Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina Banco de Dados
AD2 2° semestre de 2013.

Nome:_____

Observações:

1. Prova COM consulta.

Atenção: Como a avaliação à distância é individual, caso seja constatado que provas de alunos distintos são cópias umas das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem sim, ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual.

A entrega das ADs será feita exclusivamente via plataforma. Não serão aceitas ADs entregues via correio ou no polo.

Questão 1 (1 ponto). Considere o seguinte esquema relacional, onde as chaves primárias estão sublinhadas.

Funcionario (<u>fid: integer</u>, fnome: string, salario:real, did:integer) did referencia Departamento Departamento (<u>did: integer</u>, dnome: string)

João era um aluno que sabia bastante álgebra relacional, porém não conhecia muito SQL. Ele elaborou várias consultas em álgebra relacional e precisa de ajuda para "traduzi-las" para SQL. Você pode ajudar o João? Para cada uma das expressões abaixo apresentadas, informe a consulta SQL equivalente.

(a) $\pi_{fnome}(\sigma_{salario < 2000} Funcionario)$

```
SELECT fnome FROM FUNCIONARIO
WHERE salario < 2000
```

(b) $\pi_{dnome}(\sigma_{salario < 1500} Funcionario \bowtie Departamento)$

```
SELECT dnome FROM FUNCIONARIO f, DEPARTAMENTO d
WHERE f.did = d.did
AND f.salario < 1500
```

(c) π_{fnome} (*Funcionario* $\bowtie$ $\sigma_{dnome = "D3"}$ *Departamento*)

```
SELECT fnome FROM FUNCIONARIO f, DEPARTAMENTO d
WHERE f.did = d.did
AND d.dnome = "D3"
```

(d) π *dnome* (π *did* (*Departamento*) - π *did* (*Funcionario*)) $\bowtie$ *Departamento*)

```
SELECT dnome from
DEPARTAMENTO
WHERE did NOT IN (SELECT did FROM DEPARTAMENTO d, EMPREGADO e
WHERE d.did = e.did);
```

OU

π *did* (*Departamento*) - π *did* (*Funcionario* $\bowtie$ *Departamento*)

```
SELECT did from
DEPARTAMENTO
WHERE did NOT IN (SELECT did FROM DEPARTAMENTO d, EMPREGADO e
WHERE d.did = e.did);
```

Questão 2 (2 pontos). Sobre a base de dados da questão anterior, resolver as consultas utilizando SQL. Não usar mais tabelas que o estritamente necessário.

(a) Quais os nomes dos empregados que ganham um salário menor que 2.000.

```
SELECT fnome FROM Funcionario
WHERE salario < 2000;
```

(b) Quais os nomes dos departamentos que possuem empregados que ganham salário até 1.500,00.

```
SELECT dnome FROM Funcionario f, Departamento d
WHERE f.did = d.did
AND f.salario < 1500;
```

(c) Quais os nomes dos empregados que trabalham no departamento "D3".

```
SELECT fnome FROM Funcionario f, Departamento d
WHERE f.did = d.did
AND d.dnome = "D3";
```

(d) Qual o nome do departamento em que trabalha o empregado "Daniel de Oliveira".

```
SELECT dnome FROM Funcionario f, Departamento d
WHERE f.did = d.did
```

AND f.fnome = “Daniel de Oliveira”;

Questão 3 (4 pontos). Considere a seguinte base de dados, usada pela empresa XPTO para controle de vendas e entregas em domicílio. As chaves primárias estão sublinhadas.

```
ESTADO (codEstado, nome) /* Unidade Federativa */

UNIDADE_ESTOQUE (codUnidade, descricao) /* Unidade de Estoque */

CLIENTE (codCliente, nome, endereco, cidade, codEstado, cep, telefone,
percentualDesconto)

      (codEstado) references ESTADO

PRODUTO (CodProduto, nome, preco, codUnidade)

      (codUnidade) references UNIDADE_ESTOQUE

PEDIDO   (identificacao,   tipo,   codCliente,   dtEntrada,   valorTotal,
desconto, dtEmbarque)

      (codCliente) references CLIENTE

ITEM (identificacao, CodProduto, quantidade, valorUnit, valorTotal)

      (CodProduto) references PRODUTO

      (identificacao) references PEDIDO
```

Sobre esta base de dados, resolver as consultas a seguir usando SQL. Não utilize mais tabelas que o necessário nas consultas.

1. Selecionar todas as informações de todos os clientes

```
SELECT *
FROM cliente;
```

2. Selecionar o nome de todos os produtos.

```
SELECT nome
FROM produto;
```

3. Selecionar todas as combinações de cidade, estado e CEP dos clientes registrados, sem repetição.

```
SELECT DISTINCT c.cidade, e.nome, c.cep
FROM cliente c, estado e
WHERE c.codEstado = e.codEstado
```

4. Selecionar todos os pedidos do cliente cujo código seja igual a “09” e cujo valor total seja maior que 50,00.

```
SELECT * FROM pedido
WHERE codCliente= 09 AND valorTotal > 50.0;
```

5. Selecionar todos os pedidos cujo valor total seja menor que 100,00 ou maior que 500,00.

```
SELECT *
FROM pedido
WHERE valorTotal < 100 OR valorTotal > 500;
```

6. Selecionar todos os pedidos cuja data de entrada seja igual a “04/12/2012”.

```
SELECT *
FROM pedido
WHERE dtEntrada = 04/12/2012
```

7. Selecionar todos os pedidos cuja data de entrada seja igual a 02/12/1999 e cujo valor total esteja entre 20,00 e 50,00.

```
SELECT *
FROM pedido
WHERE (valorTotal BETWEEN 20 AND 50)
 AND dtEntrada = 02/12/1999;
```

8. Selecionar todos os clientes cujo código NÃO esteja entre 05 e 25.

```
SELECT *
FROM cliente
WHERE codCliente NOT BETWEEN 5 AND 25;
```

9. Selecionar todos os produtos cujo nome possua a primeira letra = ‘P’. Dica: procure na Web sobre o operador LIKE.

```
SELECT *
FROM produto
WHERE nome LIKE 'P%';
```

10. Selecionar todos os produtos cujo nome possua o trecho 'an' em qualquer posição. Por exemplo, nessa consulta deveriam ser selecionados produtos como “antena”, “manga”, etc. Dica: procure na Web sobre o operador LIKE.

```
SELECT *
FROM produto
WHERE nome LIKE '%an%';
```

11. Selecionar todos os produtos cujo nome comece com C ou F ou M, independente do resto. Dica: procure na Web sobre o operador LIKE.

```
SELECT *
FROM produto
WHERE nome LIKE 'C%'
 OR nome LIKE 'F%'
 OR nome LIKE 'M%' ;
```

12. Selecionar todos os clientes cujo código do estado seja “MG” ou “ES”.

```
SELECT *
FROM cliente  c
WHERE c.codEstado IN('MG','ES');
```

13. Selecionar todos os clientes cujo nome do estado NÃO seja “Rio de Janeiro” nem “São Paulo”.

```
SELECT *
FROM cliente c, estado e
WHERE c.codEstado = e.codEstado
AND e.nome <> 'Rio de Janeiro'
 AND e.nome <> 'São Paulo';
```

14. Selecionar todos os produtos cujo preço seja menor que 20 e a descrição da unidade de estoque seja ‘kilograma’ ou ‘litro’.

```
SELECT *
FROM produto p
WHERE preco < 20
AND p.codUnidade IN (SELECT ue.codUnidade FROM unidade_estoque ue
WHERE ue.descricao = 'kilograma'
OR ue.descricao = 'litro');
```

15. Selecionar todos os produtos cujo código da unidade de estoque seja ‘KG’ ou ‘UM’ e o preço seja maior que 10.

```
SELECT *
FROM produto
WHERE preco > 10
 AND (codUnidade = 'KG' OR codUnidade = 'UM');
```

16. Selecionar todos os produtos cujo preço seja menor que 5, incluindo os produtos cujo preço esteja nulo.

```
SELECT *
```

```
FROM produto
WHERE preco < 5 OR preco is null;
```

17. Selecionar o código do cliente, o nome do cliente e a data de entrada dos pedidos por ordem descendente de data de entrada.

```
SELECT t1.codCliente, t1.nome, t2.dtEntrada
FROM cliente AS t1, pedido AS t2
WHERE t1.codCliente = t2.codCliente
ORDER BY t2.dtEntrada DESC;
```

18. Contar a quantidade de pedidos feitos para o cliente de código 05.

```
SELECT COUNT(*) AS Total_de_pedidos
FROM pedido
WHERE codCliente = 05;
```

19. Obter o MAIOR e o MENOR código de cliente da tabela cliente.

```
SELECT MAX(codCliente) AS maior_id_cliente, MIN(codCliente) AS menor_id_cliente
FROM cliente;
```

20. Obter o MAIOR e o MENOR valor de pedido.

```
SELECT MAX(valorTotal) AS maior_valor_pedido,
MIN(valorTotal) AS menor_valor_pedido
FROM pedido;
```

21. Obter o somatório do valor total geral de todos os pedidos.

```
SELECT SUM(valorTotal) AS valor_total_pedidos
FROM pedido;
```

22. Obter o somatório do valor total das vendas da tabela pedido no período de 02/12/1999 até 04/12/1999.

```
SELECT SUM(valorTotal) AS valor_total_pedidos
FROM pedido
WHERE dtEntrada >= 02/12/1999 AND dtEntrada <= 04/12/1999;
```

23. Obter a média do valor total das vendas da tabela pedidos do ano de 1999.

```
SELECT AVG(valorTotal) AS valor_medio_pedidos
FROM pedido
WHERE dtEntrada >= 01/01/1999 AND dtEntrada <= 31/12/1999;
```

24. Mostrar o código do produto e a média de quantidade por produto vendido.

```
SELECT CodProduto, AVG(quantidade)
FROM item
GROUP BY CodProduto;
```

25. Mostrar a quantidade de clientes por Unidade Federativa.

```
SELECT codEstado, COUNT(*)
AS Qtde_clientes
FROM cliente
GROUP BY codEstado;
```

26. Mostrar a quantidade de clientes por Unidade Federativa para clientes com desconto maior que 10%.

```
SELECT codEstado, COUNT(*) AS Qtde_clientes
FROM cliente
WHERE percentualDesconto > 10
GROUP BY codEstado;
```

27. Mostrar os dados dos produtos para os quais não existe nenhum pedido.

```
SELECT CodProduto, nome
FROM produto
WHERE CodProduto NOT IN (SELECT DISTINCT CodProduto FROM item)
```

28. Mostrar código, nome dos produtos e valor total pedido por produto, dos produtos que já foram pedidos, classificando por nome do produto em ordem descendente. Selecione somente os itens cuja soma total pedido por produto seja maior que 80.

```
SELECT SUM(t1.valorTotal) AS ValorTotalPorProduto, t2.nome AS
nomeProduto, t2.CodProduto AS produtoCodigo
FROM item AS t1, produto AS t2
WHERE t1.CodProduto= t2.CodProduto
GROUP BY produtoCodigo
HAVING ValorTotalPorProduto > 80
ORDER BY nomeProduto DESC;
```

29. Mostrar código do pedido, código e nome do cliente, código e nome do produto e descrição da unidade de estoque do produto de todos os produtos que já foram comprados.

```
SELECT t1.identificacao, t2.codCliente, t3.nome, t4.nome,
t4.codUnidade
FROM item AS t1, pedido AS t2, cliente AS t3, produto AS t4
WHERE t1.identificacao = t2. identificacao
AND t2.codCliente = t3.codCliente
```

AND t1.CodProduto = t4.CodProduto

30. Mostrar código, nome e preço dos produtos cujo preço do produto seja maior que a média de preço de todos os produtos.

```
SELECT CodProduto, nome, preco
FROM produto
WHERE preco > (SELECT AVG(preco) FROM produto);
```

Questão 4 (2 pontos). Considere a seguinte base de dados, usada por uma empresa de computadores (Dell, HP, por exemplo) e que disponibiliza manutenção de computadores e *upgrades*. As chaves primárias estão sublinhadas.

```
CLIENTE (cpf, nome_cli)

COMPUTADOR (no_serie, modelo, cpf);

(cpf) references CLIENTE

/* tabela com os upgrades periódicos programados e realizados – para cada computador, a

empresa cadastra todos os upgrades programados. */

UPGRADE_REVISAO (no_serie, data_programada, data_ultimo_upgrade,

data_executada)

(no_serie) references COMPUTADOR

PEÇA_UPGRADE_REVISAO (no_serie, data_programada, cod_peça, quantidade)

(no_serie, data_programada) references UPGRADE_REVISAO

(cod_peça) references PEÇA

PEÇA (cod_peça, descricao_peça)
```

Sobre a base de dados acima apresentada, resolver as consultas a seguir usando SQL.

ATENÇÃO: Não usar mais tabelas que o estritamente necessário.

(a) Escreva uma instrução SQL para inserir um computador de número de série M4N68T-M, modelo igual a Acer Aspire , e CPF igual a 1010. [0.2 ponto]

```
INSERT INTO COMPUTADOR (no_serie, modelo, cpf)
VALUES ("M4N68T-M", "Acer Aspire", "1010")
```

(b) Escreva uma instrução SQL para excluir a tabela Cliente. [0.2 ponto]

```
DROP TABLE Cliente
```

(c) Faça uma consulta SQL que retorna o nome e cpf de clientes que possuem algum computador cadastrado. O resultado deve estar ordenado pelo nome do cliente. [0.2 ponto]

```
SELECT c.nome, c.cpf
FROM Cliente c, Computador co
WHERE c.cpf = co.cpf
ORDER BY c.nome_cli
```

(d) Faça uma consulta SQL que retorna a descrição das peças que foram usadas no upgrade do computador de número de série M4N68T-M. [0.2 ponto]

```
SELECT p.descricao_peca
FROM Peca p, Peca_Upgrade_Revisao pur
WHERE p.cod_peca = pur.cod_peca
AND pr.no_serie = "M4N68T-M"
```

(e) Faça uma consulta SQL que retorna o nome, o cpf do cliente e a quantidade de computadores que fazem revisão ou upgrade na empresa. [0.2 ponto]

```
SELECT c.cpf, c.nome, COUNT(*)
FROM Cliente c, Computador co
WHERE c.cpf = co.cpf
GROUP BY c.cpf, c.nome
```

Questão 5 (1 ponto). A vídeo Center of Europe Ltda. é uma cadeia de locadoras de DVDs. Ela precisa manter dados sobre os DVDs que têm para locação, os filmes dos DVDs, e locadoras da rede e suas respectivas localidades. Cada DVD para locação tem um número de série único. Um projetista de banco de dados inexperiente propôs a seguinte tabela como solução para armazenar os dados da rede de locadoras. Suponha que cada filme tenha apenas um diretor:

Locadora (CodLocadora, Nome, CodLocal, NomeLocal (no_serie, Filme, Diretor, Ano, Tipo))

O significado de cada coluna é o seguinte:

- CodLocadora: código da locadora
- Nome: nome da locadora
- CodLocal: código da localidade onde a locadora se localiza
- NomeLocal: nome da localidade onde a locadora se localiza
- no_serie: Número de série do DVD
- Filme: nome do filme
- Diretor: nome do diretor
- Ano: ano de filmagem do filme

- Tipo: classificação do filme (ação, policial, terror, etc.)

As dependências funcionais (podendo incluir dependências transitivas) que existem nesta tabela são as seguintes:

CodLocadora → Nome
CodLocal → NomeLocal
CodLocadora → CodLocal
(CodLocadora, no_serie) → Filme, Diretor, Ano, Tipo
Filme → Diretor

1. Assumindo que o profissional não conhece o conceito de normalização, explique para ele em que forma normal encontra-se a tabela.

   Não se encontra normalizada.

2. Caso a tabela não se encontre na terceira forma normal, ensine ao profissional como a transformar para a terceira forma normal. Mostre cada forma normal intermediária, entre aquela em que a tabela se encontra e a terceira forma normal.

   Passagem para a Primeira Forma Normal

   Locadora (CodLocadora, Nome, CodLocal, NomeLocal)
   DVD (CodLocadora, no_serie, Filme, Diretor, Ano, Tipo)

   Passagem para a Segunda Forma Normal

   Já está.

   Passagem para a Terceira Forma Normal

   Locadora (CodLocadora, Nome, CodLocal)
   Localidade (CodLocal, NomeLocal)
   DVD (no_serie, Filme)
   Filme (Filme, Diretor, Ano, Tipo)