Universidade Federal de Viçosa – *Campus* Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas
Sistemas de Informação
SIN 320 – Laboratório de Banco de Dados

STREAMBERRY

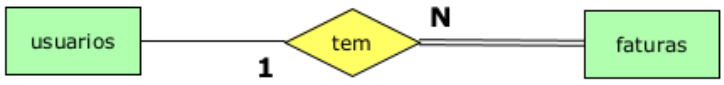
Integrantes

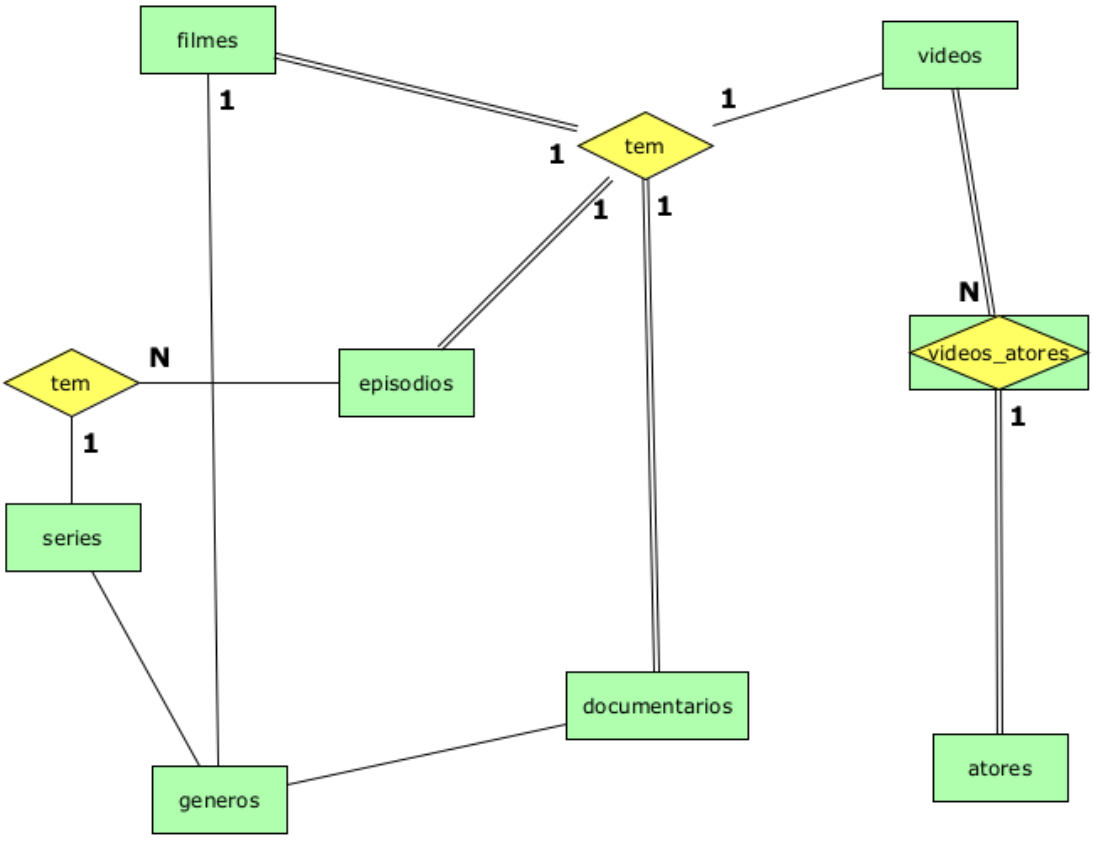
Reinaldo Gonçalves Pereira Neto - 7627

2023

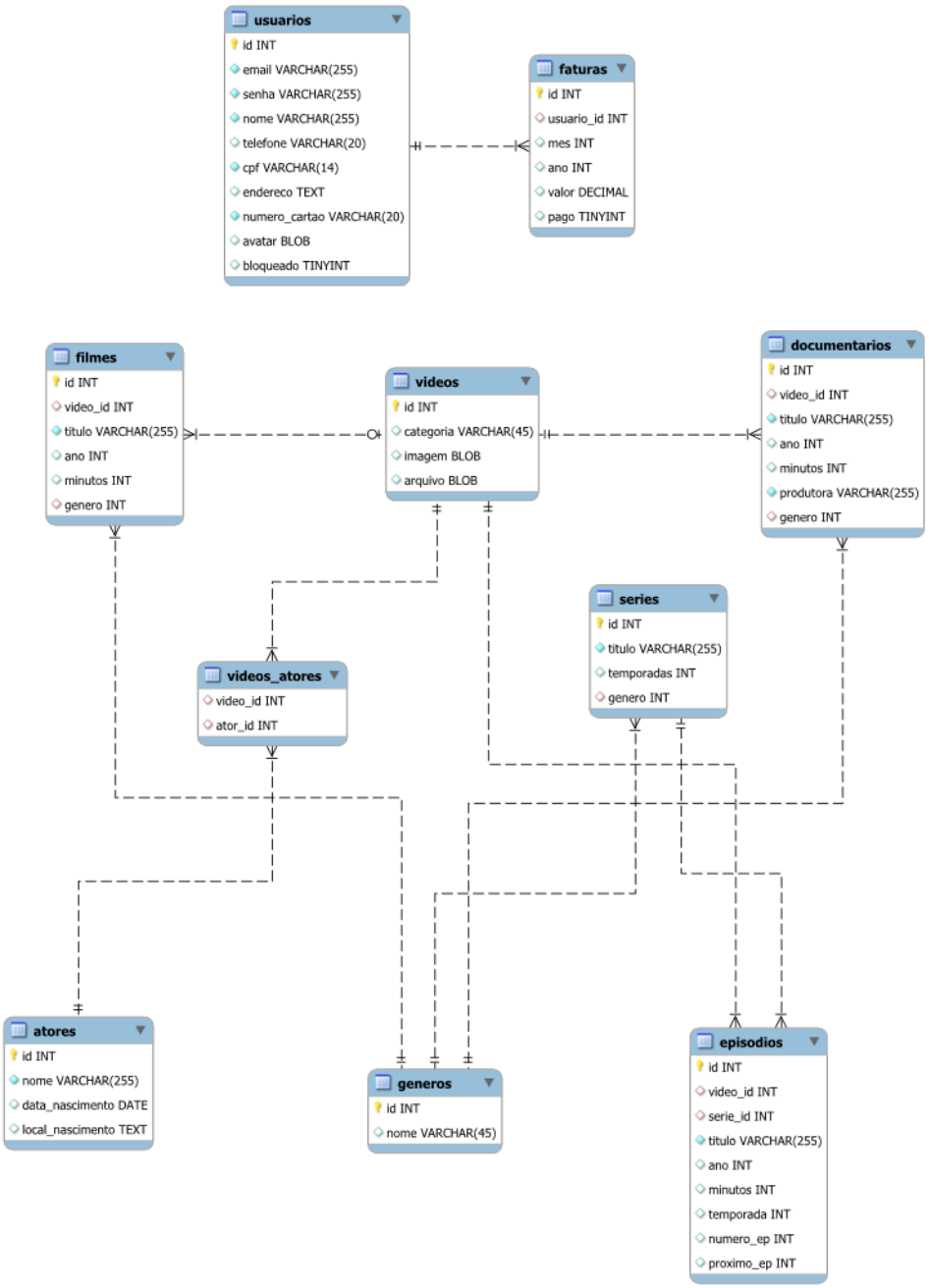
Índice Analítico

1. Modelo Entidade Relacionamento





2. Modelo Relacional



3. Código SQL de criação do banco de dados

```
-- CREATE SCRIPT


-- Create db

DROP DATABASE IF EXISTS streamberrydb;

CREATE DATABASE streamberrydb;

-- Create tables

CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
    senha VARCHAR(255) NOT NULL,
    nome VARCHAR(255) NOT NULL,
    telefone VARCHAR(20),
    cpf VARCHAR(14) NOT NULL UNIQUE,
    endereco TEXT,
    numero_cartao VARCHAR(20) NOT NULL,
    avatar OID,
    bloqueado BOOLEAN DEFAULT FALSE
);

CREATE TABLE faturas (
    id SERIAL PRIMARY KEY,
    usuario_id INT,
    mes INT,
    ano INT,
    valor DECIMAL(10, 2),
    pago BOOLEAN,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);

CREATE TABLE videos (
    id SERIAL PRIMARY KEY,
    categoria VARCHAR(45),
    imagem OID,
    arquivo OID
);

CREATE TABLE generos (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(45) NOT NULL UNIQUE
);

CREATE TABLE filmes (
    id SERIAL PRIMARY KEY,
    video_id INT,
    titulo VARCHAR(255) NOT NULL,
    ano INT,
```

```
    minutos INT,
    genero INT,
    FOREIGN KEY (video_id) REFERENCES videos(id) ON DELETE CASCADE,
    FOREIGN KEY (genero) REFERENCES generos(id)
);

CREATE TABLE series (
    id SERIAL PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    temporadas INT,
    genero INT,
    FOREIGN KEY (genero) REFERENCES generos(id)
);

CREATE TABLE episodios (
    id SERIAL PRIMARY KEY,
    video_id INT,
    serie_id INT,
    titulo VARCHAR(255) NOT NULL,
    ano INT,
    minutos INT,
    temporada INT,
    numero_ep INT,
    proximo_ep INT,
    FOREIGN KEY (video_id) REFERENCES videos(id) ON DELETE CASCADE,
    FOREIGN KEY (serie_id) REFERENCES series(id) ON DELETE CASCADE
);

CREATE TABLE documentarios (
    id SERIAL PRIMARY KEY,
    video_id INT,
    titulo VARCHAR(255) NOT NULL,
    ano INT,
    minutos INT,
    produtora VARCHAR(255) NOT NULL,
    genero INT,
    FOREIGN KEY (video_id) REFERENCES videos(id) ON DELETE CASCADE,
    FOREIGN KEY (genero) REFERENCES generos(id)
);

CREATE TABLE atores (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    data_nascimento DATE,
    local_nascimento TEXT
);

CREATE TABLE videos_atores (
    video_id INT,
    ator_id INT,
    FOREIGN KEY (video_id) REFERENCES videos (id) ON DELETE CASCADE,
    FOREIGN KEY (ator_id) REFERENCES atores (id) ON DELETE CASCADE
);
```

```
CREATE TABLE avaliacoes (
    usuario_id INT,
    video_id INT,
    nota INT,
    comentario TEXT,
    FOREIGN KEY (usuario_id) REFERENCES usuarios (id) ON DELETE CASCADE,
    FOREIGN KEY (video_id) REFERENCES videos (id) ON DELETE CASCADE
);

-- Create log tables

CREATE TABLE log_usuarios (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    cpf_usuario VARCHAR(14),
    operacao VARCHAR(20)
);

CREATE TABLE log_faturas (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_videos (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_generos (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_filmes (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_series (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);
```

```
CREATE TABLE log_episodios (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_documentarios (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_atores (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);

CREATE TABLE log_videos_atores (
    id SERIAL PRIMARY KEY,
    data_hora TIMESTAMP,
    autor VARCHAR(255),
    operacao VARCHAR(20)
);
```

4. Código SQL de inserção de registros no banco de dados

```
-- INSERT SCRIPT


INSERT INTO usuarios (email, senha, nome, telefone, cpf, endereco,
numero_cartao, avatar)

VALUES

    ('reinaldo@example.com', 'senha123', 'Reinaldo Gonçalves',
'123456789', '123.456.789-01', 'Endereço 1', '1234567890123456',
lo_import('C:\exemplos\avatar_exemplo.png')),

    ('adriana@example.com', 'senha456', 'Adriana da Silva', '987654321',
'987.654.321-01', 'Endereço 2', '6543210987654321',
lo_import('C:\exemplos\avatar_exemplo.png')),

    ('juliana@example.com', 'senha789', 'Juliana Gonçalves', '555555555',
'555.555.555-01', 'Endereço 3', '1111111111111111',
lo_import('C:\exemplos\avatar_exemplo.png'));


INSERT INTO faturas (usuario_id, mes, ano, valor, pago)

VALUES

    (1, 6, 2023, 50.00, true),

    (2, 6, 2023, 50.00, false),

    (3, 6, 2023, 50.00, false);


INSERT INTO videos (categoria, imagem, arquivo)

VALUES

    ('filmes', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('filmes', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('filmes', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('series', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('series', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('series', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('documentarios', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),

    ('documentarios', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv')),
```

```
    ('documentarios', lo_import('C:\exemplos\header_exemplo.png'),
lo_import('C:\exemplos\video_exemplo.mkv'));


INSERT INTO generos (nome)
VALUES
    ('Drama'),
    ('Fantasia'),
    ('Ficção Científica');


INSERT INTO filmes (video_id, titulo, ano, minutos, genero)
VALUES
    (1, 'O Poderoso Chefão', 1972, 175, 1),
    (2, 'Central do Brasil', 1998, 110, 1),
    (3, 'A Origem', 2010, 148, 3);


INSERT INTO series (titulo, temporadas, genero)
VALUES
    ('Game of Thrones', 8, 2),
    ('The Walking Dead', 11, 3),
    ('Black Mirror', 6, 3);


INSERT INTO episodios (video_id, serie_id, titulo, ano, minutos,
temporada, numero_ep, proximo_ep)
VALUES
    (4, 1, 'Winter Is Coming', 2011, 62, 1, 1, 2),
    (5, 2, 'What Comes After', 2018, 45, 9, 5, 6),
    (6, 3, 'The Entire History of You', 2011, 49, 1, 3, null);


INSERT INTO documentarios (video_id, titulo, ano, minutos, produtora,
genero)
VALUES
    (7, 'Documentário 1', 2022, 60, 'Produtora 1', 1),
    (8, 'Documentário 2', 2023, 75, 'Produtora 2', 2),
    (9, 'Documentário 3', 2021, 50, 'Produtora 3', 3);


INSERT INTO atores (nome, data_nascimento, local_nascimento)
VALUES
```

```
    ('Al Pacino', '1990-01-01', 'Local 1'),
    ('Fernanda Montenegro', '1985-05-10', 'Brasil'),
    ('Leonardo di Caprio', '1995-12-25', 'Local 3'),
    ('Kit Harrington', '1990-01-01', 'Local 1'),
    ('Andrew Lincoln', '1985-05-10', 'Local 2'),
    ('Toby Kebbell', '1995-12-25', 'Local 3'),
    ('Ator 1', '1990-01-01', 'Local 1'),
    ('Ator 2', '1985-05-10', 'Local 2'),
    ('Ator 3', '1995-12-25', 'Local 3');

INSERT INTO videos_atores (video_id, ator_id)
VALUES
    (1, 1),
    (2, 2),
    (3, 3),
    (4, 4),
    (5, 5),
    (6, 6),
    (7, 7),
    (8, 8),
    (9, 9);

INSERT INTO avaliacoes (usuario_id, video_id, nota, comentario)
VALUES
    (1, 1, 4, 'Bom filme!'),
    (2, 2, 3, 'Engraçado!'),
    (3, 3, 5, 'Excelente filme!'),
    (1, 4, 8, 'Como tudo começou!'),
    (1, 5, 10, 'Triste, mas é muito bom!'),
    (1, 6, 10, 'Melhor episódio da série!');
```

5. Código SQL de criação das *rules* do banco de dados

```
-- Delete/update rules

CREATE OR REPLACE RULE rl_delete_usuarios AS ON DELETE
    TO usuarios
    DO INSERT INTO log_usuarios (data_hora, autor, cpf_usuario, operacao)
    VALUES (current_timestamp, current_user, old.cpf, 'DELETE');

CREATE OR REPLACE RULE rl_update_usuarios AS ON UPDATE
    TO usuarios
    DO INSERT INTO log_usuarios (data_hora, autor, cpf_usuario, operacao)
    VALUES (current_timestamp, current_user, old.cpf, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_faturas AS ON DELETE
    TO faturas
    DO INSERT INTO log_faturas (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_faturas AS ON UPDATE
    TO faturas
    DO INSERT INTO log_faturas (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_videos AS ON DELETE
    TO videos
    DO INSERT INTO log_videos (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_videos AS ON UPDATE
    TO videos
    DO INSERT INTO log_videos (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_generos AS ON DELETE
```

```
    TO generos
    DO INSERT INTO log_generos (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_generos AS ON UPDATE
    TO generos
    DO INSERT INTO log_generos (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_filmes AS ON DELETE
    TO filmes
    DO INSERT INTO log_filmes (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_filmes AS ON UPDATE
    TO filmes
    DO INSERT INTO log_filmes (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_series AS ON DELETE
    TO series
    DO INSERT INTO log_series (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_series AS ON UPDATE
    TO series
    DO INSERT INTO log_series (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_episodios AS ON DELETE
    TO episodios
    DO INSERT INTO log_episodios (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_episodios AS ON UPDATE
    TO episodios
```

```
    DO INSERT INTO log_episodios (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_documentarios AS ON DELETE
    TO documentarios
    DO INSERT INTO log_documentarios (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_documentarios AS ON UPDATE
    TO documentarios
    DO INSERT INTO log_documentarios (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_atores AS ON DELETE
    TO atores
    DO INSERT INTO log_atores (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_atores AS ON UPDATE
    TO atores
    DO INSERT INTO log_atores (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

CREATE OR REPLACE RULE rl_delete_videos_atores AS ON DELETE
    TO videos_atores
    DO INSERT INTO log_videos_atores (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'DELETE');

CREATE OR REPLACE RULE rl_update_videos_atores AS ON UPDATE
    TO videos_atores
    DO INSERT INTO log_videos_atores (data_hora, autor, operacao)
    VALUES (current_timestamp, current_user, 'UPDATE');

-- User blob delete rule

CREATE OR REPLACE RULE rl_delete_blob_usuarios AS ON DELETE
```

```
    TO usuarios
    DO SELECT lo_unlink(OLD.avatar);

-- Video blobs delete rule

CREATE OR REPLACE RULE rl_delete_blob_videos AS ON DELETE
    TO videos
    DO SELECT lo_unlink(OLD.imagem), lo_unlink(OLD.arquivo);
```

6. Código SQL de criação das *stored procedures (functions)* do banco de dados

```
-- Function 1: Verificar mensalidades pendentes

CREATE OR REPLACE FUNCTION verificar_pendencias()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT COUNT(*) FROM faturas WHERE usuario_id = NEW.usuario_id AND pago
= false) >= 2 THEN
        UPDATE usuarios SET bloqueado = true WHERE id = NEW.usuario_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger (executar sempre que uma nova fatura for criada para o usuário)

CREATE TRIGGER verificar_pendencias_trigger
AFTER INSERT ON faturas
FOR EACH ROW
EXECUTE FUNCTION verificar_pendencias();

-- Function 2: Avaliar video

CREATE OR REPLACE FUNCTION avaliar_video(usuario_id INT, video_id INT, nota
INT, comentario TEXT)
RETURNS VOID AS $$
BEGIN
    IF nota >= 0 AND nota <= 10 THEN
        INSERT INTO avaliacoes (usuario_id, video_id, nota, comentario)
        VALUES (usuario_id, video_id, nota, comentario);
    ELSE
        RAISE EXCEPTION 'Nota inválida! A nota deve ser um valor de 0 a 10.';
    END IF;
END;
$$ LANGUAGE plpgsql;

-- Function 3: Retornar todos os vídeos disponíveis a partir do nome de um ator
ou atriz

CREATE OR REPLACE FUNCTION buscar_videos_por_ator(ator_nome VARCHAR(255))
RETURNS TABLE (
    video_id INT,
    categoria VARCHAR(20),
    titulo VARCHAR(255),
    serie VARCHAR(255)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        v.id AS video_id,
        v.categoria,
```

```
        CASE
            WHEN v.categoria = 'filmes' THEN f.titulo
            WHEN v.categoria = 'series' THEN e.titulo
            WHEN v.categoria = 'documentarios' THEN d.titulo
            ELSE NULL
        END AS titulo,
        CASE
            WHEN v.categoria = 'filmes' THEN NULL
            WHEN v.categoria = 'series' THEN s.titulo
            WHEN v.categoria = 'documentarios' THEN NULL
            ELSE NULL
        END AS serie
    FROM
        videos AS v
        LEFT JOIN filmes AS f ON v.id = f.video_id
        LEFT JOIN documentarios AS d ON v.id = d.video_id
        LEFT JOIN episodios AS e ON v.id = e.video_id
        LEFT JOIN series AS s ON e.serie_id = s.id
        LEFT JOIN videos_atores AS va ON v.id = va.video_id
        LEFT JOIN atores AS a ON va.ator_id = a.id
    WHERE
        a.nome ILIKE '%' || ator_nome || '%';
END;
$$ LANGUAGE plpgsql;

-- Function 4: Retornar todos os vídeos disponíveis a partir do nome de um ator
ou atriz

CREATE OR REPLACE FUNCTION buscar_videos_por_titulo(titulo_video VARCHAR(255))
RETURNS TABLE (
    video_id INT,
    categoria VARCHAR(20),
    titulo VARCHAR(255),
    serie VARCHAR(255)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        v.id AS video_id,
        v.categoria,
        CASE
            WHEN v.categoria = 'filmes' THEN f.titulo
            WHEN v.categoria = 'series' THEN e.titulo
            WHEN v.categoria = 'documentarios' THEN d.titulo
            ELSE NULL
        END AS titulo,
        CASE
            WHEN v.categoria = 'filmes' THEN NULL
            WHEN v.categoria = 'series' THEN s.titulo
            WHEN v.categoria = 'documentarios' THEN NULL
            ELSE NULL
        END AS serie
    FROM
        videos AS v
```

```
        LEFT JOIN filmes AS f ON v.id = f.video_id
        LEFT JOIN episodios AS e ON v.id = e.video_id
        LEFT JOIN series AS s ON e.serie_id = s.id
        LEFT JOIN documentarios AS d ON v.id = d.video_id
    WHERE
        CASE
            WHEN v.categoria = 'filmes' THEN f.titulo ILIKE titulo_video || '%'
            WHEN v.categoria = 'series' THEN e.titulo ILIKE titulo_video || '%'
            WHEN v.categoria = 'documentarios' THEN d.titulo ILIKE titulo_video
|| '%'
            ELSE NULL
        END;
END;
$$ LANGUAGE plpgsql;

-- Function 5: Retornar todos os vídeos disponíveis de acordo com o tipo
passado como parâmetro (filme, série ou documentário).

CREATE OR REPLACE FUNCTION buscar_videos_por_categoria(cat_video VARCHAR(255))
RETURNS TABLE (
    video_id INT,
    categoria VARCHAR(20),
    titulo VARCHAR(255),
    serie VARCHAR(255)
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        v.id AS video_id,
        v.categoria,
        CASE
            WHEN v.categoria = 'filmes' THEN f.titulo
            WHEN v.categoria = 'series' THEN e.titulo
            WHEN v.categoria = 'documentarios' THEN d.titulo
            ELSE NULL
        END AS titulo,
        CASE
            WHEN v.categoria = 'filmes' THEN NULL
            WHEN v.categoria = 'series' THEN s.titulo
            WHEN v.categoria = 'documentarios' THEN NULL
            ELSE NULL
        END AS serie
    FROM
        videos AS v
        LEFT JOIN filmes AS f ON v.id = f.video_id
        LEFT JOIN episodios AS e ON v.id = e.video_id
        LEFT JOIN series AS s ON e.serie_id = s.id
        LEFT JOIN documentarios AS d ON v.id = d.video_id
    WHERE
        v.categoria ILIKE '%' || cat_video || '%';
END;
$$ LANGUAGE plpgsql;
```