Universidade Federal da Paraíba

Banco de Dados

Projeto - Parte 3

Professor:
Alan Kelon Oliveira de Moraes

Alunos:
José Ricardo Bezerra de Oliveira
Julyanna de Azevedo Silva
Luiz Eduardo Dias dos Santos
Ruy de Morais e Silva
Thais Duarte Brito

17 de junho de 2022

Conteúdo

1 Especificação do Projeto

Uma Livraria, independente de seu tamanho, precisa de uma organização de seu acervo. Para tal, é indispensável catalogar cada livros de maneira correta, a fim de evitar transtornos. Além disso, fazer o cadastro de cada cliente corretamente é essencial para manter o controle, e contata-lo caso necessário.

Cada livro possui sua própria identificação: nome, gênero, autor, quantidade de exemplares, editora e seu preço. Para controle de estoque dos livros, o sistema poderá mostrar a quantidade de exemplares que ainda restam de cada título.

O sistema também poderá mostrar a relação de livros mais vendidos, bem como poderá recomendá-los, caso o cliente deseje. Além disso, para realizar a compra de um determinado livro, informações do cliente e do livro deverão ser levadas em consideração. E, ao finalizar a compra, serão registradas a data da compra.

O livro pode ser escrito por um autor (que possui nome,sobrenome e um atributo identificador), que por sua vez pode escrever um ou mais livros. Analogamente, editora pode publicar diversos livros, sendo que um livro só pode ser lança por uma única editora. Por último, cada livro se adéqua a pelo menos um gênero.

O sistema da livraria se direciona para o uso dos funcionários, pois possuem controle total sobre o cadastro dos livros, cliente, bem como a compra e as recomendações e estoques.

2 Modelo DER

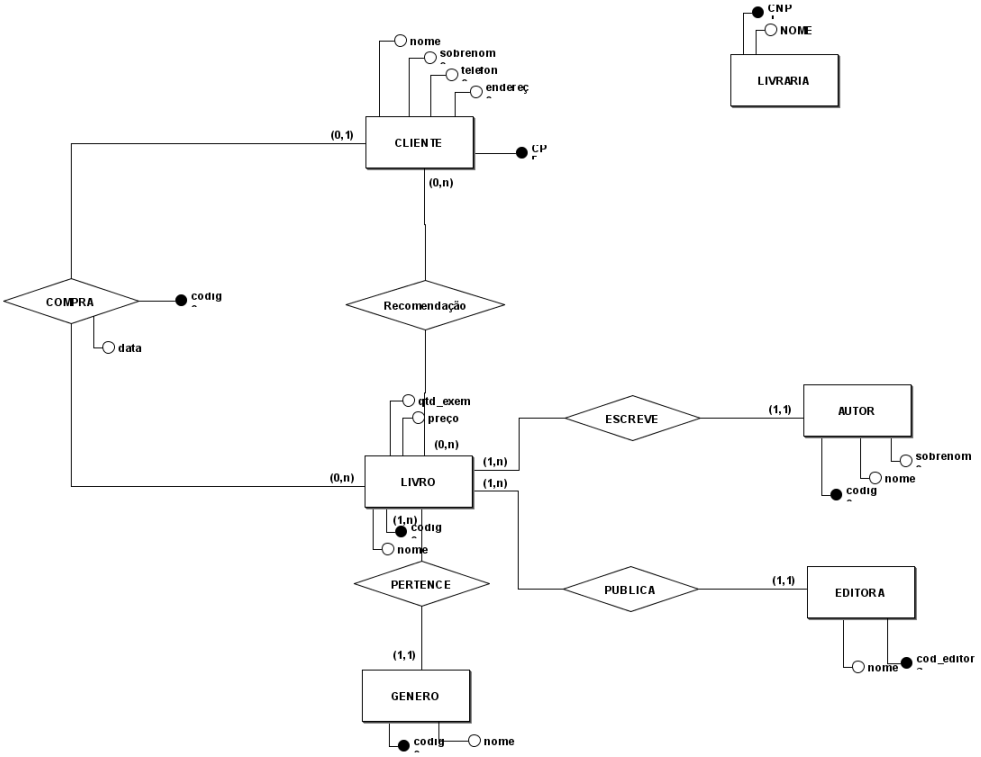


Figura 1: DER da livraria

2.1 Cliente

A entidade **CLIENTE** abstrai informações relativas à identificação de uma pessoa, como CPF(identificador), nome, sobrenome, telefone e endereço. Essa entidade, bem como **LIVROS**, são as entidades que serão o centro do projeto.

2.2 Livro

Todo livro possui um nome, um autor, uma editora e um gênero. Nesse caso, optou-se por tratar AUTOR, EDITORA e GÊNERO como entidades, pois mais de um livro podem estar relacionados a elas, o que permite também que o cliente busque um determinado livro por essas informações.

A referência do nome de um **LIVRO** um atributo identificador do mesmo, assim como o seu código de identificação. Para o **AUTOR**, é necessário atribuir dados como nome, sobrenome e também um código de identificação para facilitar a troca de informações. Já na entidade **EDITORA** são atribuídos seu nome e o seu código. E a entidade **GÊNERO** possui atribuídos de código de identificação e nome.

2.3 Compra

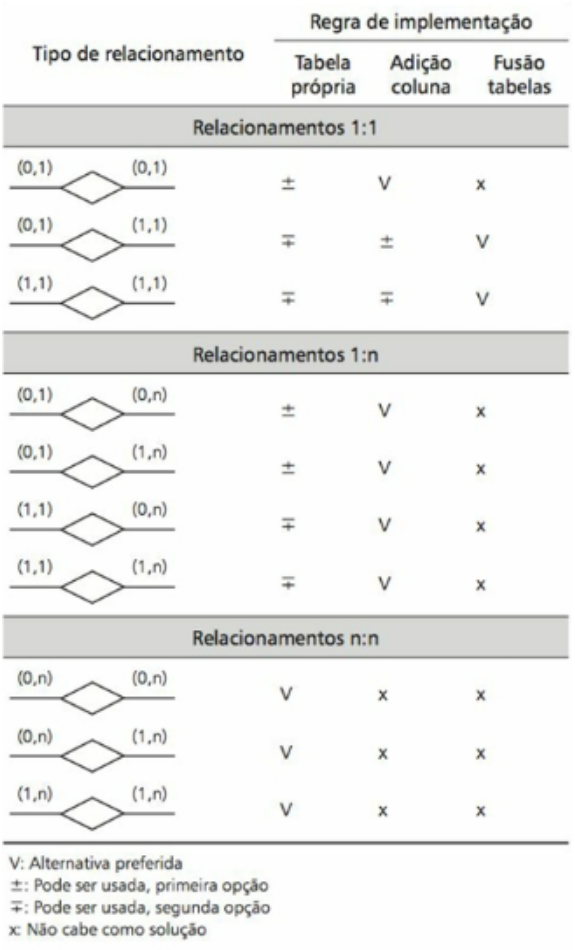
Para que um **CLIENTE** possa realizar uma compra de livro, ele deve possuir cadastro na livraria, adicionando informações como seu nome e sobrenome, seu telefone, seu endereço e o atributo identificador CPF. Ao realizar a compra, o cliente poderá escolher um livro de sua preferencia ou solicitar uma recomendação. O uso do relacionamento opcional **RECOMENDAÇÃO** simula a realidade de um cliente que deseja ou não receber uma recomendação do vendedor. Com o relacionamento **RECOMENDAÇÃO** deseja-se guardar simultaneamente, informações de LIVROS e CLIENTE.

2.4 Livraria

Por fim, a entidade **LIVRARIA** é mantida separada pois esse sistema tanto pode ser utilizado por uma única livraria quanto por um conjunto delas localizadas em uma mesma cidade que compartilham, por exemplo, o mesmo acervo. Ela possui apenas o atributo nome fantasia e o atributo identificador CNPJ.

3 Tradução para Modelo Relacional

A partir da necessidade de entender e iniciar a implementação do nosso banco de dados relacional utilizando a linguagem SQL, é preciso primeiro fazer a tradução do modelo entidade relacionamento para o modelo relacional. Para ajudar na tradução, será necessário utilizar a tabela de recomendações de tradução do livro (Heuser, 2011) [1] .



Tipo de relacionamento	Regra de implementação: Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
(0,1) — (0,1)	±	V	x
(0,1) — (1,1)	∓	±	V
(1,1) — (1,1)	∓	∓	V
Relacionamentos 1:n			
(0,1) — (0,n)	±	V	x
(0,1) — (1,n)	±	V	x
(1,1) — (0,n)	∓	V	x
(1,1) — (1,n)	∓	V	x
Relacionamentos n:n			
(0,n) — (0,n)	V	x	x
(0,n) — (1,n)	V	x	x
(1,n) — (1,n)	V	x	x

V: Alternativa preferida
±: Pode ser usada, primeira opção
∓: Pode ser usada, segunda opção
x: Não cabe como solução

Figura 2: tabela de implementação de relacionamentos

3.1 Relacionamentos 1:n

Neste caso, é recomendado a adição de colunas na tabela correspondente à entidade com cardinalidade máxima 1. Ela vai conter: chaves estrangeira das entidades relacionadas e coluna correspondente ao atributo do relacionamento, caso exista.

No exemplo do relacionamento **"Publica"**, ele possui duas entidades relacionadas(livro e editora). Contudo, nesta relação, livro possui cardinalidade máxima 1, logo será nele que será inserida a chave estrangeira (cód.editora) que referencia a tabela "Editora". O mesmo se repete com os relacionamentos **"Escreve"**, **"Pertence"** e **"Compra"**.

3.2 Relacionamento Opcional n:n

Neste caso,a a tradução do modelo entidade relacionamento para cardinalidades máximas (n:n) pode ser implementada como uma tabela própria que possui colunas referentes aos identificadores de outras tabelas(chave estrangeira) e o atributo identificador do relacionamento, guardando informações simultaneamente como no caso da tabela COMPRA, no caso de recomendação não foi necessário um atributo identificador.

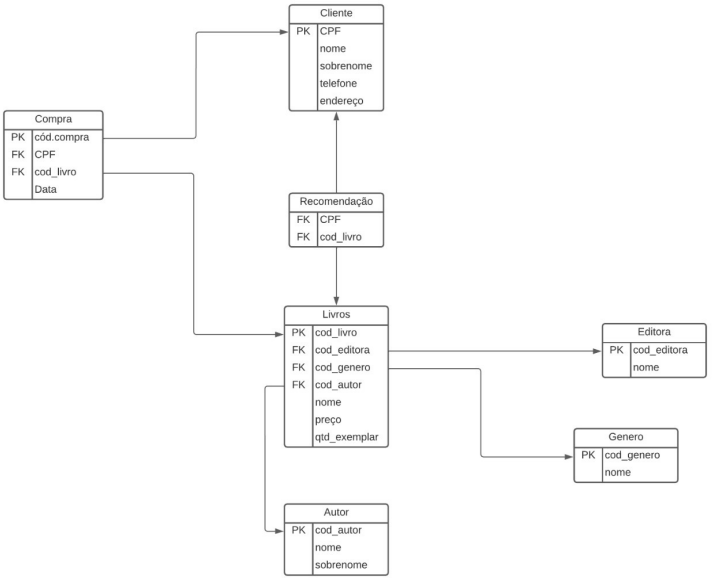


Figura 3: Modelo Relacional da Livraria

4 Esquema SQL

```sql
CREATE TABLE editora(
        cod_editora SERIAL PRIMARY KEY,
        nome_editora VARCHAR(25) NOT NULL
);

CREATE TABLE genero(
        cod_genero SERIAL PRIMARY KEY,
        nome_genero VARCHAR(25) NOT NULL
);

CREATE TABLE autor(
        cod_autor SERIAL PRIMARY KEY,
        nome_autor VARCHAR(25) NOT NULL,
        sobrenome_autor VARCHAR(25) NOT NULL
);

CREATE TABLE cliente(
        cpf_cliente VARCHAR(11) PRIMARY KEY,
        nome_cliente VARCHAR(25) NOT NULL,
        sobrenome_cliente VARCHAR(25) NOT NULL,
        telefone_cliente VARCHAR(9) NOT NULL,
        endereco_cliente VARCHAR(50) NOT NULL
);

CREATE TABLE livro(
        cod_livro SERIAL PRIMARY KEY,
        cod_editora INT NOT NULL REFERENCES editora(cod_editora),
        cod_genero INT NOT NULL REFERENCES genero(cod_genero),
        cod_autor INT NOT NULL REFERENCES autor(cod_autor),
        nome_livro VARCHAR(25) NOT NULL,
        qtd_exemplar INT NOT NULL,
        preco_livro FLOAT CHECK(preco_livro > 0) NOT NULL
);

CREATE TABLE compra(
        cod_compra SERIAL PRIMARY KEY,
        cpf_cliente VARCHAR(11) NOT NULL REFERENCES cliente(
           cpf_cliente),
        cod_livro INT NOT NULL REFERENCES livro(cod_livro),
        data_compra DATE NOT NULL
);

CREATE TABLE recomendacao(
        cpf_cliente VARCHAR(11) NOT NULL REFERENCES cliente(
           cpf_cliente),
        cod_livro INT NOT NULL REFERENCES livro(cod_livro)
)
```

5 Detalhes da Implementação

O intuito inicial é preciso é fazer um CRUD(CREATE, READ, UPDATE, DELETE), a implementação do projeto de Livraria foi predominantemente em Desenvolvimento Web. As linguagens utilizadas foram HTML, CSS e PHP, além das Query's utilizadas em SQL. O seguinte documento tem o intuito de abstrair a programação em si, apenas será apresentada quando se referir à Banco de Dados, neste caso PostGres.

5.1 Conexão com Banco de Dados

Neste arquivo, foi utilizada a função *pg_connect*, e foram passados como argumentos o *localhost* que é a nossa máquina, o nome do banco de dados, o usuário *postgres*(que foi utilizado nesta disciplina) e a senha do banco de dados.

conexao.php

```php
<?php

    $endereço = 'localhost';
    $banco = 'livraria';
    $usuario = 'postgres';
    $senha = '123';


        //procurar erro da função pg_connect...
    try{
        //$conexao = new PDO("pgsql:host=$endereço;port=5432;dbname=$banco",$usuario,$senha);
        $conexao = pg_connect("host=$endereço port=5432 dbname=$banco user=$usuario password=$senha");
        //echo "Conectado ao Banco de Dados Postgres SQL";
    }catch(PDOException $e){
        echo "Falha ao tentar conectar ao Banco de Dados...";
        die($e->getMessage());
    }

?>
```

Figura 4: código da conexão com o Banco de Dados

5.2 CREATE

Figura 5: página de cadastro para autor

Nesta seção, foi feita a inserção dentro do banco de dados por meio do SQL INSERT, no qual foi selecionada a tabela em que os dados serão inseridos, e depois foram selecionados os valores que serão enviados.

```php
<?php

    include_once('../conexao.php');

    if(isset($_POST['submit'])){
        $nomeA_bd = $_POST['nomeA'];
        $SnomeA_bd = $_POST['SnomeA'];


        $result = pg_query($conexao, "INSERT INTO Autor(nome_autor, sobrenome_autor)
        VALUES('$nomeA_bd','$SnomeA_bd')");

        if($result){
            echo "cadastrado com sucesso!! <br>";
        }
    }

?>
```

Figura 6: código em php da inserção

5.3 READ

Figura 7: página de consulta para autor

Nesta seção, a partir da tabela de compra, que mostra apenas o CPF do cliente e *cod_livro*, precisava-se utilizar uma Query de junção (INNER JOIN) para que se tivesse acesso a uma tabela mais simples de ser lida, e fornecesse informações mais úteis. Por este motivo o INNER JOIN precisaria estar associado a duas tabelas diferentes, uma de Cliente e a outra de Livro, desta forma, o código fica:

```php
<?php
    include_once('../conexao.php'); //conexao

    $consulta = "SELECT cod_compra,nome_cliente,sobrenome_cliente,nome_livro,data_compra,preco_livro
                 FROM compra
                 INNER JOIN cliente
                 ON compra.cpf_cliente = cliente.cpf_cliente
                 INNER JOIN livro
                 ON compra.cod_livro = livro.cod_livro";
    $result = pg_query($conexao,$consulta);
    $linha_check = pg_num_rows($result);

    if($linha_check > 0){

        while($linhas = pg_fetch_assoc($result)){
            echo "<tr><td>" . $linhas['cod_compra']  .
            "</td><td>" . $linhas['nome_cliente']  . " " . $linhas['sobrenome_cliente'] .
            "</td><td>" . $linhas['nome_livro']  .
            "</td><td>" . $linhas['data_compra']  .
            "</td><td> R$" . $linhas['preco_livro'] . "</td></tr>";
        }

    }else{
        echo "sem resultados";
    }

?>
```

Figura 8: código em php da consulta

A partir desta nova tabela, se obtém as informações mais relevantes.

5.4 UPDATE

```php
<?php

    include_once('../conexao.php');//conexao

    if(isset($_GET['updateid'])){
        $id=$_GET['updateid'];

        $result1 = pg_query($conexao, "SELECT * FROM Autor WHERE cod_autor=$id");
        $linhas = pg_fetch_assoc($result1);

            $nome_L=$linhas['nome_autor'];
            $sobrenome_L=$linhas['sobrenome_autor'];

        if(isset($_POST['modifica'])){

            $nomeA_bd = $_POST['nomeA'];
            $SnomeA_bd = $_POST['SnomeA'];

            if($_SERVER['REQUEST_METHOD'] == 'POST'){ }

            $sql = "UPDATE Autor SET nome_autor='$nomeA_bd', sobrenome_autor='$SnomeA_bd' WHERE cod_autor=$id;";

            $result = pg_query($conexao, $sql);
            if($result){
                header('location:../consulta/cAutor.php');
            }
        }
    }
?>
```

Figura 9: código em php do update de autor

Para se atualizar algum valor, caso tenha sido inserido de forma errada na hora do cadastro, utiliza-se a Query UPDATE, desta forma a chave primária daquela coluna permanece inalterada, porém seu valor alternativo que geralmente é o nome será alterado.

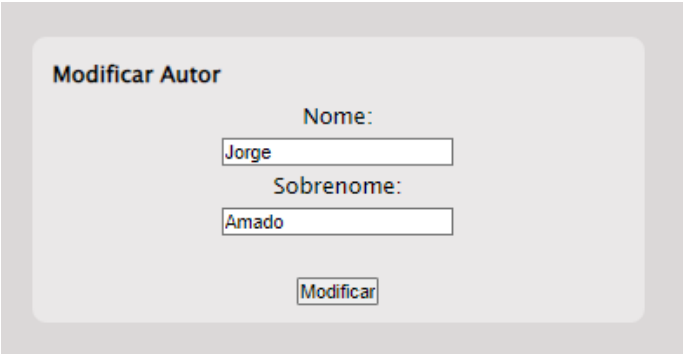


Figura 10: página de update para autor

5.5 DELETE

```php
<?php
    include_once('../conexao.php'); //conexao

    if(isset($_GET['deleteid'])){

        $id=$_GET['deleteid'];

        $consulta = "DELETE FROM Autor WHERE cod_autor=$id";
        $result = pg_query($conexao,$consulta);

        if($result){
            header('location:../consulta/cAutor.php');
        }
    }
?>
```

Figura 11: Código PHP para Deletar o Autor

Por último, foi utilizado um botão para que o valor associado ao seu clique seja deletado por meio da Query DELETE, desta forma, consegue-se excluir do banco de dados algum item específico que foi inserido de forma indesejada.

Referências

[1] Carlos A. HEUSER. *Projeto de banco de dados*. Grupo A, 2011.