Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Projeto: Filmes e Séries – Parte 1, 2 e 3

Sistema de Comercialização e Distribuição de Filmes e Séries pela Internet

Disciplina: SCC0240 Bases de Dados
Profª: Elaine Parros M. de Sousa

Aluno	**Nº USP**
Elias Italiano Rodrigues	7987251
Rafael Andreatta Martins	7564019
Rodolfo Megiato de Lima	7987286
Vinicius Katata Biondo	6783972

Data de entrega: 24/06/2014

Descrição do Problema

Uma empresa pioneira teve a ideia de desenvolver um sistema de comercialização e distribuição de filmes e seriados na Internet, seguindo o modelo de funcionamento dos conhecidos sistemas Steam[1] e PlayStation Network[2], que estão envolvidos com a venda de jogos eletrônicos. Integrado a esse sistema, existe uma plataforma de interação entre os usuários que permite o envio de mensagens, avaliação dos filmes e séries, criação de listas e outras atividades sociais.

O sistema permite a inserção de filmes e séries que são disponibilizados pelas distribuidoras para serem comercializados de forma virtual. Além disso, os usuários têm seus perfis cadastrados para interagirem de acordo com seus interesses, podendo criar playlists, wishlists e reviews, compartilhar informações sobre os filmes e séries, avaliar os vídeos assistidos e buscar por mais detalhes sobre cinema.

O cadastro de vídeos é realizado a partir das licenças concedidas pelas distribuidoras para a comercialização digital dos filmes e séries. As informações referentes aos vídeos são: título original (que identifica unicamente o filme juntamente com o seu ano de lançamento), título em português, gênero, classificação etária, duração em minutos, tamanho em GB, avaliação média (em uma nota de 0,0 à 10,0), sinopse, trailer, qualidade do vídeo (resolução), premiações, áudio, legendas e o preço em reais. Os vídeos são divididos em apenas dois tipos: filmes e séries, sendo que séries possuem, além das informações dos vídeos, o número e nome do episódio, número da temporada a que pertencem e os atores convidados. Além das informações sobre os vídeos, o sistema armazena dados sobre os artistas, que possuem nome artístico (que o identifica unicamente), nome verdadeiro e podem ser classificados em dois tipos: atores e diretores.

Os usuários do sistema necessitam ser cadastrados e autenticados com nome de usuário (que o identifica de maneira única no sistema) e senha para que possam acessar as áreas privadas que lhes permite comprar os vídeos utilizando um cartão de crédito e realizar os downloads (ou assistí-los online). Outros dados armazenados sobre os usuários do sistema são: nome completo, data de nascimento e e-mail. Os usuários podem interagir em uma espécie de rede social em que podem ser compartilhadas informações através de mensagens enviadas para seus contatos (previamente adicionados) e por meio de posts publicados em seu blog pessoal. O usuário pode também criar listas de filmes, que possuem como atributos, data e hora em que foram criadas e que podem ser apenas de dois tipos: wishlist (para planejar suas futuras compras) e playlist (para organizar e assistir seus vídeos já adquiridos). Estas listas podem ficar visíveis para os outros usuários, onde cada uma delas serve como informação dos filmes preferidos de cada usuário. As playlists, além dos dados comuns às listas, possuem também um nome e se são públicas ou não. Para cada filme comprado, pode ser redigido apenas um review e uma avaliação por usuário do sistema.

As mensagens trocadas pelos usuários do sistema, podem ser: mensagens do chat (diretas entre os usuários) e posts em seus blogs pessoais. As mensagens do chat possuem data e hora em que foram escritas, conteúdo e data e hora em que foram visualizadas. Já os posts apresentam: data e hora em que foram escritos, título, categoria, conteúdo, número de compartilhamentos realizados a partir dele e se é público ou não.

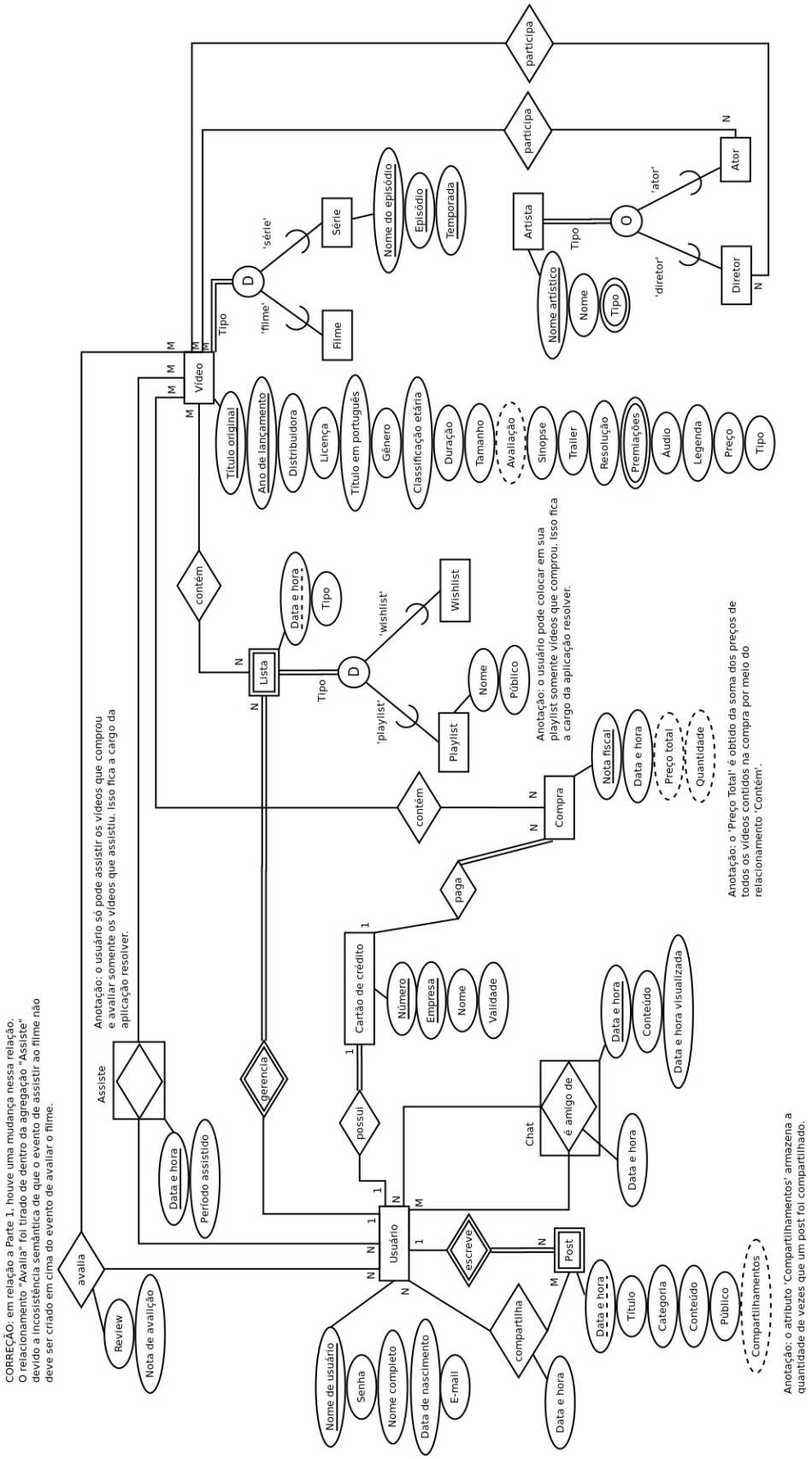
Por meio do cartão de crédito, previamente cadastrado com o nome igual escrito no cartão, número, validade e empresa, o usuário faz a compra de vídeos incluindo um ou mais filmes/séries em seu carrinho de compras. A cada compra realizada, o sistema armazena a número da nota fiscal (que identifica de forma única a compra), a data e hora em que foi realizada e calcula seu preço total a partir dos itens que compõem a compra. A partir do momento que o vídeo é comprado pelo usuário, o vídeo fica disponível para ser assistido no player do próprio sistema ou, se o usuário preferir, pode realizar o download e executá-lo externamente. Se a opção for pela escolha do player integrado do sistema, que se tornou muito comum no modelo de Internet atual, são armazenadas informações sobre os períodos do dia em que os usuários assistem seus vídeos.

De acordo com os vídeos comprados pelo usuário no sistema, são armazenadas todas as compras, podendo então, ser extraídas informações sobre os seus gêneros de filmes preferidos, atores e diretores. Além dos dados da compra, quando o usuário assiste os vídeos, o período da visualização deste também serve de informação para um sistema de recomendação presente na plataforma.

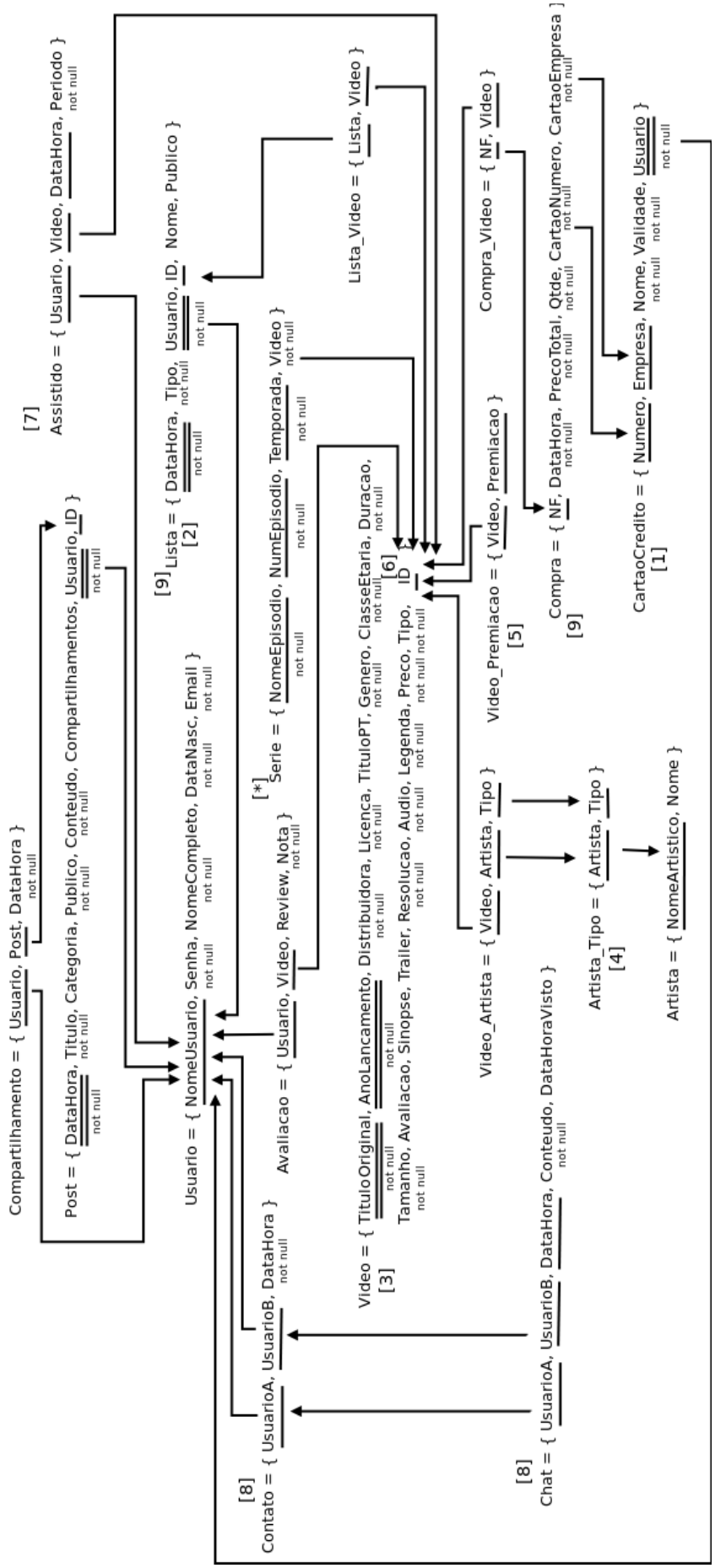
É possível pesquisar sobre os filmes pelo nome, ano de lançamento, gênero, atores e diretores participantes, assim como pesquisar episódios de seriados, listar todos os episódios de uma temporada e listar todas as temporadas de um seriado. No resultado das buscas, além de serem exibidos os links para a comercialização do vídeo pesquisado, serão exibidas playlists públicas que o contenha, assim fazendo com que os usuários tenham uma maior interação, já que o dono da playlist deve possuir gostos parecedidos.

[1] Steam: http://store.steampowered.com/
[2] PlayStation Network: http://br.playstation.com/psn/

Diagrama Entidade-Relacionamento



Modelo Relacional



Notas sobre o Modelo Relacional

[1] Fazer o mapeamento do relacionamento 1-1 criando uma nova tabela não garantiria a participação total do Cartão de Crédito. Se mapeado colocando-se o Cartão de Crédito dentro da tabela Usuário, criaria uma participação total inexistente no MER. E no caso de mapear o Usuário inteiramente (com todos os atributos) dentro da tabela Cartão de Crédito, não faria sentido semântico.

[2] Optou-se por esse mapeamento da especialização de Lista, pois nenhuma das entidades que especializam Lista participam em relacionamentos. Além disso, existem poucos atributos que as diferenciam e espera-se que existam poucas instâncias de Wishlist então não sendo um problema a quantidade de instâncias com valor null nos atributos 'Nome' e 'Público'.

[3] Como o filme não possui atributos específicos, foi mapeado somente a entidade Série em uma outra tabela ficando a cargo do atributo 'Tipo' de Video defini-lo como filme ou Série.

[4] Mapear as duas entidades, Ator e Diretor, separadamente apenas copiando os atributos de Artista traria redundância, pois há a possibilidade de existir um mesmo artista que é ator e diretor. Além disso, para mapear o relacionamento entre Artista e Video, seria necessário criar outras duas tabelas, uma para ator e outra para diretor.

[5] Optou-se por mapear o atributo 'Premiações' em uma nova tabela pois não se sabe a priori quantas premiações existem para filmes.

[6] Os IDs foram criados para diminuir os gastos com armazenamento das chaves estrangeiras nas tabelas.

[7] A agregação Assiste foi mapeada na tabela Assistido de modo que implementa o relação N—M e permite repetições em que um mesmo Usuario assista ao mesmo vídeo em horário diferentes.

[8] A agregação Chat sobre o relacionamento “é amigo de” foi mapeada da seguinte maneira: a tabela Contato implementa o relacionamento N—M “é amigo de” entre dois Usuários e a tabela Chat é responsável por guardar as mensagens trocadas entre dois usuários que já tem contato estabelecido, portanto, puxa os usuários como chave estrangeira da tabela Contato. Isso garante a consistência de que somente ocorra troca de mensagens entre usuários que adicionaram um ao outro como contato.

[9] O mapaemento da relação 1—N com participação total no lado de N é garantida nesse esquema da seguinte maneira: a chave estrangeira Usuario em Lista (ou CartaoNumero em Compra) é definida como “not null” forçando a participação total, pois é necessário que exista um Usuario para que se crie uma Lista (assim como um Cartao para que se crie uma Compra). O 1—N se dá pelas chaves estrangeiras que não limitam a quantidade de tuplas com o mesmo valor nesse campo, no caso, um Usuário pode criar várias Listas desde que em data/hora diferentes (e um Cartao pode realizar várias Compras).

[*] Correção no mapeamento da tabela Série: durante a criação do SQL das tabelas, notou-se um erro em Série que impedia que mais de um episódio de um mesmo seriado fosse inserido. Isso seu deu por causa de um erro ao definir as chaves: antes Video era chave primária em Série, o que impedia que um mesmo Vídeo tenha vários episódios de série gravados. A nova correção está de acordo com o planejado.

Implementação da base de dados em um protótipo de software operacional com SGBD

A implementação do protótipo do sistema foi realizada utilizando o sistema operacional GNU/Linux, o servidor web Apache 2.2, o SGBD relacional MySQL 5.5 e linguagem de programação PHP 5.4. Trata-se um sistema web acessado por meio de uma navegador de Internet (browser) em que o usuário preenche formulários e clica em botões que inserem, alteram, excluem ou pesquisam dados no banco de dados. A instalação e configuração dos softwares necessários foi feita em nossas máquinas locais sem necessidade de hospedar em algum servidor web na Internet.

O código-fonte do protótipo será entregue em formato .zip pelo Tidia-Ae da USP no escaninnho do aluno Elias Italiano Rodrigues.

Os scripts de criação do banco de dados juntamente com as inserções iniciais estão em anexo neste documento.

Trechos de código que implementam consultas SQL

```
// Classe controler que gerencia a tabela a USUARIO
class Controller_Usuario
{

	// Seleciona todos os dados para listagem dos usuários cadastrados
	public function index()
	{
		$db = Database::getInstance();
		$entries = $db->fetch('SELECT * FROM USUARIO');
		return
		[
			'entries' => $entries,
		];

	}


	// Cadastra um usuário no banco de dados
	public function add()
	{
		[...]

		if(count($_POST) > 0)
		{
			$query = 'INSERT INTO USUARIO(';
			foreach($formFields as $name => $label)
				$query .= "$name,";
			$query = rtrim($query, ',');
			$query .= ') VALUES(';
```

```
                foreach($formFields as $name => $label)
                        $query .= ":$name,";
                $query = rtrim($query, ',');
                $query .= ')';
                $success = (bool) $db->executeStmt($query, $_POST);
        }


    // Seleciona um usuário específico para atualização de seus dados
    public function edit()
    {
        [...]

        $entry = $db->fetch('SELECT * FROM USUARIO WHERE NOMEUSUARIO =
:nomeusuario', ['nomeusuario' => $nomeusuario]);

        […]
        // Atualiza os dados do usuário específico sendo editado
        $query = 'UPDATE USUARIO SET ';
        foreach($formFields as $name => $label)
                $query .= "$name = :$name, ";
        $query = rtrim($query, ', ');
        $query .= " WHERE NOMEUSUARIO = $nomeusuario";
        $success = $db->executeStmt($query, $_POST);

    // Exclui um usuário do banco de dados
    // Que na verdade, a exclusão como não é definitiva, apenas atualiza o registro do
usuário definindo o campo STATUS = 'x'
    public function delete()
    {
        $db = Database::getInstance();
        $nomeusuario = $_GET['nomeusuario'];
        $success = $db->executeStmt('UPDATE FROM USUARIO SET STATUS = \'x\' WHERE
NOMEUSUARIO = :nomeusuario', ['nomeusuario' => $nomeusuario]);
        return [];
    }

[…]
}

// Consultas SQL de busca de filmes e séries para compra

// Busca de vídeo por título
SELECT * FROM VIDEOS V WHERE V.TITULOORIGINAL='$NOME'

// Busca de víde por ano de lançamento
SELECT V.TITULOORIGINAL FROM VIDEOS V WHERE V.ANOLANCAMENTO='$ANO'

// Busca de vídeo considerando seu gênero
SELECT V.TITULOORIGINAL FROM VIDEOS V WHERE V.GENERO='$GENERO'

// Mostra todos os episódios de uma série de uma certa temporada
SELECT S.NOMEEPISODIO, S.NUMEPISODIO, S.TEMPORADA FROM SERIE S JOIN VIDEO V ON S.VIDEO =
V.ID WHERE V.TITULOORIGINAL='$TITULO' AND S.TEMPORADA='$TEMPORADA';

// Seleciona todas as temporadas de uma série
SELECT DISTINCT TEMPORADA FROM SERIE S JOIN VIDEO V ON S.VIDEO = V.ID WHERE
V.TITULOORIGINAL='$TITULO'

// Após selecionados os videos são listadas as playlists em que os vídeos aparecem
SELECT V.TITULOORIGINAL, L.USUARIO, L.DATAHORA, L.NOME FROM LISTA L JOIN LISTA_VIDEO LV
ON LV.LISTA = L.ID JOIN VIDEO_ARTISTA VA ON VA.VIDEO = LV.VIDEO JOIN VIDEO V ON VA.VIDEO
= V.ID WHERE VA.ARTISTA = '$ARTISTA';
```

```
// Seleciona todos os videos que não receberam nenhuma nota abaixo de 5, diferente de
não ter a média menor que cinco
SELECT V.TITULOORIGINAL, V.ID FROM VIDEO V WHERE V.ID NOT IN (SELECT A.VIDEO FROM
AVALIACAO A WHERE A.NOTA < 5.00);
```

Conclusão

O projeto, de maneira geral, conseguiu abranger todos os conceitos aprendidos durante a disciplina, fazendo com que os alunos projetassem uma pequena base de dados desde o princípio. Como o projeto foi evoluindo junto com o andamento do curso, os modelos foram sendo feitos no decorrer do aprendizado, o que facilitou o processo. A implementação do protótipo foi a parte mais complicada do projeto, já que utilizamos ferramentas de desenvolvimento web que não estamos tão familiarizados, além de ser a última parte do projeto, onde não temos mais tanto tempo devido o final do semestre.

Anexo

Scripts de criação da base de dados e inserções iniciais

```
-- MySQL das tabelas

-- Character : utf8
-- Collate   : utf8_general_ci (character encoding com suporte a acentuação)
-- Engine    : InnoDB (suporta modelo ACID, com transações, commit, rollback e
recuperação de falhas)

-- Criação do banco de dados com escolha do encoding de caracteres
CREATE DATABASE BD_T3 DEFAULT CHARACTER SET UTF8 COLLATE UTF8_GENERAL_CI;

-- Seleciona o banco a ser usado
USE BD_T3;

-- Tabela Usuario
CREATE TABLE USUARIO(
       NOMEUSUARIO  VARCHAR(20) NOT NULL,
       SENHA        VARCHAR(40) NOT NULL,
       NOMECOMPLETO VARCHAR(40) NOT NULL,
       DATANASC     DATE        NOT NULL, -- formato 'yyyy-mm-dd'
       EMAIL        VARCHAR(40) NOT NULL,
       STATUS       CHAR(1)     DEFAULT 'a', -- a: ativo, x: excluído, b: bloqueado. Esse
campo foi adicionado posteriormente pois decidiu-se não excluir definitivamente as
tuplas dos usuários.

       PRIMARY KEY(NOMEUSUARIO),

       CHECK(STATUS IN ('a', 'x', 'b')) -- Check para garantir a integridade nesse campo
que funciona como uma flag
) ENGINE=InnoDB;

-- Tabela Video
CREATE TABLE VIDEO(
       ID             INT UNSIGNED  NOT NULL AUTO_INCREMENT, -- o tipo de dado int
unsigned foi escolhido para suportar uma grande quantidade de ids. auto_increment
automaticamente incrementa a id da ultima tupla e utiliza na insercao da seguinte
       TITULOORIGINAL VARCHAR(60)   NOT NULL,
       ANOLANCAMENTO  SMALLINT      NOT NULL,
       DISTRIBUIDORA  VARCHAR(40)   NOT NULL,
       LICENCA        VARCHAR(40)   NOT NULL,
       TITULOPT       VARCHAR(60),
       GENERO         VARCHAR(40)   NOT NULL,
       CLASSEETARIA   SMALLINT      NOT NULL, -- idade mínima
       DURACAO        SMALLINT      NOT NULL, -- em minutos
       TAMANHO        INT UNSIGNED  NOT NULL, -- em MB aproximadamente
       AVALIACAO      DECIMAL(2,1), -- média das notas de avaliação recebidas
       SINOPSE        VARCHAR(200),
       TRAILER        VARCHAR(100), -- URL do trailer do vídeo
       RESOLUCAO      VARCHAR(20)   NOT NULL,
       AUDIO          VARCHAR(100)  NOT NULL,
       LEGENDA        VARCHAR(100)  NOT NULL,
       PRECO          DECIMAL(5,2)  NOT NULL,
       TIPO           CHAR(1)       NOT NULL, -- 'f' filme, 's' série

       PRIMARY KEY(ID),
       UNIQUE(TITULOORIGINAL, ANOLANCAMENTO),

       CHECK(TIPO IN ('f', 's'))
) ENGINE=InnoDB;
```

```
-- Tabela Serie
CREATE TABLE SERIE(
	NOMEEPISODIO VARCHAR(40)  NOT NULL,
	NUMEPISODIO  TINYINT      NOT NULL, -- inteiro muito pequeno para armazenar os
numeros de episodios
	TEMPORADA    TINYINT      NOT NULL, -- inteiro muito pequeno para armazenar as
temporadas
	VIDEO        INT UNSIGNED NOT NULL,

	PRIMARY KEY(NOMEEPISODIO, NUMEPISODIO, TEMPORADA),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

-- Tabela Video_Premiacao
CREATE TABLE VIDEO_PREMIACAO(
	VIDEO     INT UNSIGNED NOT NULL,
	PREMIACAO VARCHAR(40)  NOT NULL,

	PRIMARY KEY(VIDEO, PREMIACAO),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

-- Tabela Avaliacao
CREATE TABLE AVALIACAO(
	USUARIO VARCHAR(20)  NOT NULL,
	VIDEO   INT UNSIGNED NOT NULL,
	REVIEW  TEXT, -- campo que permite longas strings de texto
	NOTA    DECIMAL(2,1) NOT NULL,

	PRIMARY KEY(USUARIO, VIDEO),
	FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

-- Tabela Contato
-- Ao ser inserido um contato de A com B a aplicação se encarrega de criar o contato
contrário (B com A)
CREATE TABLE CONTATO(
	USUARIOA VARCHAR(20) NOT NULL,
	USUARIOB VARCHAR(20) NOT NULL,
	DATAHORA DATETIME    NOT NULL, -- formato 'yyyy-mm-dd hh:mm:ss'

	PRIMARY KEY(USUARIOA, USUARIOB),
	FOREIGN KEY(USUARIOA) REFERENCES USUARIO(NOMEUSUARIO),
	FOREIGN KEY(USUARIOB) REFERENCES USUARIO(NOMEUSUARIO)
) ENGINE=InnoDB;

-- Tabela Chat
-- Tabela que guarda as mensagens do usuario A para B
CREATE TABLE CHAT(
	USUARIOA  VARCHAR(20) NOT NULL,
	USUARIOB  VARCHAR(20) NOT NULL,
	DATAHORA  DATETIME    NOT NULL,
	CONTEUDO  TEXT        NOT NULL,
	DATAVISTO DATETIME,

	PRIMARY KEY(USUARIOA, USUARIOB, DATAHORA),
	FOREIGN KEY(USUARIOA, USUARIOB) REFERENCES CONTATO(USUARIOA, USUARIOB)
) ENGINE=InnoDB;

-- Tabela Artista
CREATE TABLE ARTISTA(
	NOMEARTISTICO VARCHAR(40) NOT NULL,
	NOME          VARCHAR(40),

	PRIMARY KEY(NOMEARTISTICO)
) ENGINE=InnoDB;
```

```
-- Tabela Artista_Tipo
CREATE TABLE ARTISTA_TIPO(
       ARTISTA VARCHAR(40) NOT NULL,
       TIPO    VARCHAR(20) NOT NULL,

       PRIMARY KEY(ARTISTA, TIPO),
       FOREIGN KEY(ARTISTA) REFERENCES ARTISTA(NOMEARTISTICO)
) ENGINE=InnoDB;

-- Tabela Video_Artista
CREATE TABLE VIDEO_ARTISTA(
       VIDEO   INT UNSIGNED NOT NULL,
       ARTISTA VARCHAR(40)  NOT NULL,
       TIPO    VARCHAR(20)  NOT NULL,

       PRIMARY KEY(VIDEO, ARTISTA, TIPO),
       FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID),
       FOREIGN KEY(ARTISTA, TIPO) REFERENCES ARTISTA_TIPO(ARTISTA, TIPO)
) ENGINE=InnoDB;

-- Tabela Post
CREATE TABLE POST(
       ID                INT UNSIGNED NOT NULL AUTO_INCREMENT,
       DATAHORA          DATETIME     NOT NULL,
       USUARIO           VARCHAR(20)  NOT NULL,
       TITULO            VARCHAR(40)  NOT NULL,
       CATEGORIA         VARCHAR(40),
       PUBLICO           CHAR(1)      DEFAULT 'n', -- marca se o post por ser visto por
outros usuários 'n' - nao e 'y' - sim
       CONTEUDO          TEXT         NOT NULL,
       COMPARTILHAMENTOS INT UNSIGNED DEFAULT 0, -- quando se cria um post, ele não tem
nenhum compartilhamento

       PRIMARY KEY(ID),
       UNIQUE(DATAHORA, USUARIO),
       FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO),

       CHECK(PUBLICO IN ('n', 'y'))
) ENGINE=InnoDB;

-- Tabela Compartilhamentos
-- Quando um usuario faz um compartilhamento de um post a aplicação encarrega-se de
incrementar o numero de compartilhamentos do respectivo post
CREATE TABLE COMPARTILHAMENTO(
       USUARIO  VARCHAR(20)  NOT NULL,
       POST     INT UNSIGNED NOT NULL,
       DATAHORA DATETIME     NOT NULL,

       PRIMARY KEY(USUARIO, POST),
       FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO),
       FOREIGN KEY(POST) REFERENCES POST(ID)
) ENGINE=InnoDB;

-- Tabela CartaoCredito
CREATE TABLE CARTAOCREDITO(
       NUMERO   VARCHAR(20) NOT NULL,
       EMPRESA  VARCHAR(40) NOT NULL,
       NOME     VARCHAR(40) NOT NULL,
       VALIDADE DATE        NOT NULL,
       USUARIO  VARCHAR(20) NOT NULL,

       PRIMARY KEY(NUMERO, EMPRESA),
       UNIQUE(USUARIO),
       FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO)
) ENGINE=InnoDB;
```

```
-- Tabela Compra
CREATE TABLE COMPRA(
	NF		INT UNSIGNED NOT NULL AUTO_INCREMENT,
	DATAHORA	DATETIME	NOT NULL,
	PRECOTOTAL	DECIMAL(5,2) NOT NULL,
	QTDE		SMALLINT	NOT NULL,
	CARTAONUMERO	VARCHAR(20)	NOT NULL,
	CARTAOEMPRESA	VARCHAR(40)	NOT NULL,

	PRIMARY KEY(NF),
	FOREIGN KEY(CARTAONUMERO, CARTAOEMPRESA) REFERENCES CARTAOCREDITO(NUMERO, EMPRESA)
) ENGINE=InnoDB;

-- Tabela Compra_Video
CREATE TABLE COMPRA_VIDEO(
	NF	INT UNSIGNED NOT NULL,
	VIDEO	INT UNSIGNED NOT NULL,

	PRIMARY KEY(NF, VIDEO),
	FOREIGN KEY(NF) REFERENCES COMPRA(NF),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

-- Tabela Lista
CREATE TABLE LISTA(
	ID		INT UNSIGNED NOT NULL AUTO_INCREMENT,
	DATAHORA	DATETIME	NOT NULL,
	USUARIO		VARCHAR(20)	NOT NULL,
	TIPO		CHAR(1)		DEFAULT 'p', -- 'p' playlist, 'w' wishlist
	NOME		VARCHAR(40),
	PUBLICO		CHAR(1)		DEFAULT 'n', -- marca se a lista pode ser vista por outros
usuários 'n' - nao e 'y' sim

	PRIMARY KEY(ID),
	UNIQUE(DATAHORA, USUARIO),
	FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO),
	CHECK(TIPO IN ('p', 'w')),
	CHECK(PUBLICO IN ('n', 'y'))
) ENGINE=InnoDB;

-- Tabela Lista_Video
CREATE TABLE LISTA_VIDEO(
	LISTA	INT UNSIGNED NOT NULL,
	VIDEO	INT UNSIGNED NOT NULL,

	PRIMARY KEY(LISTA, VIDEO),
	FOREIGN KEY(LISTA) REFERENCES LISTA(ID),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

-- Tabela Assistido
CREATE TABLE ASSISTIDO(
	USUARIO		VARCHAR(20)	NOT NULL,
	VIDEO		INT UNSIGNED NOT NULL,
	DATAHORA	DATETIME	NOT NULL,
	PERIODO		VARCHAR(40)	NOT NULL,

	PRIMARY KEY(USUARIO, VIDEO, DATAHORA),
	FOREIGN KEY(USUARIO) REFERENCES USUARIO(NOMEUSUARIO),
	FOREIGN KEY(VIDEO) REFERENCES VIDEO(ID)
) ENGINE=InnoDB;

COMMIT;

-- Inserção de tuplas iniciais nas tabelas

INSERT INTO USUARIO(NOMEUSUARIO, SENHA, NOMECOMPLETO, DATANASC, EMAIL, STATUS) VALUES
```

```
('joao',  'nohash0', 'João da Silva',  '1993-06-10', 'joao@example.com',  DEFAULT),
('maria', 'nohash1', 'Maria da Selva', '1991-07-01', 'selva@example.com', DEFAULT);
COMMIT;

INSERT INTO VIDEO(ID, TITULOORIGINAL, ANOLANCAMENTO, DISTRIBUIDORA, LICENCA, TITULOPT,
GENERO, CLASSEETARIA, DURACAO, TAMANHO, AVALIACAO, SINOPSE, TRAILER, RESOLUCAO, AUDIO,
LEGENDA, PRECO, TIPO) VALUES
(1, 'The Silence of the Lambs', 1991, 'Orion Pictures', 'all', 'O Silêncio dos
Inocentes', 'suspense', 18, 118, 3200, NULL, NULL, NULL, 'full hd', 'en', 'pt-br', 9.00,
'f'),
(2, 'The Lord of the Rings: The Fellowship of', 2001, 'New Line Cinema', 'all', 'O
Senhor dos Anéis: A Sociedade do Anel', 'Fantasia', 14, 178, 5100, NULL, NULL, NULL,
'full hd', 'en', 'pt-br', 10.00, 'f'),
(3, 'Breaking Bad', 2008, 'AXN', 'all', 'Breaking Bad: A Química do Mal', 'drama', 18,
47, 1000, NULL, NULL, NULL, 'full hd', 'en', 'pt-br', 3.00, 's');
COMMIT;

INSERT INTO SERIE(NOMEEPISODIO, NUMEPISODIO, TEMPORADA, VIDEO) VALUES
('Pilot', 1, 1, 3),
('Cat\'s in the Bag...', 2, 1, 3);
COMMIT;

INSERT INTO VIDEO_PREMIACAO(VIDEO, PREMIACAO) VALUES
(2, 'Oscar'),
(2, 'BAFTA');
COMMIT;

INSERT INTO AVALIACAO(USUARIO, VIDEO, REVIEW, NOTA) VALUES
('joao', 2, NULL, 10.0),
('maria', 2, NULL, 5.0);
COMMIT;

INSERT INTO CONTATO(USUARIOA, USUARIOB, DATAHORA) VALUES
('joao', 'maria', '2014-06-01 11:00:00'),
('maria', 'joao', '2014-06-02 11:00:00');
COMMIT;

INSERT INTO CHAT(USUARIOA, USUARIOB, DATAHORA, CONTEUDO, DATAVISTO) VALUES
('joao', 'maria', '2014-06-03 10:00:00', 'Oi, como vai?', '2014-06-03 10:00:01'),
('maria', 'joao', '2014-06-03 10:00:01', 'Bem, e você?', NULL);
COMMIT;

INSERT INTO ARTISTA(NOMEARTISTICO, NOME) VALUES
('Elijah Wood', NULL),
('Ian McKellen', NULL);
COMMIT;

INSERT INTO ARTISTA_TIPO(ARTISTA, TIPO) VALUES
('Elijah Wood', 'ator'),
('Ian McKellen', 'ator');
COMMIT;

INSERT INTO VIDEO_ARTISTA(VIDEO, ARTISTA, TIPO) VALUES
(2, 'Elijah Wood', 'ator'),
(2, 'Ian McKellen', 'ator');
COMMIT;

INSERT INTO POST(ID, DATAHORA, USUARIO, TITULO, CATEGORIA, PUBLICO, CONTEUDO,
COMPARTILHAMENTOS) VALUES
(1, '2014-06-20 10:57:00', 'joao', 'S2 LOTR S2', NULL, 'y', 'Eu amo Senhor dos Anéis
<3\n', DEFAULT),
(2, '2014-06-20 11:00:00', 'maria', 'Elfos', NULL, 'y', 'Queria ser um elfo :3\n',
DEFAULT);
COMMIT;

INSERT INTO COMPARTILHAMENTO(USUARIO, POST, DATAHORA) VALUES
('joao', 2, '2014-06-20 11:10:00'),
```

```
('maria', 1, '2014-06-20 11:00:00');
COMMIT;

INSERT INTO CARTAOCREDITO(NUMERO, EMPRESA, NOME, VALIDADE, USUARIO) VALUES
('123456789', 'Itau', 'Joao da Silva', '2016-06-00', 'joao'),
('123456780', 'Bradesco', 'Maria da Selva', '2018-06-00', 'maria');
COMMIT;

INSERT INTO COMPRA(NF, DATAHORA, PRECOTOTAL, QTDE, CARTAONUMERO, CARTAOEMPRESA) VALUES
(1, '2014-06-01 07:00:00', '10.0', 1, '123456789', 'Itau'),
(2, '2014-06-01 07:00:00', '10.0', 1, '123456780', 'Bradesco');
COMMIT;

INSERT INTO COMPRA_VIDEO(NF, VIDEO) VALUES
(1, 2),
(2, 2);
COMMIT;

INSERT INTO LISTA(ID, DATAHORA, USUARIO, TIPO, NOME, PUBLICO) VALUES
(1, '2014-06-02 03:00:00', 'joao', DEFAULT, 'Meus vídeos', DEFAULT),
(2, '2014-06-02 23:00:00', 'maria', DEFAULT, 'Meus vídeos', DEFAULT);
COMMIT;

INSERT INTO LISTA_VIDEO(LISTA, VIDEO) VALUES
(1, 2),
(2, 2);
COMMIT;

INSERT INTO ASSISTIDO(USUARIO, VIDEO, DATAHORA, PERIODO) VALUES
('joao', 2, '2014-06-03 10:00:00', 'integral'),
('joao', 2, '2014-06-04 09:30:00', 'integral'),
('maria', 2, '2014-06-03 11:10:00', 'integral'),
('maria', 2, '2014-06-04 05:30:00', 'integral');
COMMIT;
```