MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Alfenas . UNIFAL-MG
Rua Gabriel Monteiro da Silva, 714 . Alfenas/MG . CEP 37130-000
Fone: (35) 3299-1000 . Fax: (35) 3299-1063

Segunda Lista de Exercícios de Banco de Dados **(duplas)**
Prof. Mariane Moreira de Souza
Semestre: 2025/2
Aluno: Thallysson Luis Teixeira Carvalho 2024.1.08.022

Para responder às questões 1 e 2 considere o esquema de um banco de dados relacional de estradas definido abaixo:

```
Estrada(CodEst,NomeEst)
-- Tabela com as estradas na base de dados --

Trecho(CodEst,CodLocIni,CodLocFim, Distancia)
CodEst referencia Estrada
CodLocIni referencia Localidade
CodLocFim referencia Localidade
-- Tabela com os trechos de estrada,
   indicando localidade de início e de fim --

Localidade(CodLoc,NomeLoc,CodLocAbrang,SiglaUF)
SiglaUF referencia UF
CodLocAbrang referencia Localidade
-- Tabela das localidades -
   CodLocAbrang indica a localidade que contém (se houver) a
   localidade em questão --

UF(SiglaUF,NomeUF)
-- tabela com as unidades da federação --
```

1. Considerando o modelo relacional apresentado, mostre o código SQL para as seguintes operações:
    a. Criação das tabelas, assumindo o que for mais lógico no caso das ações engatilhadas ON DELETE/ON UPDATE, definindo as restrições que você achar necessário.

```
CREATE TABLE UF (
   SiglaUF CHAR(2) PRIMARY KEY,
   NomeUF VARCHAR(100) NOT NULL
);


CREATE TABLE Localidade (
   CodLoc INT PRIMARY KEY,
   NomeLoc VARCHAR(100) NOT NULL,
   CodLocAbrang INT NULL,
   SiglaUF CHAR(2) NOT NULL,
   FOREIGN KEY (SiglaUF)
      REFERENCES UF(SiglaUF)
      ON UPDATE CASCADE
```

```
    ON DELETE RESTRICT,
  FOREIGN KEY (CodLocAbrang)
    REFERENCES Localidade(CodLoc)
    ON UPDATE CASCADE
    ON DELETE SET NULL
);

CREATE TABLE Estrada (
  CodEst INT PRIMARY KEY,
  NomeEst VARCHAR(100) NOT NULL
);

CREATE TABLE Trecho (
  CodEst INT NOT NULL,
  CodLocIni INT NOT NULL,
  CodLocFim INT NOT NULL,
  Distancia DECIMAL(8,2) CHECK (Distancia > 0),
  PRIMARY KEY (CodEst, CodLocIni, CodLocFim),
  FOREIGN KEY (CodEst)
    REFERENCES Estrada(CodEst)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  FOREIGN KEY (CodLocIni)
  REFERENCES Localidade(CodLoc)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  FOREIGN KEY (CodLocFim)
    REFERENCES Localidade(CodLoc)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
);
```

b. Possíveis atualizações nas tabelas (uma para cada tabela). *Obs: Mostre o que acontecerá com as tabelas que estiverem referenciando a tabela alvo de atualização, de acordo com as definições de ações engatilhadas feitas na letra a*

UPDATE UF
SET NomeUF = 'Rio Grande do Sul'
WHERE SiglaUF = 'RS';

Efeito nos dados relacionados:

- *Localidade.SiglaUF referencia UF.*
- *Foi definido ON UPDATE CASCADE, então todas as localidades que têm SiglaUF = 'RS' terão seu campo atualizado automaticamente.*

UPDATE Localidade
SET NomeLoc = 'Getulio Vargas'
WHERE CodLoc = 10;

Efeito nos dados relacionados:

- *Trecho.CodLocIni e Trecho.CodLocFim referenciam Localidade*
- *Foi definido ON UPDATE CASCADE, então os trechos que usam CodLocIni ou CodLocFim = 10 são atualizados automaticamente.*

UPDATE Estrada
SET NomeEst = 'Rota das Águas'
WHERE CodEst = 5;

Efeito nos dados relacionados:

- *Trecho.CodEst referencia Estrada*
- *Foi definido ON UPDATE CASCADE, então todos os trechos com CodEst = 5 serão atualizados automaticamente.*

UPDATE Trecho
SET Distancia = 120
WHERE CodEst = 5 AND CodLocIni = 10 AND CodLocFim = 12;

Efeito nos dados relacionados:

- *Como Trecho é a tabela "filha", não afeta outras tabelas.*

c. Possíveis remoções em cada tabela (uma para cada tabela). *Obs: Mostre o que acontecerá com as tabelas que estiverem referenciando a tabela alvo de remoção, de acordo com as definições de ações engatilhadas feitas na letra a*

DELETE FROM UF
WHERE SiglaUF = 'RS';

Efeito nas tabelas relacionadas:

- Localidade.SiglaUF referencia UF.
- ON DELETE RESTRICT: a remoção falhará se houver localidades referenciando a UF

DELETE FROM Localidade
WHERE CodLoc = 10;

Efeito nas tabelas relacionadas:

- Trecho.CodLocIni e Trecho.CodLocFim referenciam Localidade.
- ON DELETE CASCADE RESTRICT: a remoção falhará

DELETE FROM Estrada
WHERE CodEst = 5;

Efeito nas tabelas relacionadas:
- Trecho.CodEst referencia Estrada.
- ON DELETE CASCADE: todos os trechos da estrada serão removidos.

DELETE FROM Trecho
WHERE CodEst = 5 AND CodLocIni = 10 AND CodLocFim = 12;

- Como Trecho é uma tabela filha, a exclusão não afeta outras tabelas.

2. Considerando o modelo relacional apresentado, mostre o código SQL para as seguintes consultas:
   a. Obter os nomes das unidades de federação que possua “Getulio Vargas” como uma de suas localidades e nesta localidade inicie um trecho de uma estrada denominada “Rota das Águas”. *Obs: Mostre a operação utilizando consultas aninhadas (IN ou EXISTS) e sem utilizá-las.*

```
SELECT NomeUF
FROM UF
WHERE SiglaUF IN (
   SELECT SiglaUF
   FROM Localidade
   WHERE NomeLoc = 'Getulio Vargas'
    AND CodLoc IN (
       SELECT CodLocIni
       FROM Trecho
       WHERE CodEst IN (
          SELECT CodEst
          FROM Estrada
          WHERE NomeEst = 'Rota das Águas'
       )
    )
);
```

```
SELECT DISTINCT u.NomeUF
FROM UF u
JOIN Localidade l ON l.SiglaUF = u.SiglaUF
JOIN Trecho t ON t.CodLocIni = l.CodLoc
JOIN Estrada e ON e.CodEst = t.CodEst
WHERE l.NomeLoc = 'Getulio Vargas'
 AND e.NomeEst = 'Rota das Águas';
```

b. Obter os nomes das localidades em que não há nenhum trecho de estrada começando ou terminando. *Obs: Mostre a operação utilizando EXCEPT e utilizando NOT IN.*

```
SELECT NomeLoc FROM Localidade
WHERE CodLoc NOT IN (
    SELECT CodLocIni FROM  Trecho
    UNION
    SELECT CodLocFim FROM Trecho
)
```

```
SELECT NomeLoc From Localidade
EXCEPT
SELECT NomeLoc FROM Localidade
WHERE CodLoc IN (
    SELECT CodLocIni FROM  Trecho
    UNION
    SELECT CodLocFim FROM Trecho
)
```

c. Uma localidade pode estar contida dentro de outra. Ex. Uma cidade pode conter uma região, uma região pode conter um bairro, etc. Cada unidade de federação é composta de municípios. Um município não está contido em uma localidade mais abrangente (a respectiva coluna denominada CodLocAbrang tem o valor NULL). Deseja-se obter o nome de cada município da UF denominada “Paraná”, seguido do nome de cada localidade contida imediatamente dentro deste município. Caso o município não contenha outras localidades, a segunda coluna deve aparecer vazia (NULL).

```
SELECT m.NomeLoc, r.NomeLoc
FROM Localidade m
JOIN UF uf ON m.SiglaUF = uf.SiglaUF
LEFT JOIN Localidade r ON r.CodLocAbrang = m.CodLoc
WHERE uf.NomeUF = 'Paraná'
  AND m.CodLocAbrang IS NULL;
```

d. Obter uma tabela com três colunas, com o seguinte conteúdo:
   i. Nome da estrada
   ii. Número de trechos da estrada
   iii. Comprimento total da estrada (soma das distâncias de cada trecho)

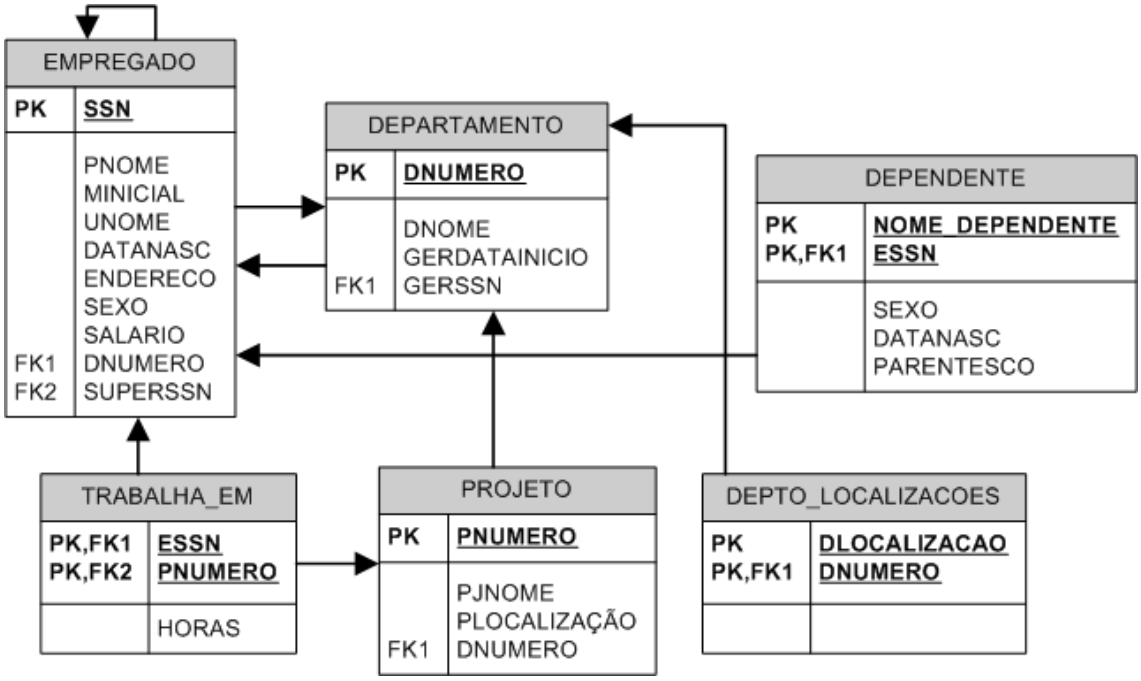
No resultado somente devem aparecer estradas que têm ao menos três trechos.

```
SELECT e.NomeEst, COUNT(t.CodLocIni, SUM(t.Distancia)
FROM Estrada e
JOIN Trecho t ON t.CodEst = e.CodEst
GROUP BY e.CodEst, e.NomeEst
HAVING COUNT(t.CodLocIni) >= 3;
```

e. Obter os nomes das estradas com os trechos mais longos (que contém pelo menos um trecho que é o mais longo de todos os trechos).

```
SELECT DISTINCT e.NomeEst
FROM Estrada e
JOIN Trecho t ON t.CodEst = e.CodEst
WHERE t.Distancia = (
   SELECT MAX(Distancia)
   FROM Trecho
);
```

3. Considere o modelo "empresa" ilustrado abaixo e já visto em sala de aula.



Neste contexto, mostre o código SQL para as seguintes operações:

a. Remova o atributo minicial, supondo que seja importante armazenar apenas o primeiro e último nome do empregado. Em seguida adicione o atributo idade, e altere o atributo sexo para ter como padrão o valor M.

```
ALTER TABLE EMPREGADO
DROP COLUMN MINICIAL;

ALTER TABLE EMPREGADO
ADD IDADE INT;

ALTER TABLE EMPREGADO
ALTER COLUMN SEXO SET DEFAULT 'M';
```

b. Insira um registro no banco referente ao empregado “João M. Souza”, que nasceu no dia 25/08/71, ganha 30.000 e cujo número de seguro social é 123456789.

```
INSERT INTO EMPREGADO (SSN, PNOME, UNOME, DATANASC, SALARIO)
VALUES (123456789, 'João', 'Souza', '1971-08-25', 30000);
```

c. Atualize o registro anterior, pois João teve um aumento de salário de 10%.

```
UPDATE EMPREGADO
SET SALARIO = SALARIO *1.1
WHERE SSN =  123456789;
```

d. Mude o nome de todos os departamentos que começam com C e possuem 4 letras para “Depto Invalido”

```
UPDATE DEPARTAMENTO
SET DNOME = ‘Depto Invalido’
WHERE DNOME LIKE ‘C _ _ _’;
```

e. Remova o registro do empregado cujo número do seguro social é 987654321, pois este empregado foi demitido.

```
DELETE FROM EMPREGADO WHERE SSN =  987654321
```

f. Remova todos os projetos que possuem mais de três pessoas trabalhando.

```
DELETE FROM PROJETO
WHERE PNUMERO IN (
   SELECT PNUMERO
   FROM TRABALHA_EM
   GROUP BY PNUMERO
   HAVING COUNT(ESSN) > 3
);
```

g. Recupere o nome do departamento em que o empregado “João” trabalha.

```
SELECT d.DNOME FROM DEPARTAMENTO d
JOIN EMPREGADO e ON e.DNUMERO = D.DNUMERO
WHERE e.PNOME = ‘João’
```

h. Recupere a data de nascimento e o primeiro nome dos empregados do sexo feminino e que trabalham no departamento “Exatas”.

```
SELECT e.DATANASC, e.PNOME FROM EMPREGADO e
JOIN TRABALHA_EM t ON e.SSN = t.ESSH
WHERE e.SEXO = ‘F’ AND t.NOME = ‘Exatas’;
```

i. Recupere o salário dos empregados que ganham mais de 3000 ou são supervisores.

```
SELECT e.SALARIO FROM EMPREGADO e
  WHERE e.SALARIO > 3000 OR EXISTS(
   SELECT SSH FROM EMPREGADOS
   WHERE SUPERSSN = e.SSH
  );
```

j. Recupere os ssn dos empregados que trabalham no departamento 5, mas que não sejam gerentes deste departamento.

```
SELECT e.SSN FROM EMPREGADO e
WHERE e.DNUMERO = 4 AND NOT EXISTS(
  SELECT 1 FROM DEPARTAMENTO d
  WHERE d.DNUMERO = 4 AND d.GERSSN = e.SSN
);
```

k. Recupere o nome dos empregados que trabalham no departamento que o empregado “Márcio” gerencia, ordenados de forma decrescente pelo último nome.

```
SELECT e.PNOME, e.UNOME FROM EMPREGADO e
JOIN DEPARTAMENTO d ON e.DNUMERO = d.DNUMERO
JOIN EMPREGADO s ON s.SSN = d.GERSSN
WHERE s.PNOME = 'Marcio'
ORDER BY e.UNOME DESC;
```

l. Recupere o primeiro nome dos empregados que ganham o mesmo salário do seu supervisor.

```
SELECT e.PNOME FROM EMPREGADO e
JOIN EMPREGADO s ON s.SSN = e.SUPERSSN
WHERE e.SALARIO = s.SALARIO
```

m. Recupere os nomes dos empregados que ganham mais que algum empregado que trabalha no projeto 3.

```
SELECT e.PNOME FROM EMPREGADO e
WHERE EXISTS(
  SELECT 1 FROM TRABALHA_EM t
  JOIN EMPREGADO e1 ON t.ESSN = e1.SSN
  WHERE t.PNUMERO = 3 AND e1.SALARIO < e.SALARIO
);
```

n. Para cada projeto localizado em cidades iniciadas com “A”, recupere o nome do projeto, o número do departamento responsável e o último nome do gerente do departamento.

```
SELECT p.PNOME, p.DNUMERO, g.UNOME FROM PROJETO p
JOIN DEPARTAMENTO d ON p.DNUMRO = d.DNUMERO
JOIN EMPREGADO g ON g.SSN = d.GERSSN
WHERE p.LOCALIZACAO LIKE 'A%';
```

o. Selecione o salário e último nome dos empregados que ganham mais do que todos os gerentes.

```
SELECT SALARIO, UNOME FROM EMPREGADO
WHERE SALARIO > (
   SELECT MAX(SALARIO) FROM EMPREGADO g
   JOIN DEPARTAMENTO d ON g.SSN = d.GERSSN
);
```

p. Utilizando a cláusula **JOIN,** recupere o nome e o endereço de todos os empregados que trabalham no departamento “Solos”.

```
SELECT E.PNOME, E.UNOME, E.ENDERECO FROM EMPREGADO E
JOIN DEPARTAMENTO D ON E.DNUMERO = D.DNUMERO
WHERE D.DNOME = 'Solos';
```

q. Faça uma lista de todos os números de projetos nos quais esteja envolvido algum empregado cujo nome seja “Carlos”; **ou** como empregado, **ou** como gerente do departamento que controle o projeto.

```
SELECT P.PNUMERO FROM PROJETO P
WHERE EXISTS(
   SELECT 1 FROM TRABALHA_EM T
   JOIN EMPREGADO E ON E.SSN = T.ESSN
   WHERE E.PNOME = 'Carlos' AND T.PNUMERO = P.PNUMERO
)OR EXISTS(
   SELECT 1 FROM EMPREGADO G
   JOIN DEPARTAMENTO D ON D.GERSSN = G.SSN
   WHERE G.PNOME = 'Carlos' AND P.DNUMERO = D.DNUMERO
)
```

r. Recupere o nome dos empregados e respectivos projetos nos quais eles trabalham, incluindo os empregados que ainda não estejam trabalhando em projeto algum, em ordem alfabética de empregado.

```
SELECT E.PNOME, E.UNOME, P.PNOME AS PROJETO
FROM EMPREGADO E
LEFT JOIN TRABALHA_EM T ON E.SSN = T.ESSN
LEFT JOIN PROJETO P ON T.PNUMERO = P.PNUMERO
ORDER BY E.PNOME ASC, E.UNOME ASC;
```

s. Recupere o nome de todos os empregados que não tem supervisor.

```
SELECT PNOME, UNOME FROM EMPREGADO
WHERE SUPERSSN IS NULL;
```

t. Recupere os dependentes dos empregados que gerenciam algum departamento.

```
SELECT D.NOME_DEPENDENTE FROM DEPENDENTE D
JOIN DEPARTAMENTO DR ON DR.GERSSN = D.ESSN;
```

u. Recupere os nomes dos departamentos que controlam todos os projetos localizados no Pavilhão 2.

```
SELECT D.DNOME FROM DEPARTAMENTO D
WHERE NOT EXISTS(
    SELECT P.PNUMERO FROM PROJETO P
    WHERE P.PLOCALIZACAO = 'Pavilhao 2'
    EXCEPT
    SELECT P.PNUMERO FROM PROJETO P
    WHERE P.DNUMERO = D.DNUMERO
      AND P.PLOCALIZACAO = 'Pavilhao 2'
);
```

v. Encontre o maior salário e a média salarial dos empregados do departamento “Exatas”.

```
SELECT MAX(E.SALARIO), AVG(E.SALARIO)
FROM EMPREGADO E
JOIN DEPARTAMENTO D ON D.DNUMERO = E.DNUMERO
WHERE D.DNOME = 'Exatas'
```

w. Recupere o número total de empregados do departamento “Pesquisa”.

```
SELECT COUNT(*) FROM EMPREGADO E
JOIN DEPARTAMENTO D ON D.DNUMERO = E.DNUMERO
WHERE D.DNOME = 'Pesquisa’
```

x. Para cada departamento, recupere seu número e média de salários de seus empregados.

```
SELECT D.DNUMERO, AVG(E.SALARIO)
FROM DEPARTAMENTO D
JOIN EMPREGADO E ON D.DNUMERO = E.DNUMERO
GROUP BY D.DNUMERO
```

y. Para cada projeto, recupere o total de horas trabalhadas pelos seus empregados, desde que tais empregados sejam do departamento 2.

```
SELECT T.PNUMERO, SUM(T.HORAS) FROM TRABALHA_EM T
JOIN EMPREGADO E ON E.SSN = T.ESSN
JOIN DEPARTAMENTO D ON D.DNUMERO = E.DNUMERO
WHERE D.DNUMERO = 2
GROUP BY T.PNUMERO
```

z. Para cada projeto controlado pelo departamento 3, nos quais trabalhem mais de dois empregados, recupere o nome do projeto.

```
SELECT P.PNOME
FROM PROJETO P
JOIN DEPARTAMENTO D ON P.DNUM = D.DNUMERO
JOIN TRABALHA_EM T ON T.PNUMERO = P.PNUMERO
WHERE D.DNUMERO = 3
GROUP BY P.PNUMERO P.PNOME
HAVING COUNT(*) > 2;
```