MINISTÉRIO DA EDUCAÇÃO

INSTITUTO FEDERAL DO ESPÍRITO SANTO

REITORIA

Avenida Rio Branco, 50 – Santa Lúcia – 29056-255 – Vitória – ES

27 3357-7500

CONCURSO PÚBLICO
EDITAL Nº 03 / 2014

Professor do Magistério do Ensino Básico, Técnico e Tecnológico

ÍNDICE DE INSCRIÇÃO	312
CAMPUS	Santa Teresa
ÁREA/SUBÁREA/ESPECIALIDADE	Ciência da Computação

PROVA DE CONHECIMENTOS ESPECÍFICOS | DISCURSIVA
MATRIZ DE CORREÇÃO

QUESTÃO 01
a) • Elicitação de requisitos ou Análise de Requisitos Levantar, avaliar e formalizar requisitos Traduzir os requisitos em abstrações computacionais para futura implementação • Projeto especificação de arquitetura, tecnologias, algoritmos e estratégias para a codificação do software. • Construção (implementação ou codificação) codificação do software na linguagem e nas tecnologias selecionadas. • Integração Integração dos componentes de software construídos separadamente; • Teste e depuração teste das unidades de código, dos componentes funcionais, dos artefatos instalados em diferentes computadores; • Instalação Implantação dos artefatos de software e treinamento dos usuários • Manutenção de software Manutenção corretiva e evolutiva do software.
b) • Análise Inicial de Requisitos • Ciclos sucessivos das etapas (aqui reside a diferença): Projeto Construção (implementação ou codificação) Teste e depuração Validação Versão intermediária

Análise e formalização de novos requisitos

- Instalação
- Manutenção de software

c)

Gerência de Mudança - mapear, para cada mudança efetuada no sistema, qual foi o motivo, quais os requisitos alterados, incluídos ou excluídos.

Gerência de Versão - dar um controle maior sobre tudo que se altera no projeto de software. Permitir histórico de tudo o que mudou no código, conforme evolução do desenvolvimento ou em função de mudanças nos requisitos (gerência de mudança).

d)

Caixa Preta – teste de validação de entradas e saídas em conformidade com as especificações. Testa o funcionamento do ponto de vista externo (interface);

Caixa Branca – testa a execução do código pelo ponto de vista interno, avaliando desempenho, complexidade, cobertura e estilo de codificação. Testa a qualidade do código escrito.

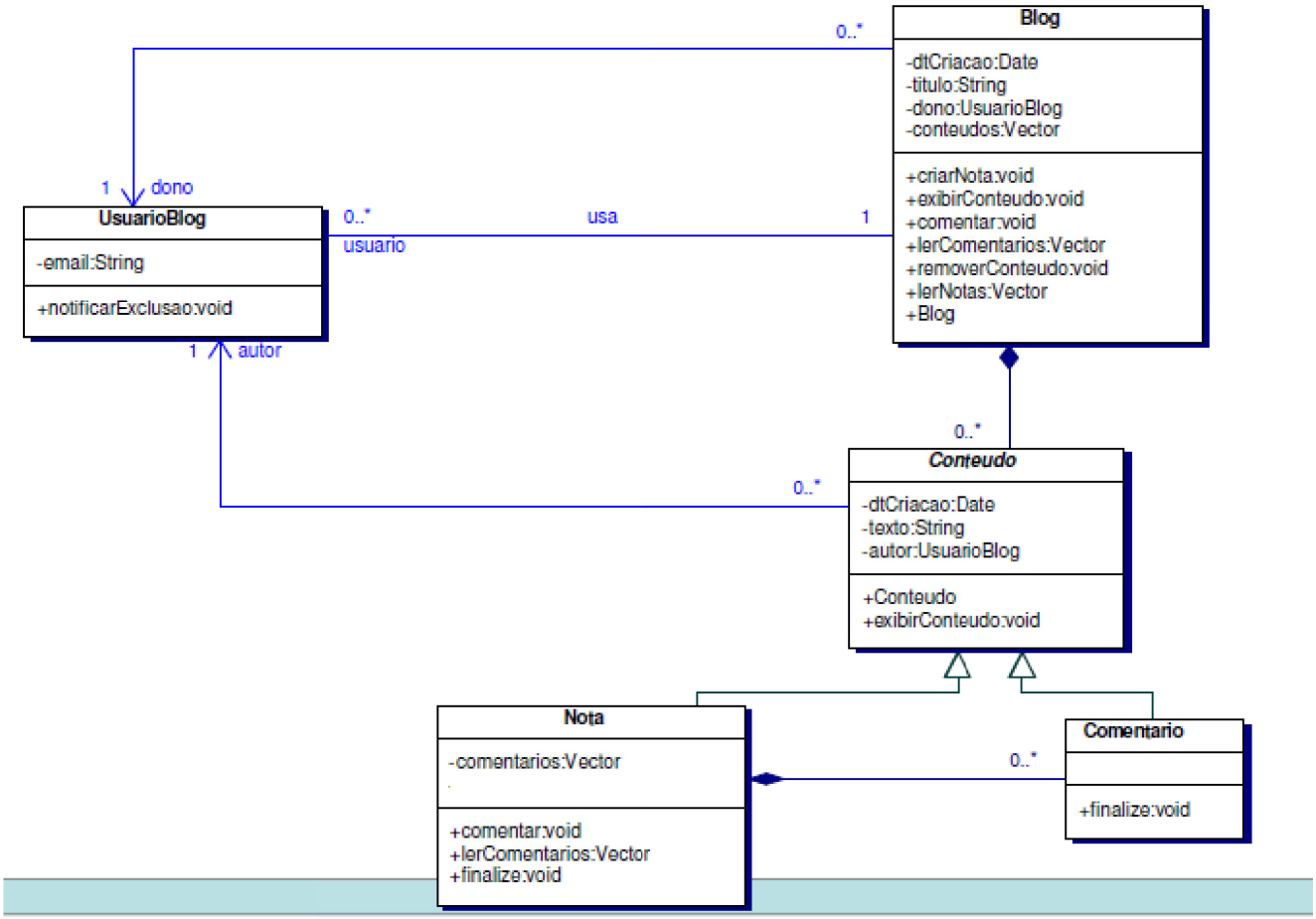
QUESTÃO 02

a)

Classes

- Blog - atributos: título, data de criação, dono do blog, conteudos / métodos: Blog , criarNota, lerNotas, exibirConteudo, comentarNota, lerComentarios, removerConteudo.
- UsuárioBlog (alternativa - UsuárioBlog e DonoBlog herdam de Usuário) – atributos: e-mail / métodos: notificar exclusão
- Conteúdo – atributos: data de criação, texto autor / métodos: conteúdo e exibir ou ler conteúdo,
- Nota - herda de Conteúdo – atributos: comentários / métodos: comentar, exibir ou ler Comentários
- Comentários - herda de Conteúdo – atributos: *nenhum* / métodos: *nenhum*

Herança – Nota e Comentário herdam de Conteúdo (alternativa: UsuárioBlog e DonoBlog herdam Usuário)
Agregações – Nota recebe agregado de Comentários e Blog recebe agregado e Conteúdos
Relações – Autoria (1 UsuárioBlog e 0..* Conteúdo), Utiliza (1 UsuárioBlog e 0..* Blogs) e Dono (1 UsuárioBlog e 0..* Blogs).



b)
Nesta questão, as classe “UsuarioBlog” e “Blog” serão mapeadas como tabelas e os relacionamentos e suas cardinalidades se mantém.
Para o caso da hierarquia de Conteúdo com Nota e Comentário, de acordo com o livro do Elsmari e Navathe (6ª edição em português) em seu capítulo 9 sobre mapeamento da especialização ou generalização, há 4 possibilidades:
1) opção A: Multiplas relações – super-classe e sub-classes;
2) opção B: Múltiplas relações – apenas relações de sub-classes
3) opção C: Relação única com um atributo de tipo
4) opção D: Relação isolada com atributos de múltiplos tipos

Apenas a opção B não pode ser utilizada, pois a classe conteúdo não é abstrata.
O candidato poderia ter optado pelas opções: A, C e D.

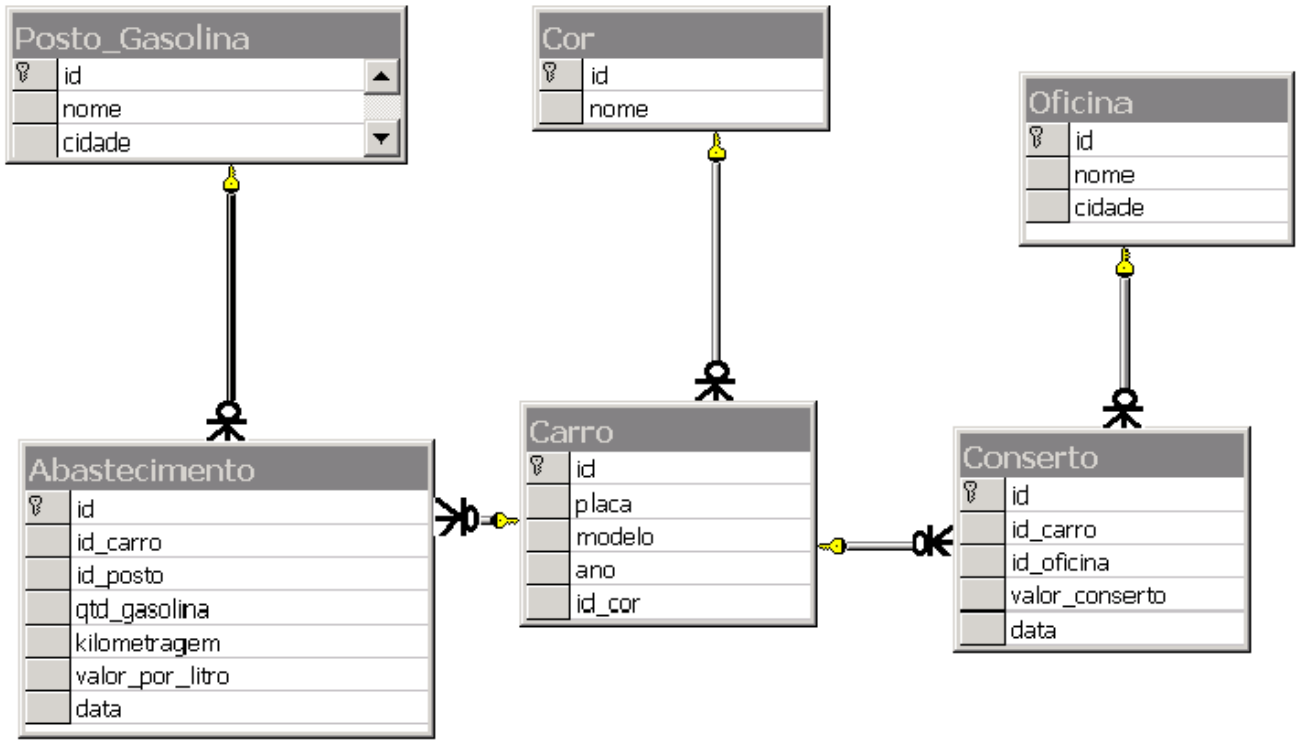
A análise será se o candidato fez o mapeamento correto, de acordo com a opção que ele escolheu, independente se o modelo no item anterior esteja ou não correto.

c)
Atores – UsuárioBlog e DonoBlog (alternativa: DonoBlog herda de UsuárioBlog e ator Sistema)

Casos de Uso – CriarBlog (DonoBlog), CriarNota (DonoBlog), LerConteudo(UsuárioBlog), CriarCometário(UsuarioBlog - inclui: LerConteudo), ApagarConteudo(DonoBlog – incui: LerConteudo e NotificarUsuárioBlog) e NotificarUsuárioBlog(DonoBlog ou Sistema)

QUESTÃO 03

a)



b)

```
Select p.nome, avg(valor_por_litro) as media, sum(qtd_gasolina) as soma
from posto_gasolina p, abastecimento a
where p.id = a.id_posto
group by p.nome
order by soma desc, media
```

c)

```
Select c.placa, sum(qtd_gasolina*valor_por_litro), sum(valor_conserto)
from carro c LEFT OUTER JOIN  abastecimento a on(c.id=a.id_carro)
                LEFT OUTER JOIN conserto co on (c.id = co.id_carro)
group by c.placa
```

d)

Primeiramente, o UNIQUE indica que o conjunto dos valores das colunas não podem se repetir.

O significado do “unique(id_carro, kilometragem)” é que o mesmo carro não pode ser abastecido mais de uma vez com a mesma indicação de kilometragem. O significado do “unique(id_carro, data)” é que o mesmo carro não pode ser abastecido mais de uma vez no mesmo dia.

A restrição “unique(id_carro, kilometragem, data)” significa que o mesmo carro pode ser abastecido no mesmo dia, desde que indique kilometragem diferente OU o mesmo carro poderá ter abastecimentos na mesma kilometragem, desde que sejam em dias diferentes. Note que neste caso, são os três valores simultaneamente que não podem ser repetidos.

QUESTÃO 04

a) Estruturas de dados homogêneas são estruturas usadas para armazenamento de qualquer tipo de dados, porém, os elementos pertencentes a estas estruturas devem ser todos do mesmo tipo, o qual é definido durante a sua declaração. Basicamente as estruturas de dados homogêneas são representadas por vetores, sejam eles:

1- estáticos ou dinâmicos;
2- unidemensionais ou multidimensionais.

Quanto ao número de dimensões, quando os vetores possuem 2 ou mais dimensões, esses são chamados de matrizes.

b) Em memória, essas estruturas encontram-se alocadas de forma contígua (consecutiva) e indexida. Esse tipo de alocação foi estabelecido para que o acesso a qualquer elemento da estrutura possa ser feito de forma rápida e eficiente, bastando para isso que seja(m) informado(s) o(s) índice(s) da posição onde na estrutura se deseja recuperar ou inserir o dado.

c) Algumas vantagens da Lista implementa da com VETOR estático em relação a estrutura de dados LISTA:

- fácil implementação;
- retirada e inserção de dados de forma simples e rápida;
- facilidade de busca de um elemento, bastando para isso o uso de apenas uma estrutura simples de repetição.

A maior desvantagem da Lista implementa da com VETOR estático em relação a estrutura de dados LISTA é o fato de não haver como controlar o tamanho da estrutura em tempo de execução, podendo o VETOR ficar com espaço ocioso, ou falta de espaço dependendo da demanda.

d) Uma possível função em C para o método Bolha é mostrada abaixo:

```c
void bolha ( int vet[], int tam)
{
        int x, y, aux;

        for( x = 0; x < tam; x++ )
        {
                for( y = x + 1; y < tam; y++ ) // sempre 1 elemento à frente
                {
                        if ( vet[y] > vet[x] )
                        {
                                aux = vet[y];
                                vet[y] = vet[x];
                                vet[x] = aux;
                        }
                }
        } // fim da ordenação

        // exibe elementos ordenados
        printf("\n Elementos ordenados (Decrescente):");
        for( x = 0; x < tam; x++ )
        {
                printf("%d ",vet[x]); // exibe o vetor ordenado
        }
}
```

e) Ordenação decrescente do vetor vet[5]={4, 6, 2, 3, 9} com o algoritmo bolha da letra **d**.

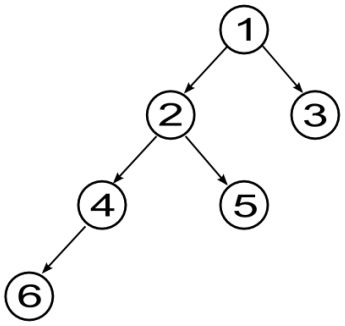
6 4 2 3 9 - não troca
6 4 2 3 9 - não troca
6 4 2 3 9 - não troca
6 4 2 3 9 - troca para 9 4 2 3 6
9 4 2 3 6 - não troca
9 4 2 3 6 - não troca
9 4 2 3 6 - troca para 9 6 2 3 4

9 6 2 3 4 - troca para 9 6 3 2 4
9 6 3 2 4 - troca para 9 6 4 2 3
9 6 4 2 3 - troca para 9 6 4 3 2
9 6 4 3 2 - **vetor ordenado**

QUESTÃO 05

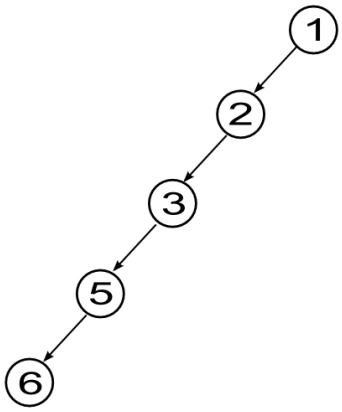
a)

Uma possível solução seria a Árvore de Busca da figura a seguir, onde a ordem de acesso dos nós foi: **1**, **2**, **3**, **4**, **5**, **6**.



b)

Devido a restrição da **Obs.7**, não haverá "retrocesso" (*backtraking*) na árvore, logo, o **nó 4** não será acessado. Sendo assim a Árvore de Busca ficaria com na figura a seguir com a seguinte ordem de acesso: **1**, **2**, **3**, **5**, **6**.



c)

O Algoritmo passaria a fazer uma busca em profundidade, com algumas restrições.

A Busca em Profundidade, como o seu nome sugere, procura o "mais fundo" possível em um grafo. Na busca em profundidade as arestas são exploradas a partir de um vértice **v** recentemente descoberto que ainda possua arestas, saindo dele, que não tenham sido exploradas. Quando todas as arestas de **v** são exploradas, a busca "regressa" (*backtraking*) para explorar as arestas que deixam o vértice a partir do qual **v** foi descoberto. Esse processo continua até descobrirmos todos os vértices acessíveis a partir do vértice de origem original. Se restarem quaisquer vértices não descobertos, então seleciona-se um desses vértices para ser uma nova origem, e a busca se repetirá a partir daquela origem. Esse processo inteiro será repetido até que todos os vértices sejam descobertos. Como o algoritmo do exercício em questão faz a retirada do elemento da pilha no momento que este é acessado, conforme a **Obs.7**, o elemento não será mais acessado em outra oportunidade, fazendo assim como que o algoritmo não "regresse", logo, alguns elementos não serão acessados e a Árvore de Busca, para o grafo em questão, não possuirá todos os vértice do grafo.

Uma aplicação prática para o algoritmo de Busca em Profundidade é a Ordenação Topológica de um digrafo acíclico.