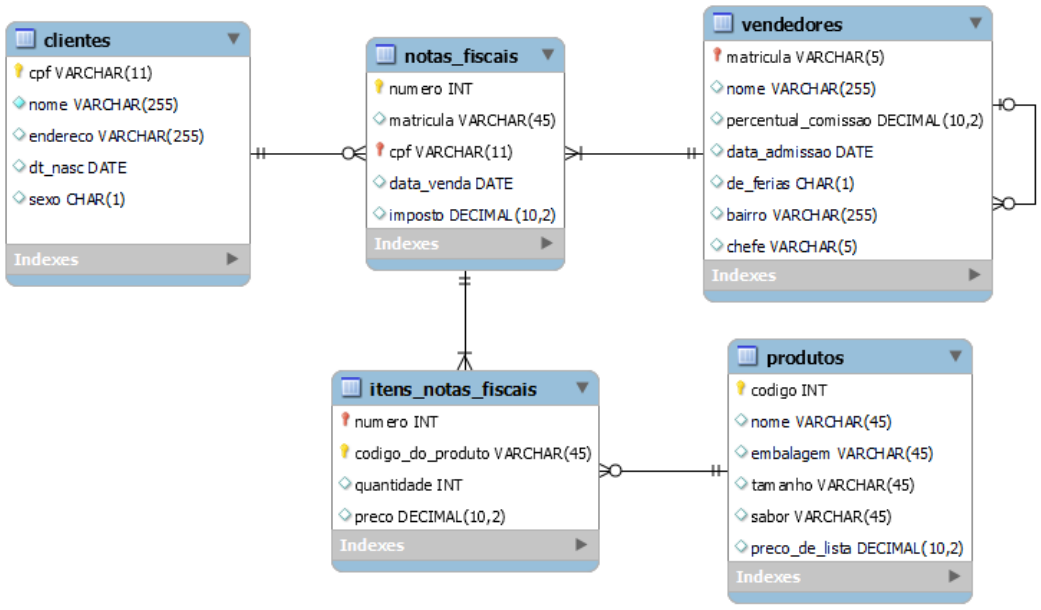
QUESTÃO DISCURSIVA

Considere o modelo para responder aos itens abaixo.



Observações:

- Na tabela "**notas_fiscais**", as colunas "*cpf*" e "*matricula*" são chaves estrangeiras e referenciam, respectivamente, as tabelas "**clientes"** (*cpf*) e "**vendedores"** (*matricula*);
- Na tabela "**vendedores**", a coluna "*chefe*" é chave estrangeira e referencia a própria tabela "**vendedores**" (*matricula*);
- Na tabela "**itens_notas_fiscais**", as colunas "*numero*" e "*codigo_do_produto*" são a chave primária composta da tabela e referenciam, respectivamente, as tabelas "**notas_fiscais**" (*numero*) e "**produtos**" (*codigo*).
- Na tabela "**itens_notas_fiscais**", a coluna "*preco*" resulta da multiplicação da coluna "*quantidade*" pela coluna "preco_de_lista" da tabela "**produtos"**.

1. Escreva uma consulta SQL que retorne o nome de todos os vendedores e dos respectivos chefes (se houver);
2. Por um erro na aplicação que é responsável por popular as tabelas acima, a coluna "*preço*" da tabela "**itens_notas_fiscais**" não foi computada, ou seja, todos os valores da coluna estão como "*null*". Felizmente a equipe de programadores já realizou a correção na aplicação, porém os registros já inseridos ainda estão com valor "*null*" para essa coluna. Considerando as observações acima, escreva uma instrução SQL que popule essa coluna corretamente para todos os registros já existentes;
3. Utilizando apenas o operador "EXISTS" realizar associação entre tabelas (proibido o uso de junções e subconsultas não relacionadas), escreva uma consulta SQL que retorne todos os produtos que não tenham sido comprados pelo cliente de nome "*Leonardo Chapisco*", considere que só há um cliente com tal nome;
4. Escreva uma consulta SQL que retorne o nome dos clientes e o valor total gasto por esses clientes (ex: "Leonardo Chapisco", "400.0") considerando todas as compras já realizadas. Além disso, a consulta só deve mostrar os clientes que compraram mais de R$ 100,00 (cem) reais em produtos.

SUGESTÕES DE SOLUÇÕES:

1. Escreva uma consulta SQL que retorne o nome de todos os vendedores e dos respectivos chefes (se houver);

```
SELECT v1.nome AS vendedor, v2.nome AS chefe
    FROM vendedores v1
    LEFT OUTER JOIN vendedores v2
    ON v1.chefe = v2.matricula;
```

Percebe-se pelo modelo que há um autorrelacionamento da tabela "Vendedores", e que o campo "chefe" nada mais é do que uma chave estrangeira que referencia a chave primária "matricula".

Também é possível inferir que estamos diante de um relacionamento opcional de ambos os lados do relacionamento (bolinha branca/vazada), dessa forma pode existir vendedor sem chefe, bem como chefe sem chefiar ninguém.

Em outras palavras, devemos retornar uma consulta que liste todos os vendedores (LEFT JOIN) e mostre os respectivos chefes (quando houver). Para tanto, utilizamos na condição de junção o atributo "chefe" da tabela vendedor (lado esquerdo) e, no lado direito, o campo "matricula";

2. Por um erro na aplicação que é responsável por popular as tabelas acima, a coluna "*preço*" da tabela "**itens_notas_fiscais**" não foi computada, ou seja, todos os valores da coluna estão como "*null*". Felizmente a equipe de programadores já realizou a correção na aplicação, porém os registros já inseridos ainda estão com valor "*null*" para essa coluna. Considerando as observações acima, escreva uma instrução SQL que popule essa coluna corretamente para todos os registros já existentes;

```
UPDATE itens_notas_fiscais inf
   SET inf.preco = cast(inf.quantidade AS DECIMAL(10,2)) *
     (SELECT p.preco_de_lista FROM produtos p WHERE p.codigo =
inf.codigo_do_produto);

COMMIT; -- execute o commit para manter os dados atualizados, caso
contrário o SQL Fiddle não mostrará os dados atualizados em
requisições subsequentes;
```

No update acima, o Oracle não faz conversão de tipo implicitamente, sendo necessário o uso da função "cast()", caso tenha esquecido desse detalhe, considere a solução como correta.

A instrução "commit" não faz parte da solução, adicionei ela ao código para que garantisse a persistência dos preços alterados para requisições subsequentes, pois a solução do item 4 considera que a coluna preço da tabela "**itens_notas_fiscais**" não estejam nulas.

3. Utilizando apenas o operador "EXISTS" realizar associação entre tabelas (proibido o uso de junções e subconsultas não relacionadas), escreva uma consulta SQL que retorne todos os produtos que não tenham sido comprados pelo cliente de nome "*Leonardo Chapisco*", considere que só há um cliente com tal nome.

```
SELECT p.nome FROM produtos p
WHERE NOT EXISTS
    (SELECT 1 FROM itens_notas_fiscais inf
    WHERE EXISTS
        (SELECT 1 FROM notas_fiscais nf
        WHERE EXISTS
            (SELECT 1 FROM clientes c
                WHERE p.codigo = inf.codigo_do_produto
                AND inf.numero = nf.numero
                AND nf.cpf = c.cpf
                AND c.nome = 'Leonardo Chapisco'
            )
        )
    )
```

O entendimento de consultas com a cláusula EXISTS pode ser melhor explicado pelos vídeos abaixo:

Exists e Not Exists (parte 1)

Exists e Not Exists (parte 2)

4. Escreva uma consulta SQL que retorne o nome dos clientes e o valor total gasto por esses clientes (ex: “Leonardo Chapisco”, “400.0”) considerando todas as compras já realizadas. Além disso, a consulta só deve mostrar os clientes que compraram mais de R$ 100,00 (cem) reais em produtos.

```sql
SELECT c.nome, sum(inf.preco) AS total FROM itens_notas_fiscais inf
    INNER JOIN notas_fiscais nf
        ON inf.numero = nf.numero
    INNER JOIN clientes c
        ON nf.cpf = c.cpf
    GROUP BY c.nome
    HAVING sum(inf.preco) > 100
    ORDER BY total;
```

- Todas as soluções podem ser devidamente testadas no site SQL Fiddle utilizando o Oracle 11g.
- Os códigos de DDL e DML estão no repositório https://github.com/pierryangelo/sql_chapisco