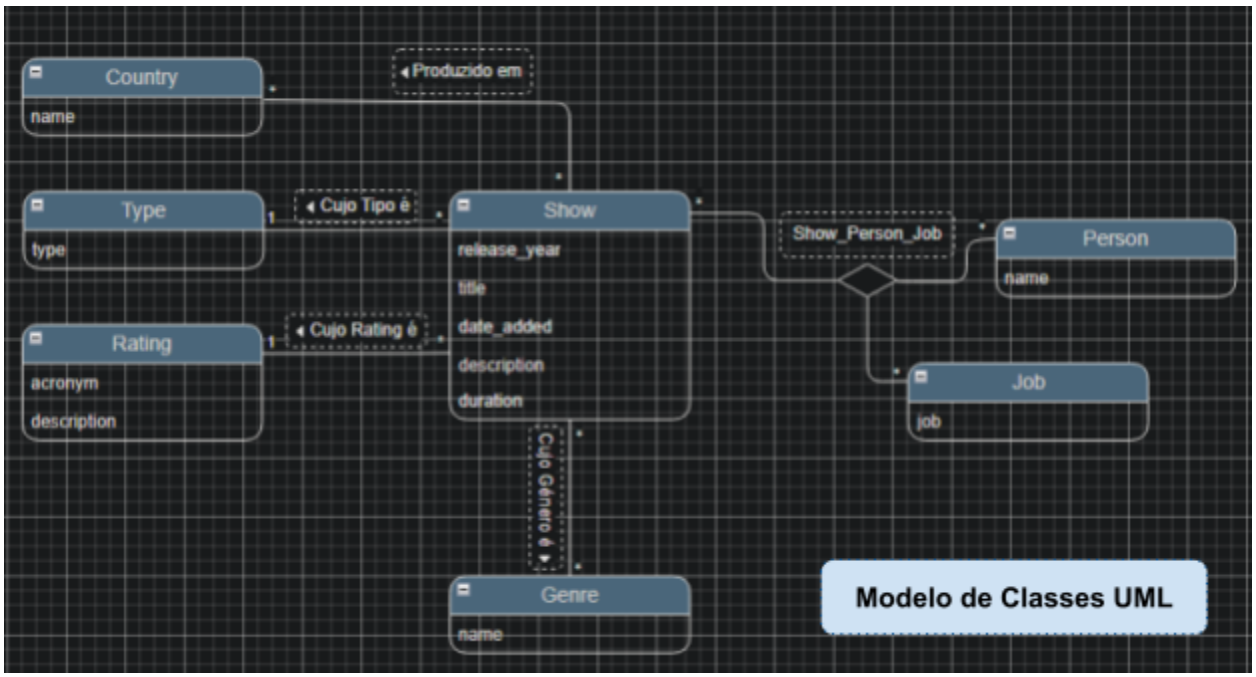
Tendo em conta que qualquer filme/série pode ter sido gravado em múltiplos países (e em cada país serem gravados vários filmes/séries), conter diversas pessoas associadas à sua produção (e estas trabalharem em vários filmes/séries) e ainda pertencer a múltiplas géneros (e cada género conter múltiplos filmes/séries), tem-se que a classe *Show* estará relacionada com as classes *Country*, *Person* e *Genre* através de associações de multiplicidade muitos para muitos.

Por outro lado, como cada entretenimento (objecto da classe *Show*) apresenta a si associado um único tipo (filme ou série) mas existem múltiplos filmes/séries na base de dados, então estamos perante uma associação de multiplicidade muitos para um.

Analogamente, o mesmo acontece com a classe *Rating.* Como cada filme/série apresenta como destino um único tipo de audiência e tendo em conta que estas podem ser alvo de múltiplas produções, verifica-se a existência de outra relação de multiplicidade muitos para um.

Mais ainda, como podem existir pessoas que desempenham múltiplas funções em várias produções, é necessária a utilização de uma classe-ternária (Show*_Person_Job*) entre as classes *Show, Person e Job* de forma a discriminar o(s) trabalho(s) de cada indivíduo.

Assim, através da análise pormenorizada do universo em questão, foi-nos possível construir o seguinte modelo de classes UML:



3. Modelo relacional

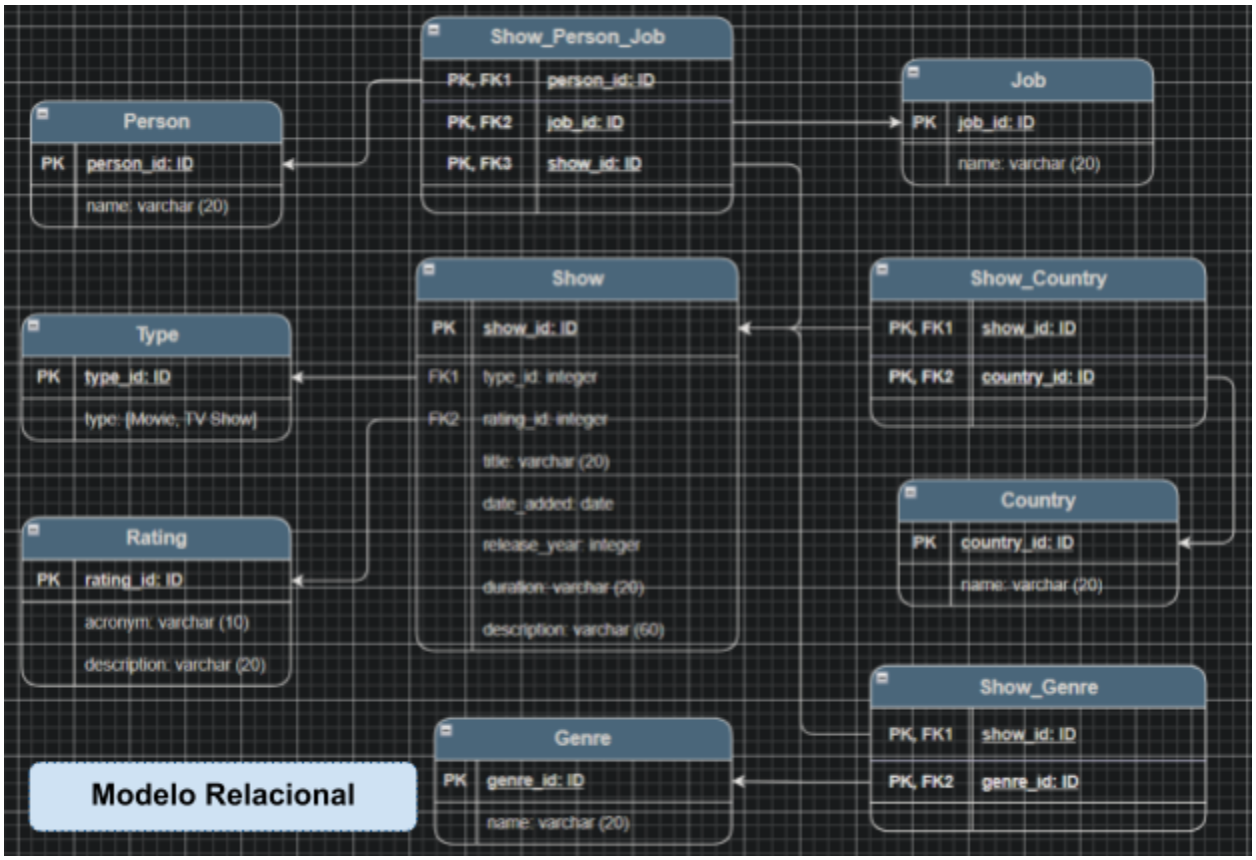
Inicialmente, ao traduzir um modelo de classes UML para um modelo relacional, é necessário, não só, ter em conta as classes utilizadas, bem como a multiplicidade das várias associações entre elas.

Deste modo, para cada classe criada anteriormente teremos a uma tabela no modelo relacional. Estas, por sua vez, apresentam uma única chave primária (responsável por identificar uma única entrada na tabela), como é o caso de *show_id* na tabela *Show*.

A tradução de uma associação muitos para muitos para o modelo relacional implica a introdução de uma nova tabela. Esta apresenta uma chave primária composta que alberga ambas as chaves primárias das tabelas/classes intervenientes. Assim, esta abordagem pode ser verificada através da utilização das tabelas *Show_Country, Show_Genre* e *Show_Person_Job*.

Por outro lado, a tradução de uma associação muitos para um implica a introdução de uma chave externa (*Foreign Key*) na tabela cuja classe no modelo UML pertence ao lado muitos da associação, que por sua vez, referencia a chave primária da tabela cuja classe pertence ao lado um da associação muitos para um no modelo de classes UML. Assim, facilmente se observa este fenómeno através da utilização de chaves externas na tabela *Show* para referenciar o tipo de *show* e a audiência alvo presente nas tabelas *Type* e *Rating*, respectivamente.

Por fim, foi-nos possível desenvolver o seguinte modelo relacional que foi posteriormente implementado na 2.ª parte do Projeto:


Show_Person_Job
PK, FK1 person_id: ID
PK, FK2 job_id: ID
PK, FK3 show_id: ID
Job
PK job_id: ID
name: varchar (20)
Person
PK person_id: ID
name: varchar (20)
Show
PK show_id: ID
FK1 type_id: integer
FK2 rating_id: integer
title: varchar (20)
date_added: date
release_year: integer
duration: varchar (20)
description: varchar (60)
Show_Country
PK, FK1 show_id: ID
PK, FK2 country_id: ID
Type
PK type_id: ID
type: (Movie, TV Show)
Country
PK country_id: ID
name: varchar (20)
Rating
PK rating_id: ID
acronym: varchar (10)
description: varchar (20)
Show_Genre
PK, FK1 show_id: ID
PK, FK2 genre_id: ID
Genre
PK genre_id: ID
name: varchar (20)
Modelo Relacional