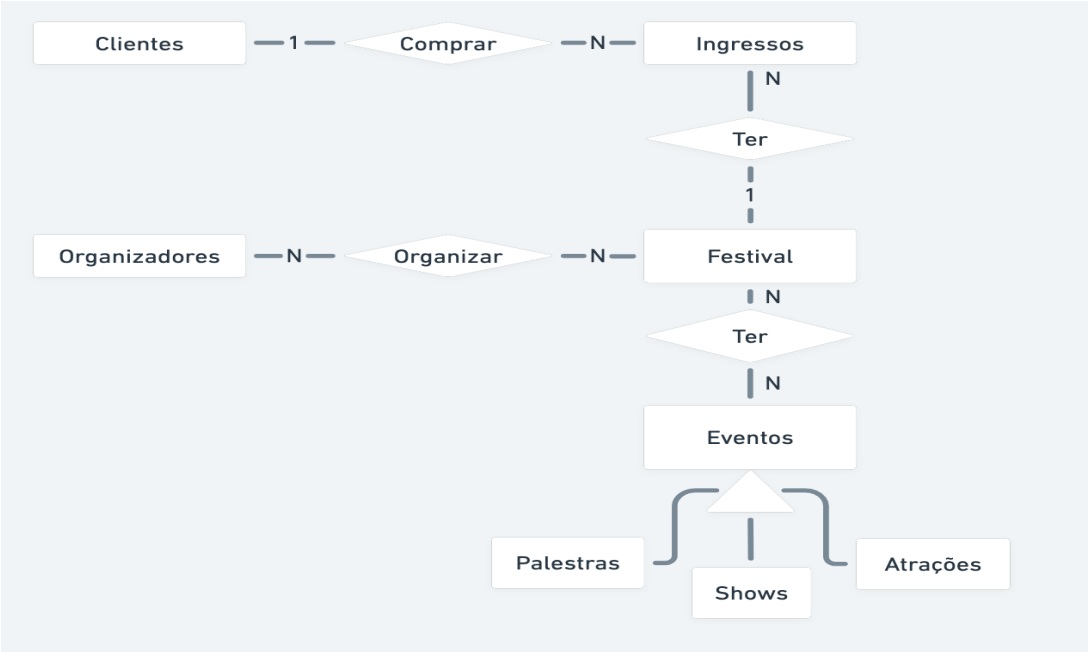
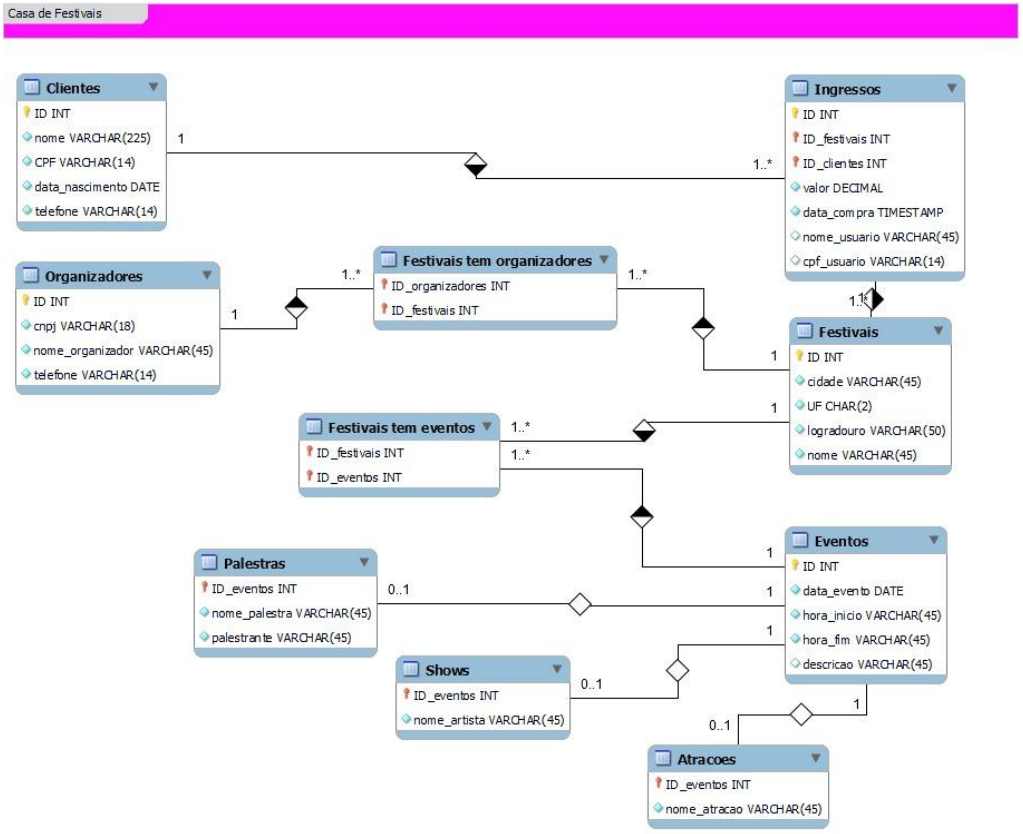
Disciplina de Banco de Dados

Trabalho AP1 BD - Angela Alves e Marcus Barcelos





A regra de negócio utilizada para esse trabalho consiste em uma plataforma de gerenciamento de ingressos de festivais. As cardinalidades nesta modelagem são:

- Um cliente pode comprar um ou muitos ingressos;
- Um ingresso é comprado por apenas um cliente, refere-se a apenas um festival e contém os dados do usuário daquele ingresso;
- Um festival tem um ou muitos eventos, e pode ter um ou muitos organizadores;
- Um organizador realiza um ou muitos festivais;
- Um evento pode estar em um ou muitos festivais;
- Um evento pode ser de três tipos: shows, palestras e atrações;

Há algumas restrições de integridade que podemos observar nessa modelagem. Entre elas, podemos citar:

- Integridade de chave: representada pela presença da chave primária (PK), o que assegura a unicidade das tuplas. No nosso banco de dados, pode ser exemplificado essa restrição através da coluna "id" presente em cada tabela.
- Integridade referencial: garantida pela presença da chave estrangeira (FK), que se refere a chave primária da tabela que está se mantendo um relacionamento. Como exemplo, na relação entre Clientes e Ingressos, que tem cardinalidade (1, N), a tabela Ingressos mantém referência do cliente que o comprou por meio de uma FK, o id_cliente - sendo que esse ID deve ser equivalente a PK denominada ID na tabela Clientes.
- Integridade de domínio: demonstrada em nossa modelagem pela tipificação dos dados, garante a consistência dos dados inseridos. Logo, por exemplo, o valor do ingresso adicionado na coluna da tabela Ingressos deverá ser sempre um "Decimal" e não será possível introduzir um dado do tipo "Varchar" no lugar.
- Unicidade com Unique: a propriedade Unique garante que aquele atributo seja único na tabela, mesmo não sendo uma chave primária. Utilizamos essa propriedade na nossa modelagem para a coluna "CPF" em Clientes para

garantir que não haverá duplicidade do valor desse campo em clientes diferentes.

Além disso, deixamos exposto as restrições de atualização e exclusão. A opção "cascade" assegura que, se um registro na tabela pai for atualizado, as alterações serão automaticamente refletidas nas tuplas correspondentes na tabela filha. Já a opção "restrict" garante que não ocorrerá a exclusão de registros relacionados - por exemplo, não permite deletar um festival que possui ingressos relacionados, isso se dá através da FK "id festival" presente na tabela Ingressos.

Views

```sql
-- Uma view que mostra informações sobre as atrações de um festival específico,
-- incluindo nome do festival, nome da atração e horário de início do evento.
create view atracao_dj
as
select  f.nome as nome_festival,
        a.nome_atracao as nome_atracao,
        e.hora_inicio as hora_inicio
from    festivais f
            inner join festivais_eventos fe
                on fe.id_festival = f.id
            inner join eventos e
                on e.id = fe.id_evento
            inner join atracoes a
                on a.id_evento = e.id
where   a.nome_atracao like '%DJ 1%'
```

```sql
-- Uma view que mostra informações sobre todos os eventos agendados no Rio de Janeiro,
-- incluindo nome do evento, data e hora do evento e o nome do festival
create view eventos_agendados_RJ
as
select  e.descricao as evento,
        f.nome as nome_festival,
        e.data_evento as data_evento,
        e.hora_inicio as hora
from    eventos e
            inner join festivais_eventos fe
                on fe.id_evento = e.id
            inner join festivais f
                on f.id = fe.id_festival
where   f.cidade like '%Rio de Janeiro%'
```

```sql
-- Uma view que mostra informações completas sobre um ingresso vendido,
-- incluindo nome do cliente, nome do festival, data do evento e valor do ingresso.
create view informacoes_completas
as
select  c.nome as nome_cliente,
        f.nome as nome_festival,
        e.data_evento as data_evento,
        i.valor as preco_pago
from    ingressos i
            inner join clientes c
                on c.id = i.id_cliente
            inner join festivais f
                on f.id = i.id_festival
            inner join festivais_eventos fe
                on fe.id_festival = f.id
            inner join eventos e
                on fe.id_evento = e.id
```

Stored Procedure

```
6
7   -- Listar todos os ingressos comprados por um cliente com base no seu CPF
8   DELIMITER $$
9   CREATE PROCEDURE ingressos_por_cpf(p_cpf_cliente VARCHAR(14))
0   BEGIN
1       SELECT i.id, i.valor, i.data_compra, f.nome, e.descricao
2       FROM ingressos i
3       INNER JOIN clientes c ON i.id_cliente = c.id
4       INNER JOIN festivais f ON i.id_festival = f.id
5       INNER JOIN eventos e ON f.id = e.id
6       WHERE c.cpf = p_cpf_cliente;
7   END $$
8
9   DELIMITER ;
0
1   call ingressos_por_cpf("123.456.789-00")
2
3   -- Deletar ingresso por id
4   DELIMITER $$
5   CREATE PROCEDURE excluir_ingresso(p_ingresso_id INT)
6   BEGIN
7     DELETE FROM ingressos WHERE id = p_ingresso_id;
8   END $$
9   DELIMITER ;
0
1   --- Deletar Cliente por id
2   DELIMITER $$
3   CREATE PROCEDURE excluir_cliente(p_cliente_id INT)
4   BEGIN
5     DELETE FROM clientes WHERE id = p_cliente_id;
6   END $$
7   DELIMITER ;
8
```

```sql
-- Inserir Cliente
DELIMITER $$
CREATE PROCEDURE inserir_cliente(p_nome varchar(70), p_telefone varchar(11), p_cpf varchar(14), p_data_nascimento date)
BEGIN
    INSERT INTO
        clientes (nome, telefone, cpf, data_nascimento)
    VALUES
        (p_nome, p_telefone, p_cpf, p_data_nascimento);
END $$
DELIMITER ;

call inserir_cliente('Angela Alves', '51981009791', '045.422.422-00', '2001-05-16')

--- Alterar cliente
DELIMITER $$
CREATE PROCEDURE alterar_cliente(p_id_cliente int, p_nome varchar(70), p_telefone varchar(11), p_cpf varchar(14), p_data_nascimento date)
BEGIN
    UPDATE clientes
        SET nome = p_nome,
            telefone = p_telefone,
            cpf = p_cpf
            data_nascimento = p_data_nascimento
        WHERE id = p_id_cliente
END $$
DELIMITER ;

call alterar_cliente(51, 'Angela', '12345678910', '000.422.422-00', '2001-06-16');
```

```sql
--- Receber o ID de um cliente como parâmetro e retornar todas as informações relacionadas a esse cliente,
--- incluindo seus ingressos comprados e os eventos associados a esses ingressos.

DELIMITER $$
CREATE  PROCEDURE buscar_dados_por_id(p_id int)
BEGIN
    SELECT  c.nome as nome_cliente,
            c.telefone as telefone,
            c.cpf as cpf_cliente,
            c.data_nascimento as data_nascimento,
            e.descricao as evento
    FROM    clientes c
                left join ingressos i
                    on c.id = i.id_cliente
                left join festivais f
                    on f.id = i.id_festival
                left join  festivais_eventos fe
                    on fe.id_festival = f.id
                left join eventos e
                    on fe.id_evento = e.id
    WHERE   c.id = p_id;
END $$
DELIMITER ;
```