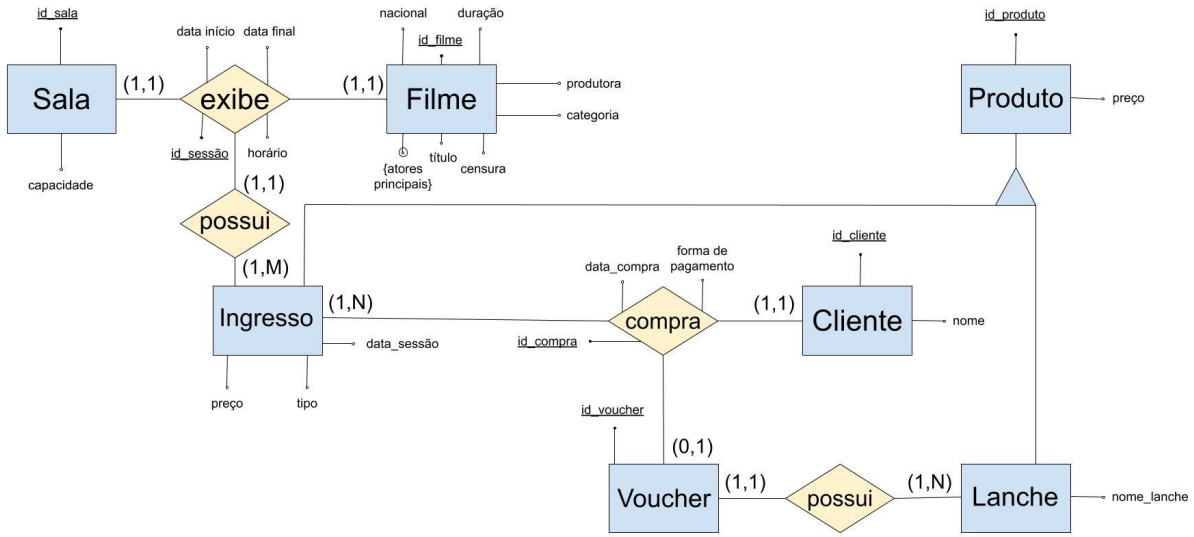
Universidade Federal da Paraíba
Centro de Informática
Banco de Dados
Professor Marcelo Iury

Documentação do banco de dados do Cinema Sauro

Lara di Cavalcanti Pontes (20190031550)
Humberto Navarro de Carvalho (20190029131)

João Pessoa
6 de dezembro de 2022

Documentação Modelo ER



Disponível também em: Modelo ER

Filme:

Em razão do atributo atores principais não seguir a primeira regra normal, por ser um atributo multivalorado, ele foi decomposto em tabelas na conversão para o modelo relacional. Assim, a tabela filme não possui o atributo de atores, e foi criada a tabela ator, com uma chave única e a informação do nome do ator.

Também foi criada a tabela filme_ator, que relaciona as duas, armazenando um id de filme e um id de ator em cada linha. Essa tabela obedece às 3 formas normais, porque a chave é única, composta por id_filme e id_ator.

A criação de uma tabela separada para ator foi necessária para seguir a segunda forma normal, já que se a coluna ator estivesse junto de filme_ator, existiriam casos com chaves diferentes e mesmos atributos.

A tabela produtora não era obrigatória para a normalização, mas foi criada para facilitar a manutenção. Quando for necessário acrescentar novas informações de uma produtora, por exemplo, é necessário atualizar apenas os valores de sua tabela. Também foi criada a tabela filme_produtora para relacionar as duas.

Sala e Cliente:

O modelo ER dessas entidades foi convertido diretamente para as tabelas de Sala e Cliente no modelo relacional.

Programação:

A tabela programação está ligada ao relacionamento exibe, por isso, possui os atributos próprios desse relacionamento e os ids da sala e do filme para a relação poder ser feita.

Produto, Ingresso e Lanche:

Como ingresso e lanche possuem o atributo preço em comum, e funcionam de forma semelhante, gerando a receita do cinema, ambos foram colocados na tabela produto, sendo diferenciados pelo atributo cod_produto. Isso também facilita um possível cadastro futuro de um novo produto a ser vendido, permitindo que o cálculo da receita do cinema continue sendo feito da mesma maneira.

Foi feita uma herança dessa tabela para lanche e para ingresso, cada uma com seus atributos próprios, sendo cada linha dessas tabelas um produto diferente que está sendo vendido, ou seja, na tabela lanche, estão apenas as informações de cada produto à venda. O cadastro de vendas de unidades de lanches será discutido na tabela voucher. Como cada ingresso vendido é diferente, o cadastro de vendas individuais desse produto é feito na própria tabela.

Compra e Voucher:

O relacionamento de compra é ternário entre ingresso, cliente e voucher no modelo ER. No modelo relacional, por sua vez, o cliente é representado por seu identificador foreign_key id_cliente como um atributo da compra, que é responsável pelo relacionamento entre as tabelas Compra e Cliente.

Como uma compra pode ter múltiplos ingressos e um voucher com múltiplos lanches, para normalizar as tabelas e calcular os preços dos produtos mais facilmente, foi criada a tabela compra_produto, aplicando a mesma estratégia feita em filme_ator, para relacionar uma compra com todos os ingressos e lanches referentes a ela.

Na tabela voucher, são inseridas as informações associadas à compra do lanche. Cada voucher é identificado com um id_voucher, e cada linha da tabela contém um produto vendido registrado sob esse voucher, que por sua vez, também é identificado pelo id_produto. A chave primária composta da tabela é feita pela junção desses dois ids. Como não seria possível cadastrar várias linhas iguais, em caso de um mesmo lanche comprado mais de uma vez em uma compra, a coluna quantidade foi criada no modelo relacional para identificar o número de unidades vendidas deste produto. Isso é expresso no modelo ER pela relação (1,N) entre voucher e lanche.

Modelo relacional

- Filme(id_filme, título, categoria, duração, censura, nacional)
- Filme_Produtora(id_filme, id_produtora) *id_filme referencia Filme:id_filme; id_produtora referencia Produtora:id_produtora*
- Produtora(id_produtora, nome_produtora)
- Filme_Ator(id_filme, id_ator) *id_filme referencia Filme:id_filme; id_ator referencia Ator:id_ator*
- Ator(id_ator, nome_ator)

- Sala(id_sala, capacidade)

- Programação(id_sessão, id_sala, id_filme, horário, data_início, data_final) *id_sala referencia Sala:id_sala; id_filme referencia Filme:id_filme*

- Produto(id_produto, cod_produto, preço)
- Ingresso(id_produto, id_sessão, data_sessão, tipo_ingresso) *id_produto referencia Produto:id_produto; id_sessão referencia Programação:id_sessão*
- Lanche(id_produto, nome_lanche) *id_produto referencia Produto:id_produto*

- Compra(id_compra, id_cliente, forma_de_pagamento, data_compra) *id_cliente referencia Cliente:id_cliente*
- Compra_Produto(id_compra, id_produto) *id_compra referencia Compra:id_compra; id_produto referencia Produto:id_produto*
- Compra_Voucher(id_compra, id_voucher) *id_compra referencia Compra:id_compra; id_voucher referencia Voucher:id_voucher*

- Cliente(id_cliente, nome_cliente)

- Voucher(id_voucher, id_produto, quantidade) *id_produto referencia Lanche:id_produto*

Documentação DDL, DML e DQL

Filme, Filme_Produtora, Filme_Ator

Criação Filme:

```
CREATE TABLE if not exists filme (
    id_filme integer PRIMARY KEY autoincrement NOT NULL,
    titulo varchar(90) NOT NULL,
    categoria varchar(90) NOT NULL,
    duracao integer NOT NULL,
    censura char NOT NULL,
    nacional BOOLEAN
)
```

Criação Filme_Produtora:

```
CREATE TABLE if not exists filme_produtora (
    id_filme integer,
    id_produtora integer,
    FOREIGN KEY (id_filme) REFERENCES filme (id_filme),
    FOREIGN KEY (id_produtora) REFERENCES produtora (id_produtora)
)
```

Criação Filme_Ator:

```
CREATE TABLE if not exists filme_ator (
    id_filme integer,
    id_ator integer,
    FOREIGN KEY (id_filme) REFERENCES filme (id_filme),
    FOREIGN KEY (id_ator) REFERENCES ator (id_ator)
)
```

Inserção Filme:

```
INSERT INTO filme (titulo, categoria, duracao, censura, nacional) VALUES('{title}',
'{genre}', {duration}, '{rating}', {national})
```

Inserção Filme_Produtora:

```
INSERT INTO filme_produtora(id_filme, id_produtora) VALUES({id_movie},
{id_producer})
```

Inserção Filme_Ator:

```
INSERT INTO filme_ator(id_filme, id_ator) VALUES({id_movie}, {id_actor})
```

Seleção dos atributos do filme de id {id_movie} e da produtora correspondente:

SELECT titulo, categoria, duracao, censura, nacional, nome_produtora FROM filme, filme_produtora, produtora WHERE filme.id_filme == filme_produtora.id_filme AND produtora.id_produtora == filme_produtora.id_produtora AND filme.id_filme == {id_movie}

Seleção dos atores do filme de id {id_movie}:
SELECT nome_ator FROM filme_ator, ator WHERE ator.id_ator == filme_ator.id_ator AND filme_ator.id_filme == {id_movie}

Seleção dos atributos de todos os filmes e de suas produtoras correspondentes:
SELECT filme.id_filme, titulo, categoria, duracao, censura, nacional, nome_produtora FROM filme, filme_produtora, produtora WHERE filme.id_filme == filme_produtora.id_filme AND produtora.id_produtora == filme_produtora.id_produtora

Seleção dos atores de cada filme:
SELECT nome_ator FROM filme_ator, ator WHERE ator.id_ator == filme_ator.id_ator

Produtora

Criação:
```
CREATE TABLE if not exists produtora (
    id_produtora integer PRIMARY KEY autoincrement NOT NULL,
    nome_produtora varchar(90) NOT NULL
)
```

Inserção:
INSERT INTO produtora(nome_produtora) VALUES('{name}')

Seleção do id da produtora de nome {name}:
SELECT id_produtora FROM produtora WHERE nome_produtora == '{name}'

Ator

Criação:
```
CREATE TABLE if not exists ator (
    id_ator integer PRIMARY KEY autoincrement NOT NULL,
    nome_ator varchar(90) NOT NULL
```

)

Inserção:
INSERT INTO ator(nome_ator) VALUES('{name}')

Seleção do id do ator de nome {name}:
SELECT id_ator FROM ator WHERE nome_ator == '{name}'

Sala

Criação:
```
CREATE TABLE if not exists sala (
    id_sala integer PRIMARY KEY autoincrement NOT NULL,
    capacidade integer NOT NULL
)
```

Inserção:
INSERT INTO sala(capacidade) VALUES({capacity})

Seleção da capacidade da sala de id {id_screen}:
SELECT capacidade FROM sala WHERE id_sala == {id_screen}

Seleção de todos os atributos de todas as salas:
SELECT * FROM sala

Seleção da capacidade da sala onde a sessão de id {id_session} será exibida:
SELECT capacidade FROM sala, programacao WHERE sala.id_sala == programacao.id_sala AND id_sessao == {id_session}

Programação

Criação:
```
CREATE TABLE if not exists programacao (
    id_sessao integer PRIMARY KEY autoincrement NOT NULL,
    id_filme integer NOT NULL,
    id_sala integer NOT NULL,
    horario time NOT NULL,
    data_inicio date NOT NULL,
    data_fim date NOT NULL,
    FOREIGN KEY (id_filme) REFERENCES filme (id_filme),
```

```
    FOREIGN KEY (id_sala) REFERENCES sala (id_sala)
)
```

Inserção:
```
INSERT INTO programacao (id_filme, id_sala, horario, data_inicio, data_fim)
VALUES({id_movie}, {id_screen}, '{time}', '{date_beginning}', '{date_end}')
```

Seleção dos atributos da sessão de id {id_session}:
```
SELECT id_filme, id_sala, horario, data_inicio, data_fim FROM programacao
WHERE id_sessao == {id_session}
```

Seleção dos atributos das sessões disponíveis na data {date}:
```
SELECT id_filme, id_sala, horario, data_inicio, data_fim, id_sessao FROM
programacao WHERE '{date}' between data_inicio AND data_fim
```

Seleção dos atributos de todas as sessões:
```
SELECT * FROM programacao
```

Produto, Ingresso, Lanche

Criação Produto:
```
CREATE TABLE if not exists produto (
    id_produto integer PRIMARY KEY autoincrement NOT NULL,
    cod_produto integer NOT NULL,
    preco numeric(7,2) NOT NULL
)
```

Criação Ingresso:
```
CREATE TABLE if not exists ingresso (
    id_produto integer,
    id_sessao integer,
    data_sessao date NOT NULL,
    tipo_ingresso varchar(15) NOT NULL,
    FOREIGN KEY (id_produto) REFERENCES produto (id_produto),
    FOREIGN KEY (id_sessao) REFERENCES programacao (id_sessao)
)
```

Criação Lanche:
```
CREATE TABLE if not exists lanche (
    id_produto integer,
    nome_lanche varchar(90) NOT NULL,
    FOREIGN KEY (id_produto) REFERENCES produto (id_produto)
```

)

Inserção Produto ({cod} = 1 para ingresso, {cod} = 2 para lanche):
INSERT INTO produto(cod_produto, preco) VALUES({cod}, {price})

Inserção Ingresso:
INSERT INTO ingresso(id_produto, id_sessao, data_sessao, tipo_ingresso) VALUES({id_product}, {id_session}, '{date}', '{type}')

Inserção Lanche:
INSERT INTO lanche(id_produto, nome_lanche) VALUES({id_product}, '{name}')

Seleção do preço do produto de id {id}:
SELECT preco FROM produto WHERE id_produto == {id}

Contagem de quantos ingressos foram vendidos para a sessão de id {id_session} na data {date}:
SELECT COUNT(*) FROM ingresso WHERE id_sessao == {id_session} AND data_sessao == {date}

Seleção dos atributos do ingresso de id {id}:
SELECT id_sessao, data_sessao, tipo_ingresso, preco FROM produto, ingresso WHERE produto.id_produto == ingresso.id_produto AND produto.id_produto == {id}

Seleção dos atributos de todos os ingressos:
SELECT produto.id_produto, id_sessao, data_sessao, tipo_ingresso, preco FROM produto, ingresso WHERE produto.id_produto == ingresso.id_produto

Seleção dos atributos do lanche de nome {name}:
SELECT produto.id_produto, nome_lanche, preco FROM produto, lanche WHERE produto.id_produto == lanche.id_produto AND nome_lanche == '{name}'

Seleção dos atributos de todos os lanches:
SELECT produto.id_produto, nome_lanche, preco FROM produto, lanche WHERE produto.id_produto == lanche.id_produto

Compra, Compra_Produto, Compra_Voucher

Criação Compra:
```
CREATE TABLE if not exists compra (
    id_compra integer PRIMARY KEY autoincrement NOT NULL,
    id_cliente integer,
```

```
    forma_de_pagamento varchar(45) NOT NULL,
    data_compra datetime NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente)
)
```

Criação Compra_Produto:

```
CREATE TABLE if not exists compra_produto (
    id_compra integer,
    id_produto integer,
    FOREIGN KEY (id_compra) REFERENCES compra (id_compra),
    FOREIGN KEY (id_produto) REFERENCES produto (id_produto)
)
```

Criação Compra_Voucher:

```
CREATE TABLE if not exists compra_voucher (
    id_compra integer,
    id_voucher integer,
    FOREIGN KEY (id_compra) REFERENCES compra (id_compra),
    FOREIGN KEY (id_voucher) REFERENCES voucher (id_voucher)
)
```

Inserção Compra:

```
INSERT INTO compra(id_cliente, forma_de_pagamento, data_compra)
VALUES({id_client}, '{form_of_payment}', '{date}')
```

Inserção Compra_Produto:

```
INSERT INTO compra_produto(id_compra, id_produto) VALUES({id_purchase},
{id_product})
```

Inserção Compra_Voucher:

```
INSERT INTO compra_voucher(id_compra, id_voucher) VALUES('{id_purchase}',
'{id_voucher}')
```

Seleção dos atributos da compra de id {id_purchase}:

```
SELECT id_cliente, forma_de_pagamento, data_compra FROM compra WHERE
id_compra == {id_purchase}
```

Seleção do preço dos ingressos comprados na compra de id {id_purchase}:

```
SELECT preco FROM compra_produto, produto WHERE
compra_produto.id_produto == produto.id_produto AND id_compra == {id_purchase}
AND cod_produto == 1
```

Seleção do id e da quantidade de cada produto presente no voucher relacionado à compra de id {id_purchase}:
SELECT id_produto, quantidade FROM compra_voucher, voucher WHERE compra_voucher.id_voucher == voucher.id_voucher AND id_compra == {id_purchase}

Cliente

Criação:
```
CREATE TABLE if not exists cliente (
    id_cliente integer PRIMARY KEY autoincrement NOT NULL,
    nome_cliente varchar(90) NOT NULL
)
```

Inserção:
INSERT INTO cliente (nome_cliente) VALUES('{name}')

Atualização do nome do cliente de {name} para {new_name}:
UPDATE cliente SET nome_cliente = '{new_name}' WHERE nome_cliente == '{name}'"

Atualização do nome do cliente de id {id} para {new_name}:
UPDATE cliente SET nome_cliente = '{new_name}' WHERE id_cliente == {id}

Seleção de todos os atributos do cliente de nome {name}:
SELECT * FROM cliente WHERE nome_cliente == '{name}'

Seleção do id do cliente de nome {name}:
SELECT id_cliente FROM cliente WHERE nome_cliente == '{name}'

Seleção de todos os atributos de todos os clientes:
SELECT * FROM cliente

Deleção do cliente de nome {name}:
DELETE FROM cliente WHERE nome_cliente == '{name}'

Deleção do cliente de id {id}:
DELETE FROM cliente WHERE id_cliente == {id}

Voucher

Criação

```
CREATE TABLE if not exists voucher (
    id_voucher integer NOT NULL,
    id_produto integer NOT NULL,
    quantidade integer NOT NULL,
    FOREIGN KEY (id_produto) REFERENCES lanche (id_produto),
    CONSTRAINT id_voucher_comp PRIMARY KEY (id_voucher, id_produto)
)
```

Inserção

```
INSERT INTO voucher(id_voucher, id_produto, quantidade) VALUES({id_voucher}, {id_product}, {quantity})
```

Seleção dos atributos do voucher de id {id_voucher}:

```
SELECT nome_lanche, quantidade, preco FROM produto, lanche, voucher WHERE produto.id_produto == lanche.id_produto AND lanche.id_produto == voucher.id_produto AND id_voucher == {id_voucher}
```

Seleção dos atributos de todos os vouchers:

```
SELECT id_voucher, nome_lanche, quantidade, preco FROM produto, lanche, voucher WHERE produto.id_produto == lanche.id_produto AND produto.id_produto == voucher.id_produto
```

Seleção dos atributos do último voucher gerado:

```
SELECT MAX(id_voucher), id_produto, quantidade FROM voucher
```