Projeto de Banco de Dados

LOCADORA DE VÍDEO

Fábio Vicente
Fernanda Barros
Rafael Coelho
nanda_cbarros@hotmail.com
rafallcoelho@gmail.com
fab10_lima@hotmail.com

Salvador/BA
Setembro/2018

1. INTRODUÇÃO

A utilização de banco de dados no mundo moderno garante a agilidade nas transações e a memória de operações realizadas, bem como garante maior confiabilidade nas informações.

Assim sendo, este projeto foi idealizado para uma pequena locadora de vídeos.

2. JUSTIFICATIVA

A elaboração de um sistema tem seu início quando da implementação de um banco de dados. Fica claro ao analisar diversos autores que conhecer as teorias basilares de banco de dados garante maior integridade, confiabilidade e desempenho do sistema como um todo. Os conhecimentos deixados por Chen, Codd e outros entusiastas do passado nos garantem estas características.

Baseando-se nesses conceitos o projeto em pauta abrange as etapas de construção de uma base de dados hipotética criada com a finalidade de verificar os conceitos aprendidos em sala de aula.

O banco de dados final foi resultado de um levantamento detalhado de informações sobre uma pequena locadora de vídeo e posterior modelagem.

Para tanto se utilizou a estratégia de projeto de banco de dados Top-Down. A elaboração do projeto contemplou às seguintes fases:

1. Levantamento de requisitos
2. Modelagem conceitual (DER)
3. Modelagem lógica
4. Modelagem física

3. ETAPAS

Foram observadas as principais funcionalidades e ações realizadas, para então, se determinar quais seriam as entidades e os seus atributos relevantes. Em seguida, foram determinadas as relações entre essas entidades e os atributos dessas relações.

De posse desses elementos, foi efetuada uma normalização dos mesmos de forma a evitar redundâncias e problemas semânticos no banco de dados. A partir dessa normalização, foram criados os dicionários de dados para cada um desses sistemas, separadamente.

Foram definidas também quais seriam as retrições de igualdade que temos no banco.

Finalizando essa etapa, foi feita a modelagem dos sistemas, separadamente, utilizando-se o software BRmodelo. Foi utilizado o método semântico para a modelagem de banco de dados denominado MER – Modelo Entidade-Relacionamento.

3.1. Levantamento de Requisitos

O BD em questão efetua um <u>controle de locação de fitas.</u>

- ➢ Foi constatado que cada fita está associada a um número. Este número é o seu identificador;

- ➢ Cada fita possui um filme. Este é exclusivo;
- ➢ O filme possui, obrigatoriamente, atributos como título e categoria. Ele também é identificado por um código (identificador);
- ➢ O filme pode ter como informação o ator que estrela o filme. Porém, nem todo filme possui estrela.
- ➢ Não há filmes sem fita;
- ➢ Para os atores, deve ser informado o nome real e a data de nascimento;
- ➢ Para locar fitas, é necessário que o cliente seja cadastrado;
- ➢ Para efetuar o cadastro do cliente, é necessário informar o prenome, sobrenome, telefone e endereço. Para cada cliente é associado um número.
- ➢ Se faz necessário saber quais fitas os clientes possuem emprestadas. Porém, uma vez devolvida, não há necessidade de registrar histórico.

3.2. Elaboração do Modelo Conceitual

A construção deste modelo conceitual tem a finalidade de mostrar ao cliente os principais aspectos do banco de dados, assim como permitir uma interação mínima do usuário final com a tecnologia de banco de dados. Dessa forma, é possível a compreensão desse usuário de modo a garantir correção e respeito às regras de negócio por ele impostas.

Ver modelo conceitual no final do documento (*Anexo 1*)

3.3. Elaboração do Modelo Lógico

Finalizada a modelagem lógica, foi definido um dicionário de dados considerado o sistema como um todo. Nesta etapa foram revistas as restrições de integridade.

Ver modelo lógico no final do documento (*Anexo 2*)

3.3.1 Dicionário de Dados

Cliente			
Atributo	**Classe**	**Descrição**	**Domínio**
num_associado	Determinante	Cada cliente da locadora recebe um número associado.	Inteiro positivo.
telefone	Multivalorado	Telefone do cliente. Um cliente pode ter mais de um número de telefone associado.	Telefone fixo e/ou celular.
prenome	Simples	Nome do cliente	Primeiro nome de uma pessoa.
sobrenome	Simples	Sobrenome do cliente	Sobrenome de uma pessoa.

endereco	Composto	Endereço do cliente. Um endereço é um atributo composto de: CEP, rua, número.	3 valores consecutivos, sendo o primeiro um número do tipo XXXXX-XXX, o segundo o nome da rua, e o terceiro o número da casa.

Filme			
Atributo	**Classe**	**Descrição**	**Domínio**
identificador	Determinante	Cada filme recebe um identificador próprio.	Inteiro positivo.
titulo	Simples	Título do filme.	Nomes de filme.
duracao_locacao	Simples	Quanto tempo o filme pode permanecer na mão do cliente (filmes mais velhos normalmente tem tempos mais longos, enquanto lançamentos tem tempos mais curtos).	Inteiro positivo.
categoria	Simples	Categoria do filme (ex: comédia, drama, aventura…).	Classificação do filme quanto à abordagem do filme.

Ator			
Atributo	**Classe**	**Descrição**	**Domínio**
identificador	Determinante	O ator pode ser encontrado por um identificador único.	Inteiro positivo.
nome_real	Simples	Nome do ator.	Nome verdadeiro do ator.
data_nascimento	Simples	Data de nascimento do ator.	Data.

Fita			
Atributo	**Classe**	**Descrição**	**Domínio**
numero	Determinante	Cada fita possui um número único associado.	Inteiro positivo.
disponibilidade	Simples	A fita pode estar disponível para locação ou estar alugada.	Falso ou verdadeiro.

Empréstimo (Entidade Associativa)			
Atributo	**Classe**	**Descrição**	**Domínio**
data_devolução	Simples	Data que o cliente deve devolver a fita.	Data.
data_aluguel	Simples	Data que o cliente alugou a fita.	Data.
preco	Simples	Valor do empréstimo.	Valor em reais.

3.3.2 Restrições de Integridade

O sistema como um todo deverá impor uma série de restrições por ocasião da inserção, atualização e exclusão dos dados, as quais são descritas abaixo:

- Para alugar fitas, o cliente deve estar cadastrado no sistema;
- Um cliente não pode alugar fitas se ela não estiver disponível;
- Alguns poucos filmes necessitam de duas fitas para todo o conteúdo;
- O cliente e as fitas precisam ter um número associado;
- O filme e o ator precisam ter um código associado;
- Um fita só pode ser alugada por um cliente.

3.4. Implementação do Projeto (Modelo Físico)

Com o modelo do banco de dados concluído, resta a migração deste para o gerenciador de banco de dados, o qual se optou por utilizar-se o MySQL.

O script gerado neste documento foi gerado pela ferramenta case MySQL WorkBench, contudo foi observada a sintaxe de criação do Schema de modo a garantir a correção do código.

Ver modelo físico no final do documento (*Anexo 3*).

3.5. Normalização de Dataset Externo

Para realizar a importação de dados externos para a nossa base de dados, foi escolhido um dataset de filmes retirado do IMDB, que pode ser encontrado em https://data.world/popculture/imdb-5000-movie-dataset/discuss/popculture-imdb-5000-movie-dataset-general-discussion/mq2giylb.

Link da planilha normalizada: https://docs.google.com/spreadsheets/d/1FGzV2X0QzDAHAlWhH9nUJ54jFemJGYOeXeQMiTTVd6k/edit?usp=drive_web&ouid=113608157978008747331

3.5.1. Etapas de normalização

- **1FN**

As colunas *genres* e *plot_keywords* possuíam múltiplos valores para cada registro. Sendo assim, ambas foram transformadas em tabelas separadas, necessitando também da criação de tabelas relacionais, já que *genre* e *plot_keywords* se relacionam de forma *nxn* com *movie*.

- **2FN**

A base já se encontrava na primeira forma normal, já que a tabelas eram identificadas apenas por uma chave primária (*movies*) ou tinhas apenas chaves primárias (*genre_movie*, *plot_keyword_movie*)

- **3FN**

Para transformar nosso dataset para a segunda forma normal, foram identificados atributos que possuíam relação de dependência funcional com atributos não-primários, tendo uma dependência funcional por transitividade com o atributo primário. Esses atributos são *director_facebook_likes*, *director_name*, que são informações que estão relacionadas com o diretor, e *actor_1_facebook_likes*, *actor_2_facebook_likes*, *actor_3_facebook_likes*, *actor_1_name*, *actor_2_name*, *actor_3_name*, que estão diretamente relacionadas com o conjunto de atores que estão presentes no filme. Sendo assim, *director_facebook_likes* e *director_name* passam a ser colunas da tabela *director*, e *actor_1_facebook_likes*, *actor_2_facebook_likes*, *actor_3_facebook_likes*, *actor_1_name*, *actor_2_name*, *actor_3_name* tornam-se as colunas *actor_facebook_likes* e *actor_names* da tabela *actor*, e para garantir que os filmes tenham 3 atores e que cada ator possa estar em mais de um filme, foi criada uma tabela relacional contendo *movie_id*, *actor1_id*, *actor2_id* e *actor3_id.*

3.6. Importação do Dataset Externo

O script escrito em python *(locadora_populate.py)* é responsável por importar o Dataset externo e popular as seguintes tabelas do banco de dados:

- filme
- categoria

- ator
- filme_has_categoria
- estrelato

A população dessas tabelas pode ser vista realizando as seguintes consultas:

- SELECT * FROM filme;
- SELECT * FROM categoria
- SELECT * FROM ator;
- SELECT * FROM filme_has_categoria;
- SELECT * FROM estrelato;

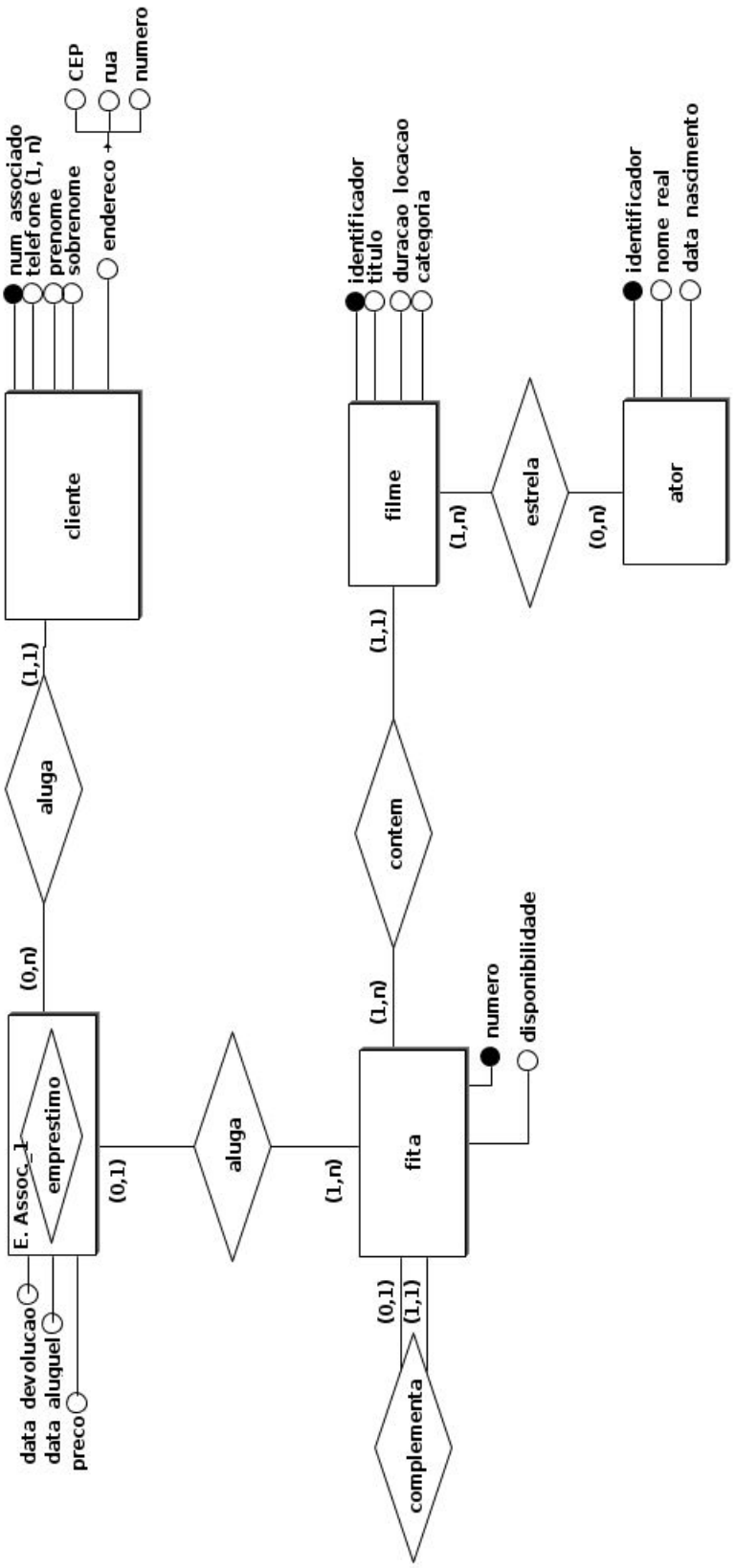
Colocamos no drive um vídeo de demonstração do processo de execução do script de criação do banco e de população das tabelas. O vídeo pode ser assistido no link: https://drive.google.com/drive/folders/113v5KrE3kcL9Qi-4b6pmyTsNj8PLys75

Nota: A base de dados externa vinha com a possibilidade de existir mais de uma categoria por filme. No nosso banco, categoria era um atributo da tabela filme, uma vez que não tínhamos pensado nessa possibilidade. Para melhor importar a base de dados e deixar o nosso banco mais interessante, optamos por criar uma tabela de categorias e uma tabela relacionando os filmes com suas categorias respectivas. Alteramos o nosso script de criação no banco e a representação do modelo físico com o MySQL Workbench, o que pode ser visto na parte dos anexos.

Ver script de criação do banco de dados (*Anexo 4*).

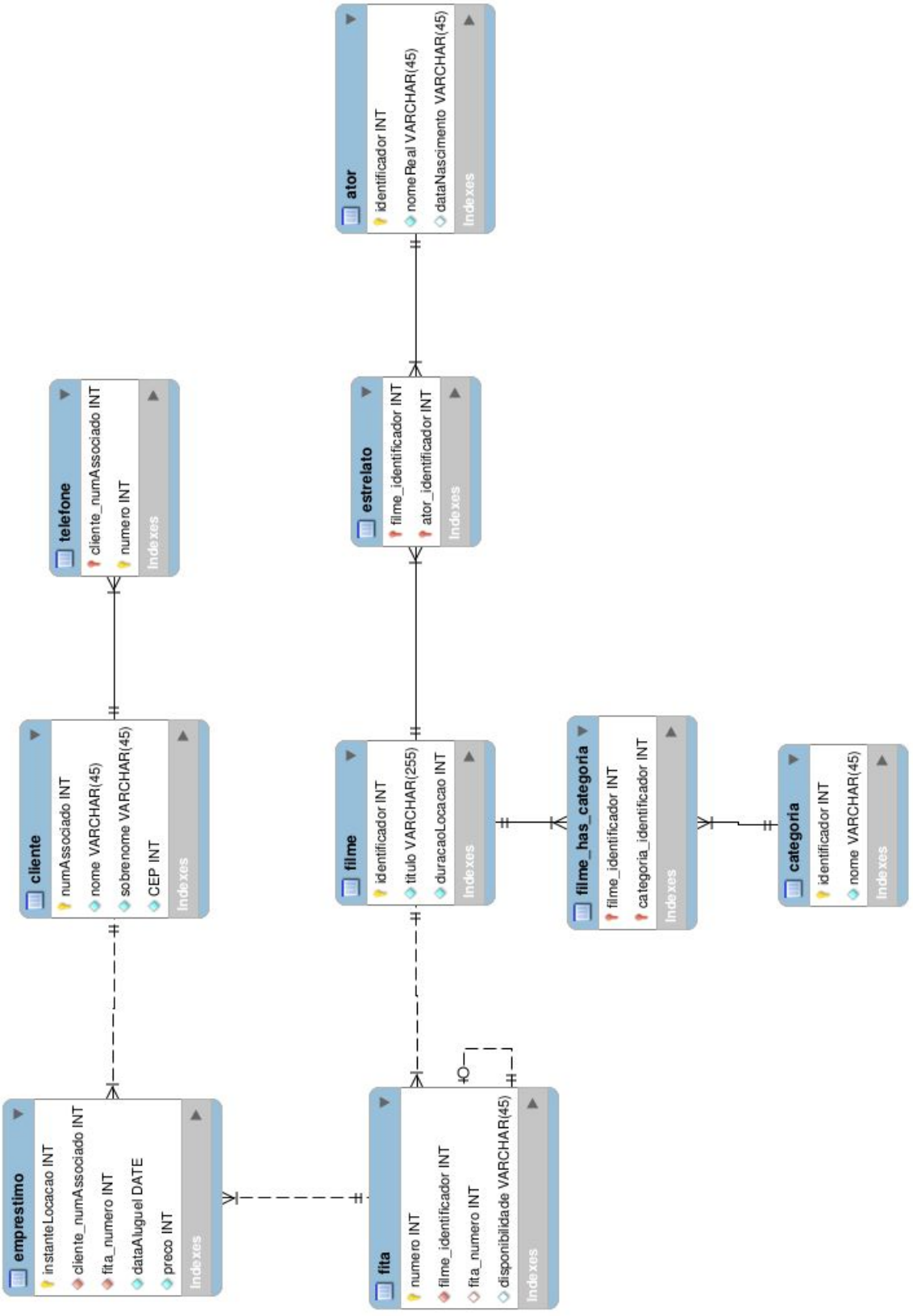
Ver script de população do banco de dados com importação de dataset externo no final do documento (*Anexo 5*).

Anexo 1 - Modelo Conceitual



Anexo 2 - Modelo Lógico

Anexo 3 - Modelo Físico



Anexo 4 - Script de Criação


```
-- MySQL Script generated by MySQL Workbench
-- Sun 11 Nov 2018 11:26:21 PM -03
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema locadora
-- -----------------------------------------------------

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

-- -----------------------------------------------------
-- Table `locadora`.`cliente`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`cliente` (
  `numAssociado` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `sobrenome` VARCHAR(45) NOT NULL,
  `CEP` INT NOT NULL,
  PRIMARY KEY (`numAssociado`),
  UNIQUE INDEX `numAssociado_UNIQUE` (`numAssociado` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`filme`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`filme` (
  `identificador` INT NOT NULL,
  `titulo` VARCHAR(255) NOT NULL,
  `duracaoLocacao` INT NOT NULL,
  PRIMARY KEY (`identificador`),
  UNIQUE INDEX `identificador_UNIQUE` (`identificador` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`fita`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`fita` (
  `numero` INT NOT NULL,
  `filme_identificador` INT NOT NULL,
  `fita_numero` INT NULL,
  `disponibilidade` VARCHAR(45) NULL,
  PRIMARY KEY (`numero`),
  UNIQUE INDEX `numero_UNIQUE` (`numero` ASC),
  INDEX `fk_fita_filme1_idx` (`filme_identificador` ASC),
  INDEX `fk_fita_fita1_idx` (`fita_numero` ASC),
  CONSTRAINT `fk_fita_filme1`
    FOREIGN KEY (`filme_identificador`)
    REFERENCES `locadora`.`filme` (`identificador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_fita_fita1`
    FOREIGN KEY (`fita_numero`)
    REFERENCES `locadora`.`fita` (`numero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`emprestimo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`emprestimo` (
  `instanteLocacao` INT NOT NULL,
  `cliente_numAssociado` INT NOT NULL,
  `fita_numero` INT NOT NULL,
```

```
  `dataAluguel` DATE NOT NULL,
  `preco` INT NOT NULL,
  PRIMARY KEY (`instanteLocacao`),
  UNIQUE INDEX `instanteLocacao_UNIQUE` (`instanteLocacao` ASC),
  INDEX `fk_emprestimo_cliente1_idx` (`cliente_numAssociado` ASC),
  INDEX `fk_emprestimo_fita1_idx` (`fita_numero` ASC),
  CONSTRAINT `fk_emprestimo_cliente1`
    FOREIGN KEY (`cliente_numAssociado`)
    REFERENCES `locadora`.`cliente` (`numAssociado`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_emprestimo_fita1`
    FOREIGN KEY (`fita_numero`)
    REFERENCES `locadora`.`fita` (`numero`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`telefone`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`telefone` (
  `cliente_numAssociado` INT NOT NULL,
  `numero` INT NOT NULL,
  PRIMARY KEY (`cliente_numAssociado`, `numero`),
  CONSTRAINT `fk_telefone_cliente`
    FOREIGN KEY (`cliente_numAssociado`)
    REFERENCES `locadora`.`cliente` (`numAssociado`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`ator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`ator` (
  `identificador` INT NOT NULL,
  `nomeReal` VARCHAR(45) NOT NULL,
  `dataNascimento` VARCHAR(45) NULL,
  PRIMARY KEY (`identificador`),
  UNIQUE INDEX `identificador_UNIQUE` (`identificador` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`estrelato`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`estrelato` (
  `filme_identificador` INT NOT NULL,
  `ator_identificador` INT NOT NULL,
  PRIMARY KEY (`filme_identificador`, `ator_identificador`),
  INDEX `fk_estrelato_ator1_idx` (`ator_identificador` ASC),
  CONSTRAINT `fk_estrelato_filme1`
    FOREIGN KEY (`filme_identificador`)
    REFERENCES `locadora`.`filme` (`identificador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_estrelato_ator1`
    FOREIGN KEY (`ator_identificador`)
    REFERENCES `locadora`.`ator` (`identificador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`categoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`categoria` (
  `identificador` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`identificador`),
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),
  UNIQUE INDEX `identificador_UNIQUE` (`identificador` ASC))
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `locadora`.`filme_has_categoria`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `locadora`.`filme_has_categoria` (
  `filme_identificador` INT NOT NULL,
  `categoria_identificador` INT NOT NULL,
  PRIMARY KEY (`filme_identificador`, `categoria_identificador`),
  INDEX `fk_filme_has_categoria_categoria1_idx` (`categoria_identificador` ASC),
  INDEX `fk_filme_has_categoria_filme1_idx` (`filme_identificador` ASC),
  CONSTRAINT `fk_filme_has_categoria_filme1`
    FOREIGN KEY (`filme_identificador`)
    REFERENCES `locadora`.`filme` (`identificador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_filme_has_categoria_categoria1`
    FOREIGN KEY (`categoria_identificador`)
    REFERENCES `locadora`.`categoria` (`identificador`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Anexo 5 - Script de População do Banco de Dados com Dataset Externo


```
import csv
import MySQLdb
import getpass

# variaveis do ambiente local
path = raw_input("Digite o caminho para o arquivo de metadados: ")
user = raw_input("Digite o usuario de acesso ao banco: ")
password = getpass.getpass("Digite a senha: ")
database = raw_input("Digite o nome do banco: ")
qt_insertions = raw_input("Digite o numero maximo de insercoes: ")

#conexao com o banco
mydb = MySQLdb.connect(host='localhost',
    user=user,
    passwd=password,
    db=database)
cursor = mydb.cursor()

# seta os contadores de ids
count_tiles = 1
count_actors = 1
count_genres = 1

# inicializa o dicionario de todos os generos de filmes e de todos os atores
dict_genres = dict()
dict_actors = dict()

i = 0

with open(path, 'r') as movies_file:

    csv_reader = csv.reader(movies_file)

    # percorre as linhas do arquivo csv
    for row in csv_reader:

        # limita o numero de linhas percorridas ao valor de qt_insertions, para reduzir o tempo de execucao
        if(i == int(qt_insertions)):
            exit(1)
        i+=1

        movie_tile = row[11]
        movie_actors = set([row[6], row[10], row[14]])
        movie_genres = set(row[9].split("|"))

        # insere o titulo dos filmes
        sql = "INSERT INTO filme (identificador, titulo, duracaoLocacao) VALUES (%s, %s, %s)"
        val = (count_tiles, movie_tile, 1)
        cursor.execute(sql,val)
        mydb.commit()

        # percorre o conjunto de generos
        for genre in movie_genres:
            # se o genero ja foi cadastrado
            if(genre in dict_genres):
                # obtem o id referente ao genero
                id_genre = dict_genres[genre]

                # insere na tabela filme_has_categoria
                sql = "INSERT INTO filme_has_categoria (filme_identificador, categoria_identificador) VALUES
                (%s, %s)"
                val = (count_tiles, id_genre)
                cursor.execute(sql,val)

                # salva as mudancas
                mydb.commit()

            # se o genero nao foi cadastrado ainda
            else:
                # insere na tabela categoria
                sql = "INSERT INTO categoria (identificador, nome) VALUES (%s, %s)"
                val = (count_genres, genre)
                cursor.execute(sql,val)

                # insere na tabela filme_has_categoria
                sql = "INSERT INTO filme_has_categoria (filme_identificador, categoria_identificador) VALUES
```

```
                sql = "INSERT INTO filme_has_categoria (filme_identificador, categoria_identificador) VALUES
                (%s, %s)"
                val = (count_tiles, count_genres)
                cursor.execute(sql,val)

                # salva as mudancas
                mydb.commit()

                # atualiza valores
                dict_genres[genre] = count_genres
                count_genres+=1

        # percorre o conjunto de atores
        for actor in movie_actors:
            # se o ator ja foi cadastrado
            if(actor in dict_actors):
                # obtem o id referente ao ator
                id_actor = dict_actors[actor]

                # insere na tabela estrelato
                sql = "INSERT INTO estrelato (filme_identificador, ator_identificador) VALUES (%s, %s)"
                val = (count_tiles, id_actor)
                cursor.execute(sql,val)

                # salva as mudancas
                mydb.commit()

            # se o ator nao foi cadastrado ainda
            else:
                # insere na tabela ator
                sql = "INSERT INTO ator (identificador, nomeReal) VALUES (%s, %s)"
                val = (count_actors, actor)
                cursor.execute(sql,val)

                # insere na tabela estrelato
                sql = "INSERT INTO estrelato (filme_identificador, ator_identificador) VALUES (%s, %s)"
                val = (count_tiles, count_actors)
                cursor.execute(sql,val)

                # salva as mudancas
                mydb.commit()

                # atualiza valores
                dict_actors[actor] = count_actors
                count_actors+=1

        # incrementa o contador de filmes
        count_tiles+=1

# encerra a conexao com o banco
cursor.close()
```