UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

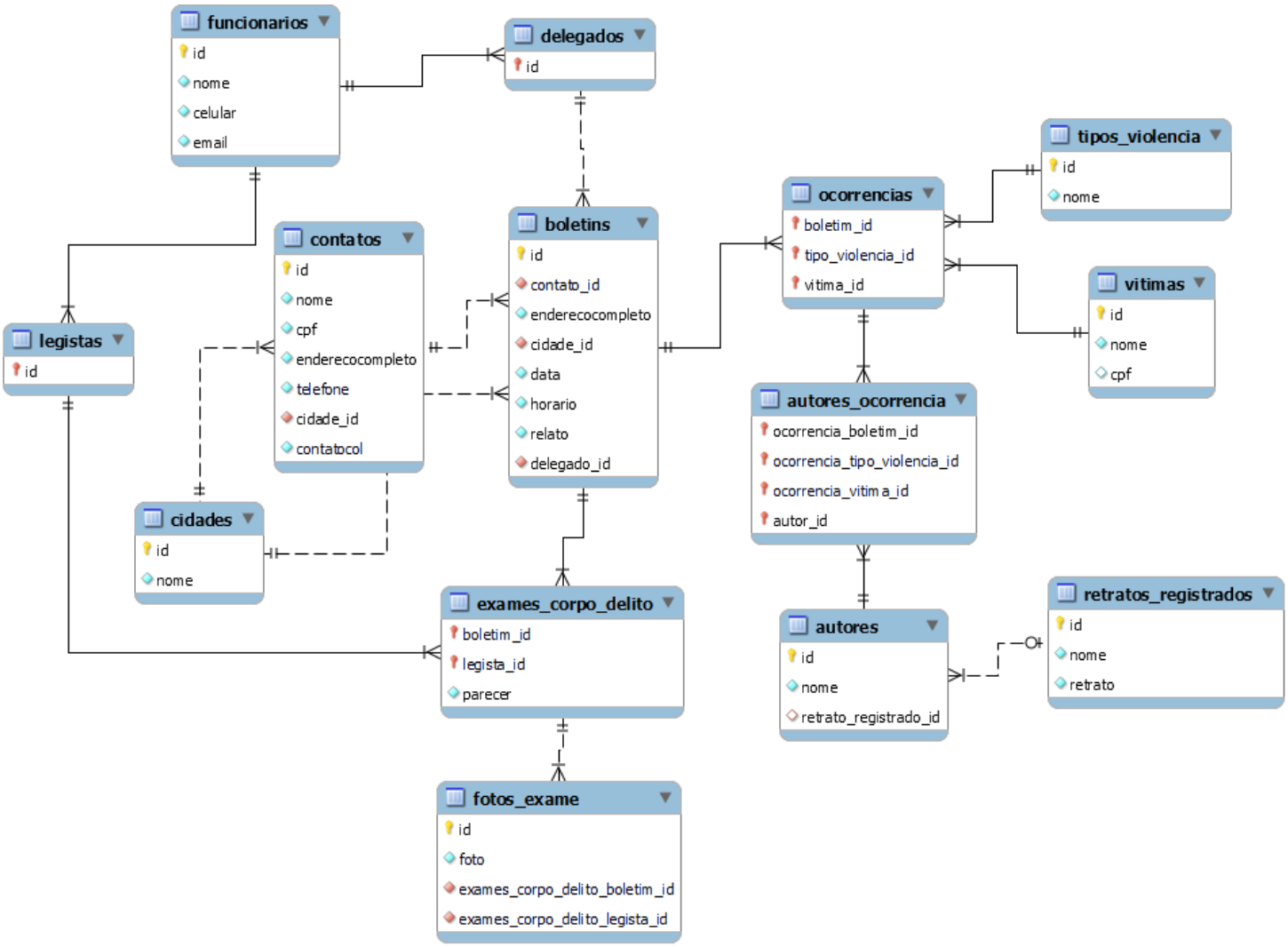
CONCURSO PÚBLICO – 05/2025

Área de Conhecimento: BANCO DE DADOS E PROGRAMAÇÃO DE COMPUTADORES

PROVA ESCRITA – PADRÃO DE GABARITO

QUESTÃO 1

Mínimo de elementos esperados. A solução pode ter variações, porém espera-se que seja atendido o equivalente ao modelo abaixo. Devido a ferramenta utilizada para realizar a modelagem, todas as PKs são equivalentes ao ícone de chave, e todos os ícones vermelhos correspondem às FKs.



QUESTÃO 2

Variações nas respostas precisam ser testadas para comparar se o resultado é equivalente

a) Quantos chamados ocorreram em cada categoria? Mostre somente o nome da categoria com a sua quantidade.

```
select ca_nome, count(*) from chamados join categorias using
   (ca_codigo)   group by ca_nome;
```

b) Quais os nomes dos produtos que tiveram chamados em mais de uma categoria no mês de novembro de 2025? Mostre somente o nome do produto.

```
select pr_nome from chamados join produtos using (pr_codigo)
      where ch_data like "2025-11%" group by pr_nome
      having count(distinct ca_codigo) > 1;
```

OU

```
select pr_nome from chamados join produtos using (pr_codigo)
      where ch_data between "2025-11-01" and  "2025-11-30"
      group by pr_nome having count(distinct ca_codigo) > 1;
```

c) De quais estados os clientes mais fazem chamados (ficar somente com aqueles que tiveram o maior valor)? Mostre somente a sigla do estado.

```
select ci_estado from chamados join clientes using
       (cl_codigo) join cidades using (ci_codigo) group by
       ci_estado having count(*) = (
          select max(conta) from (
             select ci_estado, count(*) conta from chamados
                 join clientes using (cl_codigo)
                 join cidades using (ci_codigo)
                 group by ci_estado) r1
           );
```

d) Quais os nomes dos produtos que possuem a menor quantidade de chamados? Mostre some o nome.

```
select pr_nome from chamados join produtos using
     (pr_codigo) group by pr_nome having count(*) = (
          select min(conta) from (
               select pr_nome, count(*) conta from chamados
                  join produtos using (pr_codigo) group by
                  pr_nome) r1
          );
```

QUESTÃO 3

Exemplo de resposta esperada:

```
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public abstract class Obra {
    private String titulo;

    public Obra(String titulo) {
        this.titulo = titulo;
    }

    public String getTitulo() {
        return titulo;
    }

    public abstract int getPrazo();
    public abstract String getTipo();
}

class Livro extends Obra {
    private List<String> autores;
    private String editora;

    public Livro(String titulo, int anoPublicacao, int numPaginas,
                 List<String> autores, String editora) {
        super(titulo, anoPublicacao, numPaginas);
        this.autores = autores;
        this.editora = editora;
    }

    @Override
    public int getPrazo() {
        return (numPaginas <= 300) ? 7 : 14;
    }

    @Override
    public String getTipo() {
        return “livro”;
    }
}

class Revista extends Obra {
    private int numeroEdicao;

    public Revista(String titulo, int anoPublicicao, int numPaginas, int numeroEdicao)
{
        super(titulo, anoPublicicao, numPaginas);
        this.numeroEdicao = numeroEdicao;
    }

    @Override
    public int getPrazo() {
```

```
        int anoAtual = LocalDate.now().getYear();
        return (anoPublicacao == anoAtual) ? 5 : 20;
    }

    @Override
    public String getTipo() {
        return “revista”;
    }
}

class Usuario {
    private String numeroCadastro;
    private String nome;
    private String telefone;

    public Usuario(String numeroCadastro, String nome, String telefone) {
        this.numeroCadastro = numeroCadastro;
        this.nome = nome;
        this.telefone = telefone;
    }

    public String getNumeroCadastro() {
        return numeroCadastro;
    }

    public String getNome() {
        return nome;
    }
}

class Emprestimo {
    private Obra obra;
    private Usuario usuario;
    private LocalDate dataEmprestimo;
    private LocalDate prazoDevolucao;

    public Emprestimo(Obra obra, Usuario usuario, LocalDate dataEmprestimo) {
        this.obra = obra;
        this.usuario = usuario;
        this.dataEmprestimo = dataEmprestimo;
        this.prazoDevolucao = dataEmprestimo.plusDays(obra.getPrazo());
    }

    public Obra getObra() {
        return this.obra;
    }

    public Usuario getUsuario() {
        return this.usuario;
    }

    public LocalDate getDataEmprestimo() {
        return this.dataEmprestimo;
    }

    public LocalDate getPrazoDevolucao() {
        return this.prazoDevolucao;
    }
```

```
}

public class Biblioteca {

    static List<Obra> obras = new ArrayList<>();
    static List<Usuario> usuarios = new ArrayList<>();
    static List<Emprestimo> emprestimos = new ArrayList<>();
    static Scanner sc = new Scanner(System.in);

    public static void emprestarObra() {
        System.out.print("Título da obra: ");
        String titulo = sc.nextLine();

        Obra obra = obras.stream()
                .filter(o -> o.getTitulo().equalsIgnoreCase(titulo))
                .findFirst().orElse(null);

        if (obra == null) {
            System.out.println("Obra não encontrada.");
            return;
        }

        System.out.print("Número de cadastro do usuário: ");
        String numero = sc.nextLine();

        Usuario u = usuarios.stream()
                .filter(x -> x.getNumeroCadastro().equals(numero))
                .findFirst().orElse(null);

        if (u == null) {
            System.out.println("Usuário não encontrado.");
            return;
        }

        Emprestimo e = new Emprestimo(obra, u, LocalDate.now());
        emprestimos.add(e);

        System.out.println("Empréstimo registrado! Devolver até: " +
e.getPrazoDevolucao());
    }

    public static void listarObrasEmprestadas() {
        System.out.print("Listar (L)ivros ou (R)evistas? ");
        String tipo = sc.nextLine().toUpperCase();

        for (Emprestimo e : emprestimos) {
            if (tipo.equals("L") && e.getObra().getTipo().equals(“livro”)) {
                System.out.println("- " + e.getObra().getTitulo() +
                        " (usuário: " + e.getUsuario().getNome() + ")");
            } else if (tipo.equals("R") && e.getObra().getTipo().equals(“revista”)) {
                System.out.println("- " + e.getObra().getTitulo() +
                        " (usuário: " + e.getUsuario().getNome() + ")");
            }
        }
    }

    public static void listarAtrasados() {
        LocalDate hoje = LocalDate.now();
```

```
        System.out.println("Usuários com empréstimos atrasados:");

        for (Emprestimo e : emprestimos) {
            if (e.getPrazoDevolucao().isBefore(hoje)) {
                System.out.println("- " + e.getUsuario().getNome() +
                                   " (obra: " + e.getObra().getTitulo() + ")");
            }
        }
    }

    public static void consultarPorUsuario() {
        System.out.print("Número de cadastro do usuário: ");
        String numero = sc.nextLine();

        Usuario u = usuarios.stream()
                .filter(x -> x.getNumeroCadastro().equals(numero))
                .findFirst().orElse(null);

        if (u == null) {
            System.out.println("Usuário não encontrado.");
            return;
        }

        System.out.println("Empréstimos do usuário " + u.getNome() + ":");
        for (Emprestimo e : emprestimos) {
            if (e.getUsuario() == u) {
                System.out.println("- " + e.getObra().getTitulo() +
                        ", devolver até " + e.getPrazoDevolucao());
            }
        }
    }
}
```

QUESTÃO 4

Exemplo de resposta esperada:

Definição dos axiomas:
[0,5] A Coesão Alta é fundamental porque um conceito com baixa coesão pode se tornar rapidamente confuso e difícil de manter. Um design coeso garante que as classes sejam focadas e unitárias, tornando-as mais estáveis e fáceis de entender.

[0,5] O Acoplamento Baixo é o princípio que busca minimizar as dependências entre classes. O acoplamento ocorre sempre que existe algum tipo de visibilidade entre objetos. O design orientado a objetos alcança o Acoplamento Baixo pela minimização da quantidade dessas linhas de visibilidade.

2. Importância e Complementaridade

[0,5] A Coesão Alta é considerada um princípio ou axioma em vez de um padrão propriamente dito devido à sua natureza fundamental, abstrata e universal como uma diretriz de qualidade, e não como uma solução específica e concreta.

[0,5] Quando aplicados em conjunto, eles criam um design onde as classes são como peças de Lego: cada peça é focada e bem construída internamente (Alta Coesão) e se conecta a outras peças com o mínimo de dependências desnecessárias (Baixo Acoplamento). Sem a aplicação sistemática desses princípios, as empresas continuarão a produzir software com falta de manutenibilidade e flexibilidade.

Os dois princípios se complementam da seguinte forma:
1. Foco Interno (Coesão): A Coesão Alta garante que cada classe seja bem definida e realize seu trabalho de forma eficiente e isolada.

2. Relações Externas (Acoplamento): O Acoplamento Baixo garante que as classes, mesmo sendo coesas internamente, não criem dependências desnecessárias com muitas outras classes.

Membros da Banca:

gov.br Documento assinado digitalmente
MARCELO DE SOUZA
Data: 29/11/2025 11:37:30-0300
Verifique em https://validar.iti.gov.br

_____
Marcelo de Souza

gov.br Documento assinado digitalmente
PAULO CESAR RODACKI GOMES
Data: 28/11/2025 08:04:27-0300
Verifique em https://validar.iti.gov.br

_____
Paulo Cesar Rodacki

gov.br Documento assinado digitalmente
ADILSON VAHLDICK
Data: 28/11/2025 15:37:43-0300
Verifique em https://validar.iti.gov.br

_____
Adilson Vahldick