UNIVERSIDADE FEDERAL FLUMINENSE
PÓLO UNIVERSITÁRIO DE RIO DAS OSTRAS
FACULDADE FEDERAL DE RIO DAS OSTRAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

3ª. Avaliação de Banco de Dados – 2º. Sem de 2007
Prof.: Carlos Bazílio
Aluno: **Matrícula:**

1ª Questão) Considere a seguinte relação:

A	*B*	*C*
10	b1	c1
10	b2	c2
11	b4	c1
12	b3	c4
13	b1	c1
14	b3	c4

a) Dada a relação anterior, quais das seguintes dependências podem ser asseguradas para a relação? Se a dependência não puder ser garantida, explique por que especificando as tuplas que causam a violação.
   i. A → B
   ii. B → C
   iii. C → B
   iv.. B → A
   v. C → A
b) A relação anterior tem chaves candidatas? Se tiver, quais são? Se não tiver, por que não tem?

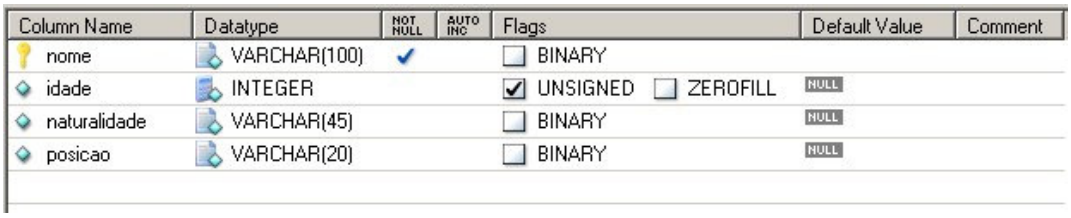
2ª Questão) Considere a seguinte relação referente a publicações:

Livro (Título, NomeAutor, TipoPublicação, Preço, Afiliação, Editora)

Título → Editora, TipoPublicação
TipoPublicação → Preço
NomeAutor → Afiliação

a) Em que forma normal está a relação? Justifique sua resposta.
b) Aplique normalizações até que não se possa decompor mais as relações. Explique cada decomposição.

3ª Questão) Seja a seguinte tabela definida num SGBD para a definição de jogadores de um time:



Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
nome	VARCHAR(100)	✓		BINARY		
idade	INTEGER			UNSIGNED ZEROFILL	NULL	
naturalidade	VARCHAR(45)			BINARY	NULL	
posicao	VARCHAR(20)			BINARY	NULL	

Suponha que desejamos obter os nomes dos jogadores com idade abaixo da média total. Uma possível consulta em SQL seria:

```
select * from
jogador j
where j.idade < (
        select avg(idade)
        from jogador);
```

Caso desejássemos exibir o resultado desta consulta em alguma aplicação Java, poderíamos ter o seguinte trecho de código:

```
// Executa uma consulta SQL à partir do objeto Statement criado e
armazena o resultado numa coleção ResultSet
ResultSet rs = stmt.executeQuery("select * from jogador j where
j.idade < (select avg(idade) from jogador);");

// Percorre o resultado da consulta
System.out.println("Jogadores com idade abaixo da média:");
while (rs.next()) {
      System.out.print("Nome: " + rs.getString("nome"));
      System.out.print(" Posição: " + rs.getString("posicao"));
      System.out.print(" Idade : " + rs.getInt("idade"));
      System.out.println();
}
```

Uma segunda alternativa seria, ao invés de passar a consulta completa, recuperar apenas o conjunto de jogadores e, no código em Java, calcular a média e fazer o filtro. Para tal, utilizamos uma suposta classe "Jogador" que formata cada jogador lido e uma classe Lista para armazenar o conjunto de jogadores lidos:

```
// Executa uma consulta SQL à partir do objeto Statement criado e
armazena o resultado numa coleção ResultSet
ResultSet rs = stmt.executeQuery("select * from jogador;");

// Percorre o resultado da consulta
System.out.println("Jogadores com idade abaixo da média:");
while (rs.next()) {
      List l = new ArrayList();
      Jogador j = new Jogador(rs.getString("nome"),
rs.getString("posicao"));
      l.add(j);
      soma = soma + rs.getInt("idade");
      quant++;
}

// Impressão dos dados dos jogadores filtrados por soma/quant
rs.beforeFirst(); // Inicializa o ResultSet
while (rs.next()) {
```

```
      if (rs.getInt("idade") < soma/quant) {
            System.out.print("Nome: " + rs.getString("nome"));
            System.out.print(" Posição: " + rs.getString("posicao"));
            System.out.print(" Idade : " + rs.getInt("idade"));
            System.out.println();
      }
}
```

Uma terceira alternativa seria fazer a chamada de um procedimento armazenado (*stored procedure*) para retornar o resultado:

```
// Executa uma consulta SQL à partir do objeto Statement criado e
armazena o resultado numa coleção ResultSet
ResultSet rs = stmt.executeQuery("{call novos()}");

// Percorre o resultado da consulta
System.out.println("Jogadores com idade abaixo da média:");
while (rs.next()) {
      System.out.print("Nome: " + rs.getString("nome"));
      System.out.print(" Posição: " + rs.getString("posicao"));
      System.out.print(" Idade : " + rs.getInt("idade"));
      System.out.println();
}
```

, onde a procedure novos() tem o seguinte código:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `provabd`.`novos` $$
CREATE PROCEDURE `provabd`.`novos` ()
BEGIN
  select *
  from jogador j
  where j.idade <
    (select avg(idade)
     from jogador);
END $$

DELIMITER ;
```

Compare cada uma das alternativas com respeito ao tempo de execução, quantidade de dados trafegados na rede, amarração com um SGBD específico, amarração do esquema de banco de dados com a implementação (suponha que são tarefas feitas por pessoas diferentes), tamanho/complexidade do código e facilidade de manutenção (custo de modificação de algum dado, como por exemplo, ao invés de recuperar jogadores abaixo da média de idade, recuperar jogadores abaixo de uma data específica), levando-se em conta que, num sistema, estes dados podem ser requisitados em diversos pontos (diversas chamadas ao SGBD).

4ª Questão) Em bancos de dados distribuídos (bancos de dados cujas tabelas podem estar localizadas em diferentes máquinas servidoras) uma transação que contenha diversas operações sobre estas tabelas é um cenário complexo se comparado com o tratamento de transações feito numa única máquina. Um algoritmo conhecido neste contexto é o "commit de 2 fases". Neste algoritmo temos a figura do coordenador, o qual é responsável por comandar a execução da transação. Suponha que as tarefas em

cada máquina possam executar paralelamente. A figura abaixo representa uma possível troca de mensagens entre um coordenador e *n* máquinas servidoras:

Coordenador	Máquina Servidora$_n$
Envia operação para cada máquina servidora$_n$	
	Recebe operação e inicia execução.
	Caso execute com êxito, envia *ok* para coordenador **sem** executar commit local. Caso contrário, envia *fail*.
Espera pelas mensagens das máquinas servidoras.	
Caso receba alguma mensagem *fail*, envia mensagem de *rollback* para todas as máquinas. Caso contrário, envia uma mensagem de *commit*.	
	Recebe mensagem do coordenador (*rollback* ou *commit*) e executa a operação localmente.

Discuta as possibilidades de problemas neste cenário (pelo menos 3) e como o algoritmo pode se comportar para resolver estes problemas.