Unidade curricular: Bases de Dados

Projeto- Ano letivo 2023/2024 | 1º Semestre

Relatório P-PII

Nome: Bruno Filipe Bordalo Pereira da Silva

Número: 110005

Nome: Diogo Pedro Cordeiro Pereira

Número: 110976

Licenciatura em Engenharia Informática, 2º ano

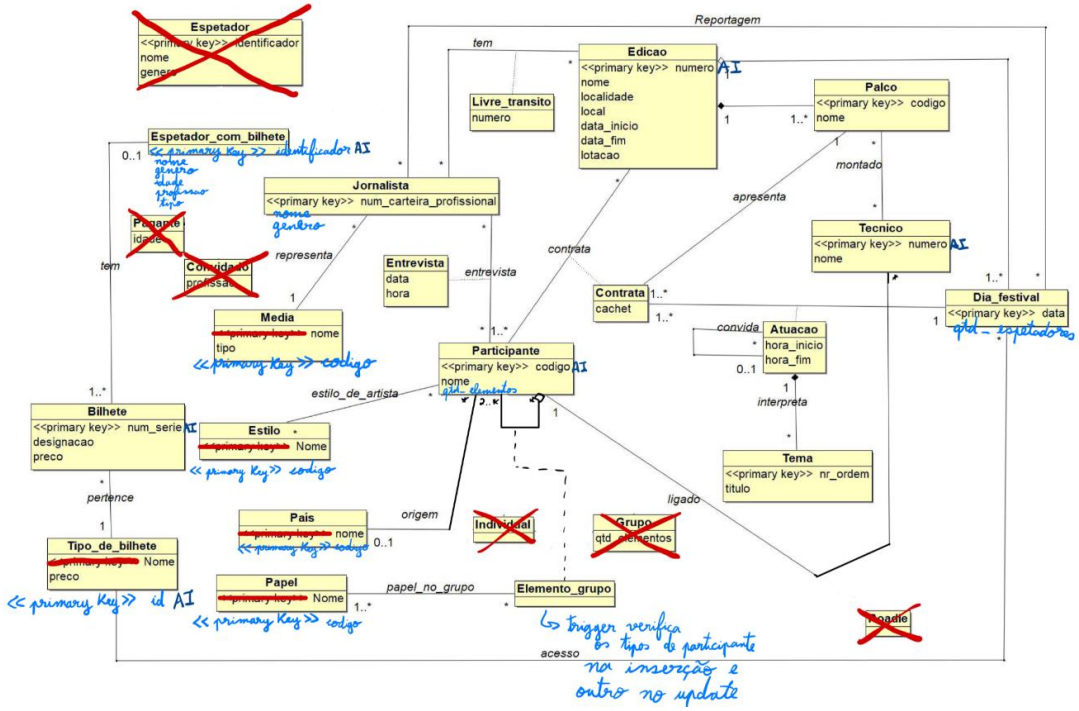
Data de entrega

18-12-2023

Índice

1 - Otimizações da BD

Esquema com as otimizações realizadas:



Alterações detalhadas e justificações:

Removeram se todas as generalizações e alteraram-se todos as primary keys com valores não numéricos.

Acabou-se com a tabela Espetador e passaram se os atributos para o Espetador_com_bilhete e o Jornalista, sendo que o Espetador_com_bilhete mantém a primary key Espetador_identificador e o jornalista fica unicamente com o num_carteira profissional como primary key.

```
create table Espetador_com_bilhete
(
   Espetador_identificador   Integer   not null AUTO_INCREMENT,
   nome   varchar(100)   null,
   genero   enum('M', 'F')   null,
   subtipo enum('Pagante', 'Convidado') not null,
   idade   tinyint   null,
   profissao   varchar(60)   null,

  constraint CK_Espetador_com_bilhete CHECK ((idade is not null and subtipo = 'Pagante' and profissao is null) or (profissao is not null and subtipo = 'Convidado' and idade is null)),
  constraint PK_Espetador_com_bilhete primary key (Espetador_identificador)

);

create table Jornalista
(
   nome   varchar(100)   null,
   genero   enum('M', 'F')   null,
   Media_codigo   char(2)   not null,
   num_carteira_profissional   Integer   not null,

   constraint PK_Jornalista primary key (num_carteira_profissional)
);
```

Para remover as generalizações do Pagante e do Convidado, adicionou-se os atributos de ambos ao Espetador_com_bilhete e adicionou-se uma coluna com dois enumerados, “Pagante” e “Convidado”, em que uma constrain, CK_Espetador_com_bilhete, adicionada na linha de código acima, delimita através desses tipos enumerados qual o campo que pode ser preenchido, tendo o campo que não pertence ao tipo selecionado de ficar a null.

Removeu-se a generalização do Roadie, passando a FK Participante_codigo para a tabela Tecnico e colocando uma constrain semelhante à da otimização anterior no código, recorrendo também aos tipos enumerados.

```
create table Tecnico
(
   numero   Integer   not null AUTO_INCREMENT,
   nome   varchar(120) not null,
   tipo enum ('Roadie','Organizacao')  not null,
   Participante_codigo   smallint   null,

   constraint CK_Tecnico CHECK ((Participante_codigo is not null and tipo = 'Roadie') or (Participante_codigo is null and tipo = 'Organizacao')),
   constraint PK_Tecnico primary key (numero)
);
```

Por último acabamos com a generalização do participante individual e grupo, recorrendo também aos tipos enumerados e a uma constrain. Esta generalização implicou a criação de um trigger, que atua antes da inserção e da atualização, na tabela elementos_grupo, para garantir que as primary keys desta última, Individual_Participante_codigo e Grupo_Participante_codigo, correspondessem respetivamente ao tipo de Participante pretendido de modo a não haver erros na base de dados

```
create table Participante
(
   codigo   smallint   not null AUTO_INCREMENT,
   nome   varchar(80)   null,
   tipo enum ('Individual', 'Grupo') not null,
   Pais_codigo   char(2)   null,
   qtd_elementos   tinyint   null,

   constraint CK_Participante CHECK ((Pais_codigo is not null and tipo = 'Individual' and qtd_elementos is null) or (qtd_elementos is not null and tipo = 'Grupo' and Pais_codigo is null)),
   constraint PK_Participante primary key (codigo)
);
```

```
1. BEGIN
2.   if (SELECT tipo FROM `participante` WHERE codigo =
          new.Individual_Participante_codigo) <> 'Individual' THEN
3.         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Participante não individual!';
4.   END IF;
5.   if (SELECT tipo FROM `participante` WHERE codigo =
          new.Grupo_Participante_codigo) <> 'Grupo' THEN
6.         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Participante não é um grupo!';
7.   END IF;
8. END
9.
```

Adicionaram-se primary keys mais eficientes às tabelas Media, Estilo, Pais, Tipo_de_Bilhete e Papel, pois estas tinham como PK o campo nome.

Por uma questão de eficiência, praticabilidade e minimização de erros, colocou-se Auto Increment às primary keys numéricas das tabelas Espetador_com_bilhete, Edicao, Tecnico, Participante, Bilhete e Tipo_de_bilhete. (identificados com 'AI' no esquema no inicio desta secção)

Removeram-se também todos os underscores ('_') do final dos nomes das variáveis pois achamos que tornaria o código mais confuso e dificultaria a criação de queries.

Introduziu-se também a coluna qtd_espetadores na tabela dia_festival, como pedido no enunciado do trigger T2.

Na tabela contrata, encontrou-se um bug, em que os atributos Convidado_Edicao_numero e Convidado_Participante_codigo estavam definidos em NOT NULL, o que impediria a inserção de uma atuação sem haver convidado registado, o que não faria sentido na base de dados e também impediria a inserção de qualquer dado nessa tabela, pois se a foreign key vem da mesma tabela e é obrigatória, nunca daria para inserir a primeira linha de dados.

Por questões de preservação de dados e coerência dos mesmos no trigger T2, alterámos foreign key que liga o Bilhete com o Espetador_com_bilhete para RESTRICT on DELETE de modo a que não seja possível apagar o bilhete.

Adicionámos um trigger extra na tabela dia_festival, que apenas permita inserir uma data que esteja entre a data_inicio e a data_fim da edição fornecida. (este trigger atua antes da inserção e atualização).

```
1. BEGIN
2.         IF(new.data not between (SELECT data_inicio FROM edicao WHERE numero =
new.Edicao_numero) AND ((SELECT data_fim FROM edicao WHERE numero = new.Edicao_numero)))
THEN
3.                   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Dia não pertence ao
festival!';
4.         END IF;
5. END
6.
```

Ponderaram-se a criação de diversos triggers extras para limitar os dados da BD e tornar estes mais coesos, em baixo uma listagem destes:

-Limitar o número de ligações de participantes individuais com grupos na tabela elemento_grupo, de maneira que não pudessem haver mais ligações ao grupo que o atributo qtd_elementos deste.

-Na tabela contrata existem vários triggers que poderiam ser criados tais como:

-um que garante a coesão de dados do valor Palco_Edicao_numero e Palco_codigo;

-o mesmo para o Convidado_Edicao_numero e Convidado_Participante_Codigo;

-um que garanta que o Dia_festival_data pertença à Edição correta;

-outro que verifica que a hora_inicio é antes da hora_fim.

-Na tabela edicao, um trigger que garanta que a data_inicio é anterior ou igual à data_fim.

Foram analisadas limitações da base de dados fornecida, tais como na tabela entrevista, as primary keys são o Participante_codigo e o Jornalista_num_carteira_profissional, isto impede que haja mais do que uma entrevista entre um jornalista e um participante, por exemplo noutra edição do festival. Sugeria-se, embora torne a BD menos eficiente, adicionar também a data com primary key.

2- Códigos dos Automatismos: Triggers, Stored Procedures e Funções:

2.1 Triggers

No código a parte "create procedure" é meramente indicativo dos argumentos a serem passados para dentro dos procedimentos, pois eles foram criados via interface gráfica do phpMyAdmin.

T1-

```
1. BEGIN
2.        IF (((SELECT Participante_codigo FROM `tecnico` WHERE numero =
new.Tecnico_numero) <> (SELECT Participante_codigo FROM `contrata` WHERE
Palco_Edicao_numero = new.Palco_Edicao_numero AND Palco_codigo = new.Palco_codigo)) AND
((SELECT tipo FROM `tecnico` WHERE numero = new.Tecnico_numero) <> 'Organizacao')) THEN
3.     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Participante não corresponde ao
Roadie';
4.     END IF;
5. END
6.
```

T2- insert after bilhete

```
1. BEGIN
2.        UPDATE dia_festival SET qtd_espetadores = (qtd_espetadores + 1) WHERE data IN
(SELECT Dia_festival_data FROM acesso WHERE Tipo_de_bilhete_id = new.Tipo_de_bilhete_id);
3. END
4.
```

T2- update after bilhete

```
 1. BEGIN
 2.       IF (old.devolvido=0 AND new.devolvido=1) THEN
 3.                UPDATE dia_festival SET qtd_espetadores = (qtd_espetadores - 1) WHERE
data IN (SELECT Dia_festival_data FROM acesso WHERE Tipo_de_bilhete_id =
                old.Tipo_de_bilhete_id);
 4.       ELSEIF (old.devolvido=1 AND new.devolvido=0) THEN
 5.                UPDATE dia_festival SET qtd_espetadores = (qtd_espetadores + 1) WHERE
data IN (SELECT Dia_festival_data FROM acesso WHERE Tipo_de_bilhete_id =
                                    new.Tipo_de_bilhete_id);
 6.       ELSEIF (old.Tipo_de_bilhete_id <> new.Tipo_de_bilhete_id AND new.devolvido<>1)
THEN
 7.                UPDATE dia_festival SET qtd_espetadores = (qtd_espetadores - 1) WHERE
data IN (SELECT Dia_festival_data FROM acesso WHERE Tipo_de_bilhete_id =
                                    old.Tipo_de_bilhete_id);
 8.                UPDATE dia_festival SET qtd_espetadores = (qtd_espetadores + 1) WHERE
data IN (SELECT Dia_festival_data FROM acesso WHERE Tipo_de_bilhete_id =
                                    new.Tipo_de_bilhete_id);
 9.       END IF;
10. END
11.
```

T2- insert e update before bilhete

```
1. BEGIN
2.         IF (new.devolvido<>1) THEN
3.                   IF (SELECT COUNT(*) FROM dia_festival WHERE ((data IN (SELECT
Dia_festival_data FROM acesso WHERE (Tipo_de_bilhete_id = new.Tipo_de_bilhete_id))) AND
4.         (qtd_espetadores=(SELECT lotacao FROM edicao WHERE(numero =
Edicao_numero)))))<>0 THEN
5.                             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Lotacao
Esgotada!';
6.                   END IF;
7.         END IF;
8. END
9.
```

2.2 Stored Procedures

P1-

```
 1. create procedure P1_ClonarEdicao(in EdicaoClonar integer, dataNova date)
 2. BEGIN
 3.       declare new_edicao_numero numeric;
 4.       declare new_data_fim date;
 5.       set new_data_fim = (SELECT DATE_ADD(dataNova, INTERVAL (SELECT DATEDIFF((SELECT
data_fim FROM edicao WHERE (numero = EdicaoClonar)),(SELECT data_inicio FROM edicao WHERE
(numero = EdicaoClonar)))) DAY));
 6.
 7.       INSERT INTO Edicao
 8.                 (`nome`, `localidade`, `local`, `data_inicio`, `data_fim`, `lotacao`)
 9.       SELECT
10.                 nome, localidade, local, dataNova, new_data_fim, lotacao
11.       FROM
12.                 edicao
13.       WHERE
14.                 numero = EdicaoClonar;
15.
16.
17.       set new_edicao_numero = (SELECT MAX(numero) FROM edicao);
18.
19.
20.
21.       INSERT INTO palco
22.                 (`Edicao_numero`, `codigo`, `nome`)
23.       SELECT
24.                 new_edicao_numero, codigo, nome
25.       FROM
26.                 palco
27.       WHERE
28.                 Edicao_numero = EdicaoClonar;
29.
30.
31.
32.       INSERT INTO dia_festival
33.                 (`Edicao_numero`, `data`)
34.       SELECT
35.                 new_edicao_numero, (SELECT DATE_ADD(dataNova, INTERVAL (SELECT
DATEDIFF( data,(SELECT data_inicio FROM edicao WHERE (numero = EdicaoClonar)))) DAY))
36.       FROM
37.                 dia_festival
38.       WHERE
39.                 Edicao_numero = EdicaoClonar;
40.
41. END
42.
```

P2-

```
Create Procedure cria_edicao_e_palcos(
nome varchar(60),
localidade varchar(60),
local varchar(60),
data_inicio date,
data_fim date,
lotacao int,
num_palcos int)

BEGIN
DECLARE nEdicao numeric;
  DECLARE i INT;
  SET i = 1;

  INSERT INTO edicao(nome,localidade,local,data_inicio,data_fim,lotacao)
  VALUES(nome,localidade,local,data_inicio,data_fim,lotacao);

set nEdicao = (SELECT MAX(numero) FROM edicao);

  WHILE i <= num_palcos DO
    INSERT INTO palco(codigo,edicao_numero,nome)
    VALUES(i,nEdicao, (SELECT CONCAT('Palco ', i)) );
    SET i = i + 1;
  END WHILE;
END

```

2.3 Funções

F1-

```
Begin
      declare totalEdicoes numeric;
      declare lucroTotal numeric;
      declare media double;
      set totalEdicoes = (SELECT COUNT(*) FROM edicao);
      set lucroTotal =  (SELECT SUM(preco) FROM tipo_de_bilhete, bilhete WHERE
bilhete.tipo_de_bilhete_id=tipo_de_bilhete.id AND bilhete.devolvido=0) - (SELECT
SUM(cachet) FROM contrata);
      set media = lucroTotal / totalEdicoes;
      return(media);
END

```

F2-

```
begin
      declare n_participantes numeric;
      declare edicao_n numeric;
      set edicao_n = (SELECT numero FROM edicao WHERE (data_inicio = (SELECT
MAX(data_inicio)
                        FROM edicao)) );
set n_participantes = (Select      count(participante_codigo)
                FROM contrata
                where Edicao_numero = edicao_n);
      return (n_participantes);
END

```

3-PESQUISA DE DADOS: SQL QUERIES E VIEWS

As queries Q1, Q3, Q6 e Q7 foram criadas como procedures porque estas requerem a inserção de argumentos para irem buscar os dados.

Q1-

```
1. create procedure Q1_Cartaz(in nEdicao integer)
2.
3. BEGIN
4.        SELECT nome, dia_festival_data
5.        FROM participante,contrata
6.        WHERE participante.codigo=contrata.Participante_codigo AND contrata.Edicao_numero
= nEdicao
7.        ORDER BY dia_festival_data ASC, cachet DESC;
8. END
9.
```

Q2-

Nesta query foi analisado que o preço estava inserido no tipo_de_bilhete, logo este não estava discrimado por dia, por isso decidiu-se dividir o preço do tipo_de_bilhete pelo número de dias que este dá acesso.

```
1. SELECT df.data,
2.        df.qtd_espetadores,
3.        SUM(CASE WHEN b.devolvido = 0 THEN tb.preco/(SELECT COUNT(tipo_de_bilhete_id)
FROM acesso WHERE tipo_de_bilhete_id=tb.id) ELSE 0 END) AS Faturacao
4. FROM dia_festival df,acesso a, tipo_de_bilhete tb , bilhete b
5. WHERE df.data = a.Dia_festival_data AND
6. a.Tipo_de_bilhete_id = tb.id AND
7. tb.id = b.Tipo_de_bilhete_id
8. GROUP BY df.data;
9.
```

Q3-

```
1. create procedure Q3_Qtd_espetadores_no_dia(in dia date)
2.
3. SELECT qtd_espetadores
4. FROM dia_festival
5. WHERE data=dia
6.
```

Q4-

```
1. SELECT edicao_numero AS Edição, estilo.nome AS Estilo, COUNT(estilo.codigo) AS
Qtd_artistas
2. FROM `estilo_de_artista`, participante, estilo , contrata
3. WHERE
4. contrata.Participante_codigo=participante.codigo AND
5. estilo_de_artista.Participante_codigo=participante.codigo AND
6. estilo_de_artista.Estilo_codigo=estilo.codigo
7. GROUP BY Edicao_numero, estilo.codigo;
8.
```

Q5-

```
 1. SELECT nome,YEAR(CURDATE()) - YEAR(Dia_festival_data) as Anos,cachet
 2. FROM participante,contrata
 3. WHERE participante.codigo=contrata.Participante_codigo
 4. AND (nome,Dia_festival_data) in (SELECT nome, Max(dia_festival_data)
 5.                                  FROM participante, contrata
 6.                                  WHERE
(participante.codigo=contrata.Participante_codigo) AND (SELECT
DATEDIFF(CURDATE(),dia_festival_data)>0)
 7.                                  GROUP BY nome)
 8.
 9.
10.
11. UNION
12. SELECT nome, 'Nunca' as Anos, NULL as cachet
13. FROM participante
14. WHERE NOT EXISTS (
15.     SELECT nome
16.     FROM contrata
17.     WHERE participante.codigo = contrata.Participante_codigo
18. );
19.
```

Existe uma limitação relacionada com o enunciado, em que no Q6 e Q7, o parâmetro de pesquisa é nome do jornalista e não o seu num_carteira_profissional, isto faz com que se houver mais do que um jornalistas com o mesmo nome vai gerar um incoerência de dados, passando a solução por passar o nome do jornalista para UNIQUE ou alterar o parâmetro de pesquisa para o num_carteira_profissional.

Q6-

```
1. create procedure Q6_Entrevistado_por(in nEdicao integer, nomeJornalista Varchar(100))
2.
3. BEGIN
4.         SELECT DISTINCT(participante.nome)
5.         FROM participante
6.         WHERE participante.codigo IN (SELECT Participante_codigo FROM entrevista WHERE
Jornalista_num_carteira_profissional = (SELECT num_carteira_profissional FROM jornalista
WHERE nome = nomeJornalista)
7.         AND data IN (SELECT data FROM dia_festival WHERE Edicao_numero = nEdicao));
8. END
9.
```

Q7-

```
1. create procedure Q7_Ainda_nao_entrevistados_por(in nomeJornalista Varchar(100))
2.
3. SELECT DISTINCT(participante.nome)
4.         FROM participante
5.         WHERE participante.codigo NOT IN (SELECT Participante_codigo FROM entrevista
WHERE Jornalista_num_carteira_profissional = (SELECT num_carteira_profissional FROM
jornalista WHERE nome = nomeJornalista)
6.         AND data IN (SELECT data FROM dia_festival WHERE Edicao_numero = (SELECT numero
FROM edicao WHERE (data_inicio = (SELECT MAX(data_inicio)
7.                             FROM edicao))));
8.
```

4 Protótipo do Sistema Web-Based(HTML/PHP + SQL):

Diagrama: