Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

**Curso de Tecnologia em Sistemas de Computação**
**Disciplina Banco de Dados**
**AD2 2° semestre de 2006.**

**Nome –**

---

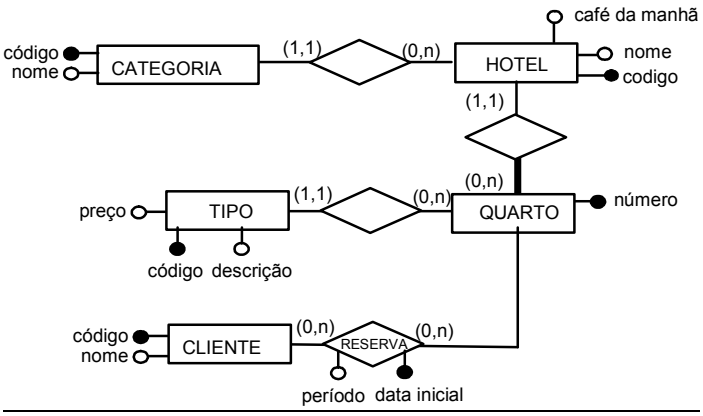
Observações:

1. Prova com consulta.
2. Não aceitaremos provas feitas à mão.

---

Questão 1

Considere o diagrama ER mostrado abaixo. Construa um esquema relacional equivalente a este diagrama ER.




Categoria (CodCategoria, Nome)
Hotel (CodHotel, Nome, Café, CodCategoria)
CodCategoria referencia Categoria
Quarto (CodHotel, NumQuarto, CodTipo)
CodHotel referencia Hotel
CodTipo referencia Tipo
Tipo (CodTipo, Preço, Descrição)
Cliente (CodCliente, Nome)

Reserva (CodHotel, Numero, CodCliente, Período, DataIni)
(CodHotel, NumQuarto) referencia Quarto
CodCliente referencia Cliente

Questão 2

Responda cada uma das questões seguintes. As questões são baseadas no seguinte esquema relacional:

| |
|---|
| Emp (*eid*: integer, *ename*: string, *idade*: integer, *salario*: real) |
| Trabalha (*eid*: integer, *did*: integer, *cargahoraria*: integer) |
| Dept (*did*: integer, *dnome*: string, *orçamento*: real, *gerenteid*: integer) |

No esquema acima, as chaves primárias estão sublinhadas.

(a) Dê um exemplo de uma chave estrangeira que envolva (referencie) a tabela Dept. Quais são as opções para garantir esta restrição quando um usuário tenta excluir uma tupla de departamento?

A tabela *Trabalha* poderia referenciar a tabela *Dept* se *did* tivesse sido definido como chave estrangeira em *Trabalha*. Esta definição poderia ser feita da seguinte forma:

```
CREATE TABLE TRABALHA (
        EID INT NOT NULL,
        DID INT,
        CARGAHORARIA INT,
        PRIMARY KEY (EID),
        FOREIGN KEY (DID) REFERENCES DEPT (DID)
        )
```

As opções para garantir a restrição quando o usuário tenta excluir uma tupla de departamento são as seguintes:

(a) Especificar o comando CREATE TABLE como acima. Neste caso, o SGBD não deixa que uma tupla de departmento seja excluída quando houver uma ou mais tuplas relacionadas em *Trabalha*. O comando de criação acima utiliza a definição *default* de declaração de restrição de integridade, que é ON DELETE RESTRICT.

(b) Especificar o comando CREATE TABLE acima usando ON DELETE CASCADE na definição da chave estrangeira.

```
CREATE TABLE TRABALHA (
        EID INT NOT NULL,
        DID INT,
```

```
    CARGAHORARIA INT,
    PRIMARY KEY (EID),
    FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE CASCADE
    )
```

Neste caso, ao excluir uma tupla de departamento, todas as tuplas relacionadas da tabela *Trabalha* também serão excluídas.

(c) Especificar o comando CREATE TABLE usando ON DELETE SET NULL na definição da chave estrangeira.

```
CREATE TABLE TRABALHA (
    EID INT NOT NULL,
    DID INT,
    CARGAHORARIA INT,
    PRIMARY KEY (EID),
    FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE SET NULL
    )
```

Neste caso, ao excluir uma tupla de departamento, todas as tuplas relacionadas da tabela *Trabalha* terão o conteúdo da coluna DID alterados para NULL.

(b) Escreva os comandos SQL necessários para criar as relações acima, incluindo as versões apropriadas de todas as restrições de chave primária e chave estrangeira. Considere que a tabela Trabalha relaciona empregados com departamentos e a tabela Dept se relaciona com Emp a partir do atributo *gerenteid*.

```
CREATE TABLE EMP (
    EID INT NOT NULL,
    ENAME VARCHAR(30),
    IDADE INT,
    SALARIO FLOAT,
    PRIMARY KEY (EID)
    )

CREATE TABLE DEPT (
    DID INT NOT NULL,
    DNOME VARCHAR(30),
    ORCAMENTO FLOAT,
    GERENTEID INT,
    PRIMARY KEY (DID),
    FOREIGN KEY (GERENTEID) REFERENCES EMP (EID) ON DELETE SET
NULL ON UPDATE CASCADE
    )
```

A chave estrangeira acima também poderia ter sido definida usando ON DELETE SET RESTRICT.

```
CREATE TABLE TRABALHA (
        EID INT NOT NULL,
        DID INT,
        CARGAHORARIA INT,
        PRIMARY KEY (EID),
        FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE RESTRICT
ON UPDATE CASCADE,
        FOREIGN KEY (EID) REFERENCES EMPREGADO (EID) ON DELETE
RESTRICT ON UPDATE CASCADE
        )
```

A chave estrangeira acima também poderia ter sido definida usando ON DELETE SET NULL ou ON DELETE CASCADE.

(c) Defina a tabela Dept em SQL de modo que cada departamento tenha necessariamente um gerente.

```
CREATE TABLE DEPT (
        DID INT NOT NULL,
        DNOME VARCHAR(30),
        ORCAMENTO FLOAT,
        GERENTEID INT NOT NULL,
        PRIMARY KEY (DID),
        FOREIGN KEY (GERENTEID) REFERENCES EMP (EID) ON DELETE
RESTRICT ON UPDATE CASCADE)
        )
```

(d) Escreva uma instrução SQL para adicionar "João da Silva" como um empregado com eid = 101, idade = 32 e salário = 15000.

```
INSERT INTO EMP (EID, ENAME, IDADE, SALARIO) VALUES (101, "João da
Silva", 32, 15000)
```

ou

```
INSERT INTO EMP VALUES (101, "João da Silva", 32, 15000)
```

(e) Escreva uma instrução SQL para dar a todos os empregados um aumento de 10%.

```
UPDATE EMP
SET SALARIO = SALARIO * 1.1
```

(f) Escreva uma instrução SQL que exclua o departamento de nome “Brinquedo”. Dadas as restrições de integridade referencial que você definiu para o esquema, explique o que acontece quando esta instrução é executada.

```
DELETE FROM DEPT
WHERE DID IN (SELECT DID FROM DEPT WHERE DNOME=”Brinquedo”)
```

A resposta da segunda parte desta questão depende de como o aluno especificou a chave estrangeira da tabela DEPT na questão 2(b). As respostas para cada opção estão abaixo:

- FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE RESTRICT

Neste caso, ao excluir o departamento brinquedo, se houver alguma tupla em *Trabalha* que referencia este departamento, a exclusão será negada pelo SGBD.

- FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE SET NULL

Neste caso, ao excluir o departamento brinquedo, se houver alguma tupla em *Trabalha* que referencia este departamento, a coluna DID desta(s) tupla(s) da tabela *Trabalha* será(ão) alteradas para NULL.

- FOREIGN KEY (DID) REFERENCES DEPT (DID) ON DELETE CASCADE

Neste caso, ao excluir o departamento brinquedo, se houver alguma tupla em *Trabalha* que referencia este departamento, esta(s) tupla(s) será(ão) excluída(s) da tabela *Trabalha*.

(g) Escreva uma instrução SQL que altere a carga horária de todos os empregados que trabalham no departamento “Vendas” para 20.

```
UPDATE TRABALHA
SET CARGAHORARIA=20
WHERE DID IN (SELECT DID FROM DEPT WHERE DNOME=”Vendas”)
```

(h) Escreva uma consulta SQL que retorne a média salarial dos empregados, por departamento.

```
SELECT DID, AVG(SALARIO)
FROM EMP e, TRABALHA t
WHERE e.EID=t.EID
GROUP BY DID
```

(i) Escreva uma consulta SQL que retorna o nome do departamento que possui o maior orçamento.

```
SELECT DNOME
FROM DEPT
WHERE ORCAMENTO=(SELECT MAX(ORCAMENTO) FROM DEPT)
```

(j) Escreva uma consulta que retorna os nomes dos empregados que trabalham no departamento que possui o maior orçamento.

```
SELECT ENAME
FROM EMP e, TRABALHA t, DEPT d
WHERE t.EID=e.EID
AND d.DID=t.DID
AND d.ORCAMENTO=(SELECT MAX(ORCAMENTO) FROM DEPT)
```

Questão 3

Defina as duas alternativas para implementação de visões vistas em aula (reescrita de consulta e materialização de visões). Discuta os prós e contras de cada uma das duas soluções.

Materialização de visões: nesta alternativa, o SGBD cria uma tabela temporária e insere nela todas as tuplas que fazem parte da visão, "materializando-as".

Reescrita de consulta: nesta alternativa, o SGBD não materializa a visão. Quando chega alguma consulta à visão, o SGBD compõem a consulta de definição da visão com a consulta do usuário, e executa esta nova consulta composta.

Prós e contras:

| | **PRÓ** | **CONTRA** |
|---|---|---|
| Materialização de visões | As consultas do usuário sobre a visão podem ser executadas de forma eficiente, uma vez que não é necessário nenhum pré-processamento da consulta para fazer a composição das consultas do usuário com a de definição da visão. | O SGBD tem que manter a visão materializada atualizada sempre que as tabelas base forem alteradas. |
| | Consultas sobre visões que envolvem operações caras, como junções, não precisam ser executadas, já que todas as tuplas da junção já estão pré-calculadas na visão materializada. | |
| Composição de consultas | O SGBD não precisa monitorar se as tabelas base sofreram modificações, pois nesta abordagem a visão não é materializada. | Consultas sobre visões que envolvem operações caras têm que ser realizadas a cada vez que a visão é consultada. |

Questão 4

Considere o seguinte esquema relacional:

| |
|---|
| Fornecedores(*fid:* integer, *fnome:* string, *end:* string) |
| Peças(*pid:* integer, *pnome:* string, *cor:* string) |
| Catálogo(*fid:* integer, *pid:* integer, *preço:* real) |

No esquema acima, as chaves primárias estão sublinhadas. A tabela de Catálogo lista o preço praticado pelos fornecedores para cada peça fornecida.
Apresente, para cada consulta a seguir, as expressões em álgebra relacional e as instruções SQL correspondente às consultas.

1. Obtenha o nome dos fornecedores que fornecem alguma peça vermelha.

$\pi_{fnome}(\pi_{fid}((\pi_{pid}\,\sigma_{cor="vermelha"}\,Peças) \bowtie Catalogo) \bowtie Fornecedores)$

SQL

```
SELECT F.fnome
FROM Fornecedores F, Peças P, Catalogo C
WHERE P.cor='vermelha' AND C.pid=P.pid AND C.fid=F.fid
```

2. Obtenha o fid dos fornecedores que fornecem alguma peça vermelha ou verde.

$\pi_{fid}(\pi_{pid}(\sigma_{cor="vermelha" \vee cor = "verde"}\,Peças) \bowtie Catalogo)$

SQL

```
SELECT C.fid
FROM Catalogo C, Peças P
WHERE (P.cor = 'vermelha' OR P.cor = 'verde')
AND P.pid = C.pid
```

3. Obtenha o fid dos fornecedores que fornecem alguma peça vermelha ou que estão no endereço “Rua sobe e desce”.

$\rho(R1, \pi_{fid}((\pi_{pid}\sigma_{cor=”vermelha”} Peças) \bowtie Catalogo))$
$\rho(R2, \pi_{fid}\sigma_{end=”Rua\ Sobe\ e\ Desce”} Fornecedores)$
$R1 \cup R2$

```
SQL
SELECT F.fid
FROM Fornecedores F
WHERE F.end = ‘Rua Sobe e Desce’
      OR F.fid IN ( SELECT C.fid
                    FROM Peças P, Catalogo C
                     WHERE P.cor=’vermelha’ AND P.pid = C.pid )
```

4. Obtenha o fid dos fornecedores que fornecem alguma peça vermelha e alguma peça verde.

$\rho(R1, \pi_{fid}((\pi_{pid}\sigma_{cor='vermelha'} Peças) \bowtie Catalogo))$
$\rho(R2, \pi_{fid}((\pi_{pid}\sigma_{cor=‘verde’} Peças) \bowtie Catalogo))$
$R1 \cap R2$

```
SQL
SELECT C.fid
FROM Peças P, Catalogo C
WHERE P.cor = ‘vermelha’ AND P.pid = C.pid
AND EXISTS ( SELECT P2.pid
FROM Peças P2, Catalog C2
WHERE P2.color = ‘verde’ AND C2.fid = C.fid
AND P2.pid = C2.pid )
```

5. Obtenha o nome dos fornecedores que fornecem alguma peça vermelha que tenha preço abaixo de 100.

$\pi_{fnome}(\pi_{fid}((\pi_{pid}\ \sigma_{cor=”vermelha”} Peças) \bowtie \sigma_{preço<100} Catalogo) \bowtie Fornecedores)$

```
SQL
SELECT F.fnome
FROM Fornecedores F, Peças P, Catalogo C
WHERE P.cor=’vermelha’
    AND preço < 100
    AND C.pid=P.pid AND C.fid=F.fid
```

6. Obtenha o nome das peças que possuem algum fornecedor.

$\pi_{pnome}$ (Peça ⋈ Catalogo)

SQL

```
SELECT DISTINCT P.pnome
FROM Peças P, Catalogo C
WHERE C.pid=P.pid
```

7. Obtenha o nome das peças que são fornecidas pelo fornecedor "Casas Populares" e não são fornecidas por nenhum outro fornecedor.

$\rho(R1, \pi_{pid, pnome}$ (Peças ⋈ Catalogo ⋈ ($\sigma_{fnome = 'Casas\ Populares'}$ Fornecedores)))
$\rho(R2, \pi_{pid, pnome}$ (Peças ⋈ Catalogo ⋈ ($\sigma_{fnome <> 'Casas\ Populares'}$ Fornecedores)))
R1 - R2

SQL

```
SELECT P.pnome
FROM Fornecedores F, Peças P, Catalogo C
WHERE F.fnome = 'Casas Populares'
AND      P.pid = C.pid AND C.fid=F.fid
AND      NOT EXISTS ( SELECT *
                      FROM Catalogo C2, Fornecedores F2
                      WHERE F2.fnome <> 'Casas Populares'
                      AND      C2.fid = F2.fid
                      AND      P.pid = C2.pid )
```

8. Para cada peça, obtenha o pid e o nome do fornecedor que cobra o preço mais alto para essa peça.

**Álgebra relacional: ANULADA**

SQL

```
SELECT P.pid, F.fnome
FROM Fornecedores F, Peças P, Catalogo C
WHERE P.pid = C.pid AND C.fid=F.fid
AND     C.preço =  ( SELECT MAX (C1.preço)
                     FROM Catalogo C1
                     WHERE P.pid = C1.pid )
```