MAC0439 – Laboratório de Bancos de Dados
Revisão para a Prova — 18 de maio de 2018

Questão 1 – *Extensible Markup Language* (XML) (valor: 2,5 pontos)

Considere o *XML Schema* a seguir:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Voos" type="tVoos"/>
  <xs:complexType name="tVoos">
    <xs:sequence>
      <xs:element name="Voo" type="tVoo" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="tVoo">
    <xs:sequence>
      <xs:element name="Passageiro" type="tPassageiro" minOccurs="2" maxOccurs="250"/>
    </xs:sequence>
    <xs:attribute name="Numero" type="xs:integer" use="required"/>
  </xs:complexType>
  <xs:complexType name="tPassageiro">
    <xs:sequence>
      <xs:element name="Nome" type="xs:string"/>
      <xs:element name="Assento" type="xs:string"/>
      <xs:choice>
        <xs:element name="Refeicao" type="xs:string"/>
        <xs:element name="Lanche" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="NumPassaporte" type="xs:string" use="required"/>
  </xs:complexType>
</xs:schema>
```

a) Escreva um exemplo de documento XML válido segundo o esquema apresentado acima.

Resposta

```
<Voos>
    <Voo Numero="815">
        <Passageiro NumPassaporte="ABC12345">
            <Nome>Steve Jobs</Nome>
            <Assento>23A</Assento>
            <Refeicao>Massa</Refeicao>
        </Passageiro>
        <Passageiro NumPassaporte="DEF54321">
            <Nome>Jim Gray</Nome>
            <Assento>38E</Assento>
            <Lanche>Misto-quente</Lanche>
        </Passageiro>
    </Voo>
</Voos>
```

b) Descreva o que faz a consulta *XPath* a seguir. Explique também como é a estrutura da resposta dessa consulta (se preferir, use um exemplo para ilustrá-la).

```
doc('voos.xml')//Voo[@Numero="815"]/Passageiro[Refeicao="Vegetariana"]/Assento
```

Resposta

A consulta devolve a lista de assentos ocupados por passageiros que solicitaram a refeição Vegetariana no voo de número 815.
Por exemplo, a resposta para essa consulta aplicada sobre o XML do item (a) é:

```
<Assento>23A</Assento>
```

c) Descreva o que faz a consulta *XQuery* a seguir. Explique também como é a estrutura da resposta dessa consulta (se preferir, use um exemplo para ilustrá-la).

```
<Lista>
{ for $p in doc('voos.xml')//Passageiro
   order by $p/Nome
   return
   <P>
     {$p/Nome}
     <Voo>{$p/../@Numero/data()}</Voo>
     {$p/Assento}
   <P/>
  }
</Lista>
```

Resposta

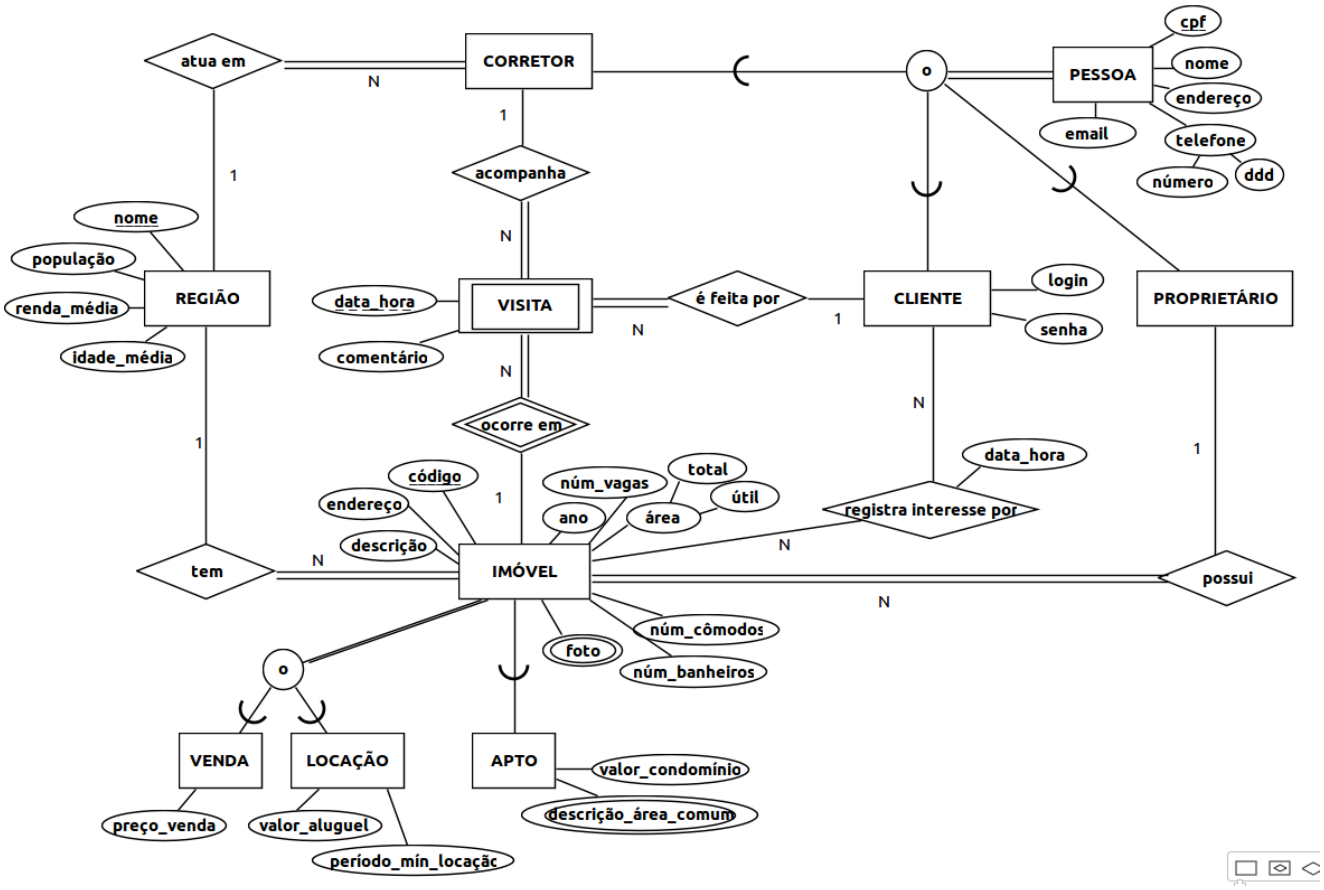
A consulta lista, para cada passageiro, o seu nome, o número do seu voo e o seu assento. Os passageiros são listados em ordem alfabética de nome.
Obs.: Se um mesmo passageiro estiver em mais de um voo, o seu nome pode aparecer mais de uma vez na resposta.

Por exemplo, a resposta para essa consulta aplicada sobre o XML do item (a) é:

```
<Lista>
    <P>
        <Nome>Jim Gray</Nome>
        <Voo>815</Voo>
        <Assento>38E</Assento>
    </P>
    <P>
        <Nome>Steve Jobs</Nome>
        <Voo>815</Voo>
        <Assento>23E</Assento>
    </P>
</Lista>
```

Questão 2 – Projeto Conceitual de BDs de Documentos

O diagrama Entidade-Relacionamento Estendido (EER) abaixo representa o esquema conceitual do BD da empresa Habitat, que atualmente é mantido em um SGBD Relacional. A Habitat é uma imobiliária que trabalha com venda e locação de casas e apartamentos em São Paulo.



Por conta do enorme sucesso do seu site Web para a consulta de imóveis, a Habitat vem vivenciado problemas frequentes de desempenho no seu BD atual. Por essa razão, ela gostaria de passar a gerenciar o seu BD com o MongoDB. Sua tarefa neste exercício é ajudar a Habitat nessa transição.

a) Defina um modelo de dados baseado em coleções de agregados que seja capaz de armazenar todos os dados que a empresa já tem. Para descrever graficamente o seu modelo, você deve usar a notação baseada em UML vista em aula.

b) Para cada uma das consultas a seguir, discorra sobre a facilidade/dificuldade de se escrever a consulta no Mongo Shell e sobre o desempenho esperado para ela no MongoDB, considerando que os dados estarão armazenados na estrutura de coleções de agregados que você definiu no item (a). Justifique suas respostas. Note que o exercício não pede para você escrever as consultas em Mongo Shell!

   Consultas:

   1. Obter uma lista dos imóveis já visitados por um dado cliente e suas características gerais
   2. Obter uma lista dos n imóveis mais visitados
   3. Obter uma lista das n regiões de maior interesse dos clientes

4. Obter todos os dados dos corretores mais "eficazes", ou seja, os que acompanharam mais visitas a imóveis

NÃO FIZ GABARITO PARA ESSA QUESTÃO

Questão 3 – Modificações e Consultas em SQL

Considere o esquema relacional a seguir:

```
FILME(idF, título, ano, diretor)
CRÍTICO(idC, nome)
AVALIAÇÃO(idC, idF, data, nota)
```
– Cada tupla em `AVALIAÇÃO` representa a avaliação (nota de 1 a 5) feita pelo crítico com identificação `idC` ao filme com identificação `idF` em uma dada data.

Escreva as consultas a seguir em SQL:

a) Para cada filme que possui uma nota média de pelo menos 4, adicione 25 ao seu ano de lançamento.

```
update FILME set ano = ano + 25
where idF in (
  select idF from AVALIACAO group by idF having avg(nota) >= 4
);

-- ou

update FILME as F set ano = ano + 25
where 4 <= (
  select avg(nota) from AVALIACAO as A where A.idF = F.idF
);
```

b) Insira uma avaliação com nota 5 da crítica *Isabela Boscov* para todos os filmes existentes no BD. Indique como data da avaliação a data de hoje.

```
insert into Avaliação
  select idc, idF, '2018-05-18' as data, 5 as nota
  from FILME, CRITICO
  were CRITICO.nome like 'Isabela Boscov';
```

c) Em todos os casos em que um mesmo crítico avaliou um mesmo filme duas vezes e deu uma nota maior na segunda vez, devolva o nome do crítico e o título do filme.

```
select nome, titulo
from FILME F, CRITICO C, AVALIACAO A1, AVALIACAO A2
where A1.idF = A2.idF and A1.idC = A2.idC and
      A1.data < a2.data and A1.nota < A2.nota
      and F.idF = A1.idF and C.idC = A1.idC
```

d) Para cada crítico armazenado no BD, exiba o nome e a maior nota atribuída por ele a um filme durante o ano de 2015. O nome de um crítico deve aparecer na resposta mesmo quando ele não tiver realizado nenhuma avaliação em 2015; nesse caso, pode-se considerar que a maior nota atribuída por ele no ano é NULL.

```
select nome, maiorNota from CRITICO natural left outer join
       ( select idC, max(nota) as maiorNota from AVALIACAO
         where data >= '2015-01-01' and data <= '2015-12-31'
         group by idC ) as temp
```

e) Para cada filme que recebeu pelo menos 10 avaliações, mostre todos os dados do filme e a nota média que ele recebeu em suas avaliações.

```
select F.*, A.notaMedia from FILME as F natural join
       ( select idF, AVG(nota) as notaMedia from AVALIACAO
         group by idF having COUNT(*) >= 10) as A
```

f) Liste o título de cada filme que recebeu avaliações de todos os críticos cadastrados no BD.

```
select título from FILME as F where not exists (
    (select idC from  CRITICO)
    except
    (select idC from AVALIACAO where idF = F.idF))
```

Questão 4 – Visões em SQL

a) Usando o esquema da Questão 3 como base, escreva um comando em SQL para criar uma visão chamada `BoasAvaliações` que contenha todos os dados das avaliações que possuem nota maior que 3 e que foram feitas por um crítico que tenha *Spielberg* em seu sobrenome. Defina a visão de forma que ela seja atualizável, ou seja, possa sofrer inserções, alterações e remoções de tuplas.

Resposta:

```
CREATE VIEW BoasAvaliações AS
    SELECT * FROM AVALIACAO WHERE nota >  3 AND idC IN
    (SELECT idC FROM CRITICO WHERE nome like '% %Spielberg%');
```

b) Dê um exemplo de uma inserção ou alteração "com anomalias" na visão `BoasAvaliações` – ou seja, um comando INSERT ou UPDATE que seja aceito como válido e que cause uma modificação no BD, mas cuja a modificação realizada por ele não seja refletida na visão em questão. Justifique a sua resposta.

Resposta:

```
INSERT INTO BoasAvaliações VALUES (100, 200, now(), 2);
-- ou
UPDATE BoasAvaliações SET nota = 2;
```

O primeiro comando inserirá uma tupla que não aparecerá na visão `BoasAvaliações`, porque a nota nela é menor que 3.

O segundo comando atribuirá o valor 2 como nota em todas as avaliações que aparecem `BoasAvaliações`, fazendo com que elas deixem de aparecer na visão.

Questão 5 – *Stored Procedures*

A *stored procedure* a seguir usa o esquema relacional da Questão 3. Descreva em detalhes o que ela faz e o que ela devolve quando executada.
Dica: No PostgreSQL, a função `extract` permite extrair os componentes (dia, mês e ano) de uma data. Exemplos: `extract(month from d)` e `extract(year from d)`.

```
CREATE TYPE NOTA_MÉDIA_MÊS AS (mês INT, ano INT, média FLOAT);

CREATE OR REPLACE FUNCTION AvaliaçãoPorMês(idFilme INT) RETURNS SETOF NOTA_MÉDIA_MÊS AS $$
DECLARE   mês_atual INT;
          ano_atual INT;
          soma_nota FLOAT;
          cont INT;
          tupla RECORD;
BEGIN
  mes_atual = 0;
  ano_atual = 0;
  FOR tupla IN SELECT nota, extract(month from data) as mês, extract(year from data) as ano
               FROM AVALIACAO WHERE idF = idFilme ORDER BY data LOOP
      IF tupla.mês <> mês_atual OR tupla.ano <> ano_atual THEN
          IF mês_atual <> 0 THEN
              RETURN NEXT (mês_atual as mês, ano_atual as ano, soma_nota/cont as média);
          ENDIF;
          mês_atual = tupla.mês;
          ano_atual = tupla.ano;
          soma_nota = tupla.nota;
          cont = 1;
      ELSE
          cont = cont + 1;
          soma_nota = soma_nota + tupla.nota;
      ENDIF;
  END LOOP;

  IF mês_atual <> 0 THEN
      RETURN NEXT (mês_atual as mês, ano_atual as ano, soma_nota/cont as média);
  ENDIF;
  RETURN;
END; $$ LANGUAGE 'plpgsql';
```

Resposta:
A função recebe como parâmetro de entrada o identificador de um filme e devolve como valor de retorno um conjunto de registros da forma `(mês, ano, notaMédia)`. A função, para cada mês em que o filme passado como parâmetro recebeu pelo menos uma avaliação, calcula e mostra na resposta a nota média recebida pelo filme no mês em questão. Portanto, o valor de retorno da função é um conjunto de tuplas no formato (mês, ano, nota_média).

Questão 6 – *Triggers*

Considere as tabelas `Estoque`, `RegistroVendas` e `ComprasPendentes`, criadas por meio dos comandos SQL mostrados a seguir:

```
CREATE TABLE Estoque(produto INT PRIMARY KEY, qtdeDisponivel INT, qtdeMinima INT, qtdeCompra INT);
CREATE TABLE RegistroVendas(pedido INT, produto INT REFERENCES Estoque(produto), qtdeVendida INT,
                            PRIMARY KEY(pedido,produto));
```

```
CREATE TABLE ComprasPendentes(produto INT PRIMARY KEY REFERENCES Estoque(produto), qtde INT,
                              data DATE);
```

Explique o que faz cada um dos três *triggers* definidos sobre o esquema acima. A resposta deve indicar também quando cada *trigger* é disparado, e quando e como sua respectiva ação é executada:

(1)
```
CREATE OR REPLACE FUNCTION F1() RETURNS TRIGGER AS $$
DECLARE x INTEGER;
BEGIN
   SELECT qtdeDisponivel INTO x FROM Estoque WHERE NEW.produto = produto;
   IF x < NEW.qtdeVendida THEN
      RETURN NULL;
   ELSE
      UPDATE Estoque SET qtdeDisponivel = qtdeDisponivel - NEW.qtdeVendida
      WHERE NEW.produto = produto;
      RETURN NEW;
   END IF;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T1
BEFORE INSERT ON RegistroVendas
FOR EACH ROW EXECUTE PROCEDURE F1();
```

Resposta:

Esse *trigger* é executado antes de toda inserção de um registro na tabela `RegistroVendas`. Antes de registrar uma venda de um determinado produto na tabela `RegistroVendas`, o *trigger* verifica se a quantidade disponível do produto no estoque é maior do que a quantidade vendida do produto. Caso a quantidade do produto em estoque não seja suficiente, o *trigger* impede que a venda seja registrada na tabela. Senão, ele atualiza a quantidade disponível do produto no estoque, subtraindo dela a quantidade vendida.

(2)
```
CREATE OR REPLACE FUNCTION F2() RETURNS TRIGGER AS $$
DECLARE x INTEGER;
BEGIN
   IF (NEW.qtdeDisponivel < NEW.qtdeMinima) THEN
      SELECT COUNT(*) INTO x FROM ComprasPendentes WHERE produto = NEW.produto;
      IF x = 0 THEN
          INSERT INTO ComprasPendentes VALUES (NEW.produto, NEW.qtdeCompra, now());
      END IF;
   END IF;
   RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T2
AFTER UPDATE OF qtdeDisponivel ON Estoque
FOR EACH ROW EXECUTE PROCEDURE F2();
```

Resposta:

Esse *trigger* é executado sempre que o atributo `qtdeDisponivel` for alterado em um registro na tabela `Estoque`, mas somente depois que a alteração já tiver sido realizada. Ele verifica se a quantidade disponível do produto no registro alterado é inferior a quantidade mínima do produto que deveria haver em estoque. Caso seja, ele verifica se a reposição desse produto no estoque já foi solicitada, consultando a tabela `ComprasPendentes`, e, se não tiver sido, ele registra em

`ComprasPendentes` uma solicitação de compra para o produto, indicando como quantidade a ser comprada a quantidade mínima definida para o produto na tabela Estoque.

(3)
```
CREATE TRIGGER T3
AFTER DELETE ON ComprasPendentes
REFERENCING OLD TABLE AS O
FOR EACH STATEMENT
BEGIN
   UPDATE Estoque as E
   SET qtdeDisponivel = qtdeDisponivel +
                        (SELECT qtde FROM O WHERE O.produto = E.produto)
   WHERE E.produto IN (SELECT produto FROM O);
END;
```

Obs.: Diferentemente dos anteriores, o *trigger* T3 não está na sintaxe do PostgreSQL, mas sim na do padrão SQL.

Resposta:

Esse *trigger* é executado uma vez para cada comando de remoção na tabela `ComprasPendentes`, depois que a remoção já tiver sido realizada. A tabela `O` (OLD TABLE) contém as tuplas removidas de `ComprasPendentes` pelo comando de remoção que disparou o *trigger*. Como cada tupla em `ComprasPendentes` se refere a uma solicitação de reposição de estoque ainda não atendida, a remoção de tuplas dessa tabela indica que aquisições de produtos para a reposição do estoque foram feitas. Assim, para cada produto referenciado por uma tupla em O (ou seja, para cada produto reposto), o *trigger* atualizará sua quantidade disponível em estoque adicionando a ela a quantidade comprada do produto.

Questão 7 – Consultas em Sistemas NoSQL

a) Explique o que a seguinte consulta em linguagem Cypher (para Neo4J) faz:

```
MATCH (a:Aluno)-[:CURSOU]->(d1:Disciplina)<-[:MINISTROU]-(p:Professor),
      (p)-[:MINISTROU]->(d2:Disciplina)
WHERE NOT EXISTS ((a)-[:CURSOU]->(d2))
RETURN a.nomeAluno, collect(d2.codDisciplina);
```

Resposta:

Para cada aluno que cursou alguma disciplina, mostra o nome do aluno junto com o conjunto das disciplinas que ele não cursou e que são ministradas por professores que ministraram as disciplinas que ele cursou.

b) Escreva uma consulta em SQL equivalente à consulta para MongoDB apresentada a seguir:

```
db.matriculas.aggregate( [
    { $match: { ano: { $eq: 2016 } } },
    { $group: { _id: "$codDisciplina", contMatriculados: {$sum: 1} } } ,
    { $match: { contMatriculados: { $gte: 50 } } },
    { $sort:  { contMatriculados: -1 } } ] );
```

A seguir é mostrado um exemplo de documento que pode ser encontrado na coleção `matriculas`:

```
{"codDisciplina": "MAC0439", "ano": 2016, "semestre": 2,
 "codAluno": 12345, "notas": [9.3, 6.8, 7.5]}
```

Resposta:

```
select * from (
select codDisciplina, count(*) as contMatriculados
from matriculas where ano = 2016
group by codDisciplina having count(*) > 50 )
order by contMatriculados desc;
```

Questão 8 – Teoria

- O que são dados semiestruturados? Caracterize-os e compare-os com dados estruturados (como os do modelo Relacional, por exemplo).
- Quais são as caracteríticas comumente encontradas em sistemas NoSQL e como elas possibilitam que esses sistemas tenham um desempenho melhor que o dos SGBDs relacionais na execução de operações simples sobre dados?
- Descreva os sistemas NoSQL MongoDB e Neo4J quanto a: (a) ao modelo de dados usado; (b) forma de manipulação dos dados (operações de modificação e consulta); (c) garantias relacionadas a consistência de dados e atomicidade de operações.

NÃO FIZ GABARITO PARA ESSA QUESTÃO