**Contexto**

A empresa está avaliando alternativas de uso ao MySQL. Uma delas é a utilização de bancos NoSQL, com destaque para o MongoDB, que é bastante recomendado. Você tem a tarefa, nesse contexto, de utilizar, incluir e manipular, no MongoDB, algumas das informações presentes no banco de dados de lojas.

**Atividade**

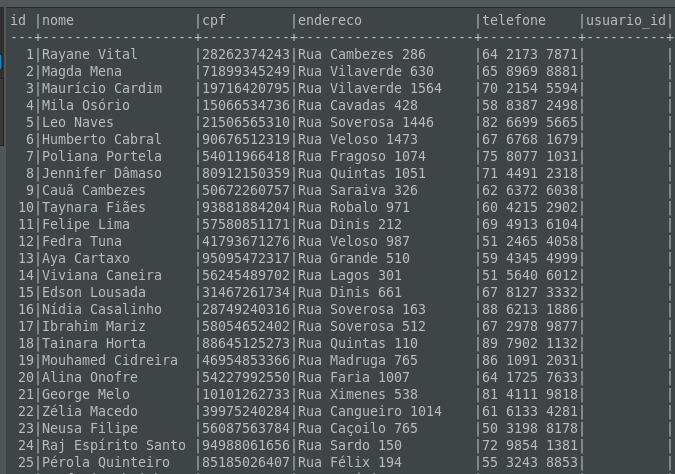
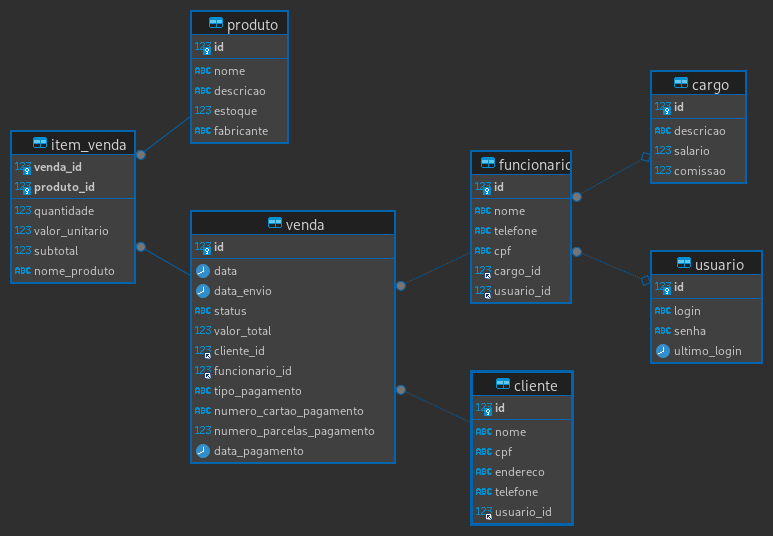
Para este momento, é requisitado que você trabalhe apenas com a tabela **Produto** do banco de dados **uc4atividade**, “traduzindo-a” para NoSQL.

Utilizando preferencialmente o MongoDB Compass, escreva e guarde em arquivo comandos para:

1. Criação de um novo banco NoSQL
2. Inclusão dos dados de nome, descricao, estoque, fabricante para a tabela “Produto”
3. Atualização de dados de produto (qualquer campo de sua preferência)
4. Busca de produto por nome
5. Busca de produto por fabricante

Para cada comando, escreva também no arquivo os resultados obtidos na execução.

#### **Atividade 04**



#### **Primeira forma normal (1FN)**

Nenhuma tabela pode conter atributos multivalorados. Para normalizar a tabela na 1FN, devemos garantir que cada atributo contenha apenas um valor.

Para isso devemos atomizar os atributos da tabela. Como é sabido, uma pessoa pode ter mais de um telefone e/ou endereço.

Para garantir a 1FN podemos criar uma tabela de telefones e associar ao cliente por meio de uma chave estrangeira.

CREATE TABLE telefone\_cliente (

id INT AUTO\_INCREMENT PRIMARY KEY,

cliente\_id INT,

numero\_telefone VARCHAR(20),

FOREIGN KEY (cliente\_id) REFERENCES cliente(id)

);

CREATE TABLE endereco\_cliente (

id INT AUTO\_INCREMENT PRIMARY KEY,

cliente\_id INT,

logradouro VARCHAR(50),

bairro VARCHAR(50),

cidade VARCHAR(50),

cep VARCHAR(50),

FOREIGN KEY (cliente\_id) REFERENCES cliente(id)

);

`usuario\_id` INT NULL DEFAULT NULL,

#### **2)Segunda forma normal (2FN)**

Segundo nosso material de estudo:

“Deve estar de acordo com a 1FN e que todos os atributos não chave sejam dependentes da chave primária. Ao utilizar a regra 2FN, você estará evitando valores redundantes em sua base de dados, o que poderá causar uma baixa performance e o uso excessivo de recursos do banco de dados.”

A tabela "Item\_venda" não está na segunda forma normal (2FN) por possuir dependências parciais em sua chave primária. Isso ocorre quando uma coluna que faz parte da chave primária depende apenas de uma parte da chave primária composta, e não da chave completa.

Criar uma tabela separada para armazenar os detalhes de cada item vendido, incluindo o preço unitário e a quantidade. Em seguida, usar um identificador único para cada item vendido e associá-lo à tabela de vendas por meio de uma chave estrangeira

CREATE TABLE item\_vendido (

id INT AUTO\_INCREMENT PRIMARY KEY,

venda\_id INT,

produto\_id INT,

quantidade INT,

preco\_unitario DECIMAL(10,2),

FOREIGN KEY (venda\_id) REFERENCES venda(id),

FOREIGN KEY (produto\_id) REFERENCES produto(id)

);

#### **3)Terceira forma normal (3FN)**

A tabela "Venda" não está na terceira forma normal (3FN) porque pode conter dependências transitivas. Por exemplo, se a tabela de vendas armazenar tanto o preço unitário quanto o preço total de um item vendido, o preço total será transitivamente dependente do preço unitário e, portanto, viola a 3FN.

Remover o preço total da tabela de vendas e calcular o preço total com base no preço unitário e na quantidade de cada item vendido. Isso elimina a dependência transitiva e normaliza a tabela na 3FN.

CREATE TABLE venda (

id INT AUTO\_INCREMENT PRIMARY KEY,

cliente\_id INT,

funcionario\_id INT,

data\_venda DATE,

tipo\_pagamento VARCHAR(50),

FOREIGN KEY (cliente\_id) REFERENCES cliente(id),

FOREIGN KEY (funcionario\_id) REFERENCES funcionario(id)

);

CREATE TABLE item\_venda (

id INT AUTO\_INCREMENT PRIMARY KEY,

venda\_id INT,

produto\_id INT,

quantidade INT,

preco\_unitario DECIMAL(10,2),

FOREIGN KEY (venda\_id) REFERENCES venda(id),

FOREIGN KEY (produto\_id) REFERENCES produto(id)

);

ALTER TABLE item\_venda ADD COLUMN preco\_total DECIMAL(10,2);

