INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA - IFPB
Unidade Acadêmica de Informação e Comunicação
CST em Sistemas para Internet

Disciplina: Banco de Dados II

Professora: Damires

Aluno: _____ Matrícula: _____

09 - Exercícios de Revisão

1. Identifique uma situação de **tabelas (duas) com relacionamento e definição de integridade referencial**. Crie as **tabelas** e as **restrições** necessárias. **Insira dados** nas tabelas.
   a. Depois, crie um trigger para realizar um **update cascade** quando a chave primária da tabela mãe for atualizada.
   b. Teste-o. Mostre os testes (*vale 0,3*).
2. Crie uma visão filmcat que apresente: Título, Ano e Categoria, ou seja, os **filmes** e suas **categorias**. Na Categoria, apresente sua descrição (por exemplo, 'Ação'). Verifique os dados da visão.
   a. Em seguida, faça um **trigger** que permita a inserção na **tabela filme a partir da view**. Se a **categoria** ainda não estiver cadastrada, faça sua inserção também. Para isso, verifique se a categoria já existe (use a exceção pré-definida *no_data_found* para ser lançada quando a categoria não existir).
   b. Teste fazendo dois inserts:

      ```
      insert into filmcat values ('Liga da Justiça',2017,'Aventura');

      insert into filmcat values('IT',2017,'TesteTerror');
      ```

   c. Mostre os resultados na view e nas tabelas (*vale 0,3*).
3. Veja a estrutura e dados das tabelas já criadas: Esporte e Atleta. Se não tiver, crie-as e insira dados.

```
Create table Esporte (
      cod_esp                NUMBER       NOT NULL,
      desc_esp               VARCHAR2(25));

Create table Atleta (
    cod_atleta             NUMBER       NOT NULL,
    Nome_atleta            VARCHAR2(25),
    data_nasc                DATE,
     bolsa                    NUMBER (12,2),
    esporte                  NUMBER);

alter table esporte add constraint pk_esp primary key(cod_esp);
alter table atleta add constraint pk_atleta primary key(cod_atleta);
alter table atleta add constraint fk_atl_esp foreign key(esporte) references esporte;
```

4. Crie uma procedure armazenada que atualize as bolsas dos atletas de acordo com um percentual passado como parâmetro. Para isso a **bolsa atual deve ser menor que a média de todas as bolsas**. Use um **cursor** contendo todos os atletas. Faça emitir mensagem (dbms_output) apresentando, para cada atleta, o nome do atleta, valor da bolsa, da média das bolsas e status (atualizado, não atualizado) (*vale 0,2*).

   Exemplo:

   ```
   Nome Atleta = Mario
   Bolsa = 2300
   Média = 2200
   Status: NÃO ATUALIZADO
   ```

5. Crie uma função que, passado o nome do esporte, retorne a quantidade de atletas existentes para ele. Se o esporte passado não existir (veja com **select into**), trate a exceção retornando **zero** (0). Execute a função **dentro de um bloco anônimo** e exiba (com dbms_output.put_line) o resultado. Teste para um **esporte existente** e para um **não existente**. (*vale 0,2*).