**BANCO DE DADOS – PROFº HOWARD CRUZ ROATTI**

**LÓGICA DO TRABALHO – PARTE 1 (SQL)**

**Aplicação a ser feita:** Sistema de Controle de Estoques (Custo Médio Ponderado)

**Grupo:** Gabriely Azevedo, Guilherme Gonçalves, Luiz Henrique, Ricardo da Silva, Rodrigo Araujo.

**CONTROLE DE ESTOQUES (CMP)**

Existem 3 formas de controlar o Estoque, que são PEPS, UEPS e CMP:

* **Primeiro que Entra, Primeiro que Sai (PEPS):** sempre que a loja vende um produto, entrega-se o produto mais antigo no Estoque.
* **Último que Entra, Primeiro que Sai (UEPS):** sempre que a loja vende um produto, entrega-se o produto mais recente, de forma que os mais antigos permanecem no Estoque.
* **Custo Médio Ponderado (CMP):** o documento em si não faz distinção de quais produtos no Estoque são mais recentes ou mais antigos. Sempre que ocorre um ressuprimento de um produto, mesclam-se as quantidades e valor total, com a verdadeira alteração ocorrendo no Custo Unitário.

O formato base dos Controles de Estoque é:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATA** | **PROD** | **ENTRADAS** | | | **SAÍDAS** | | | **SALDO** | | |
| C. Un. | Qtd. | Total | C. Un. | Qtd. | Total | C. Un. | Qtd. | Total |
|  |  |  |  |  |  |  |  |  |  |  |

Considerando os seguintes Fatos Contábeis, veremos como ficariam as demonstrações PEPS, UEPS e CMP:

* **FC1:** loja comprou, no dia 01/01, 10 TVs por R$ 4.000,00/unidade, 5 PCs por R$ 8.000,00/unidade e 8 ventiladores por R$ 500,00/unidade
* **FC2:** loja vende, no dia 05/01, 3 TVs, 2 PCs e 5 ventiladores
* **FC3:** loja vende, no dia 10/01, 5 TVs e 1PC. Neste mesmo dia, loja compra mais 3 PCs por R$ 9.000,00/unidade
* **FC4:** loja vende, no dia 20/01, 3 PCs
* **OBS:** no Controle de Estoque não se verifica Lucro, de forma que os valores serão sempre os ocorridos pela compra da loja

**PEPS**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATA** | **PROD** | **ENTRADAS** | | | **SAÍDAS** | | | **SALDO** | | |
| C. Un. | Qtd. | Total | C. Un. | Qtd. | Total | C. Un. | Qtd. | Total |
| 01/01 | TV | 4.000 | 10 | 40.000 | - | - | - | 4.000 | 10 | 40.000 |
| PC | 8.000 | 5 | 40.000 | - | - | - | 8.000 | 5 | 40.000 |
| VENT | 500 | 8 | 4.000 | - | - | - | 500 | 8 | 4.000 |
| 05/01 | TV | - | - | - | 4.000 | 3 | 12.000 | 4.000 | 7 | 28.000 |
| PC | - | - | - | 8.000 | 2 | 16.000 | 8.000 | 3 | 24.000 |
| VENT | - | - | - | 500 | 5 | 2.500 | 500 | 3 | 1.500 |
| 10/01 | TV | - | - | - | 4.000 | 5 | 20.000 | 4.000 | 2 | 8.000 |
| PC | - | - | - | 8.000 | 1 | 8.000 | 8.000 | 2 | 16.000 |
| PC | 9.000 | 3 | 27.000 | - | - | - | 9.000 | 3 | 27.000 |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| 20/01 | TV | - | - | - | - | - | - | 4.000 | 2 | 8.000 |
| PC | - | - | - | 8.000 | 2 | 16.000 | - | - | - |
| PC | - | - | - | 9.000 | 1 | 9.000 | 9.000 | 2 | 18.000 |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| **TOTAL** |  |  |  | **111.000** |  |  | **83.500** |  |  | **27.500** |

\* No dia 20, por ser PEPS, são vendidas as 2 unidades restantes do PC de R$ 8.000,00 antes, que são mais antigos

**UEPS**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATA** | **PROD** | **ENTRADAS** | | | **SAÍDAS** | | | **SALDO** | | |
| C. Un. | Qtd. | Total | C. Un. | Qtd. | Total | C. Un. | Qtd. | Total |
| 01/01 | TV | 4.000 | 10 | 40.000 | - | - | - | 4.000 | 10 | 40.000 |
| PC | 8.000 | 5 | 40.000 | - | - | - | 8.000 | 5 | 40.000 |
| VENT | 500 | 8 | 4.000 | - | - | - | 500 | 8 | 4.000 |
| 05/01 | TV | - | - | - | 4.000 | 3 | 12.000 | 4.000 | 7 | 28.000 |
| PC | - | - | - | 8.000 | 2 | 16.000 | 8.000 | 3 | 24.000 |
| VENT | - | - | - | 500 | 5 | 2.500 | 500 | 3 | 1.500 |
| 10/01 | TV | - | - | - | 4.000 | 5 | 20.000 | 4.000 | 2 | 8.000 |
| PC | - | - | - | 8.000 | 1 | 8.000 | 8.000 | 2 | 16.000 |
| PC | 9.000 | 3 | 27.000 | - | - | - | 9.000 | 3 | 27.000 |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| 20/01 | TV | - | - | - | - | - | - | 4.000 | 2 | 8.000 |
| PC | - | - | - | - | - | - | 8.000 | 2 | 16.000 |
| PC | - | - | - | 9.000 | 3 | 27.000 | - | - | - |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| **TOTAL** |  |  |  | **111.000** |  |  | **85.500** |  |  | **25.500** |

\* No dia 20, por ser UEPS, são vendidas as 3 unidades de PC mais recentes, pois foram os últimos PCs a entrarem no estoque

**CMP**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DATA** | **PROD** | **ENTRADAS** | | | **SAÍDAS** | | | **SALDO** | | |
| C. Un. | Qtd. | Total | C. Un. | Qtd. | Total | C. Un. | Qtd. | Total |
| 01/01 | TV | 4.000 | 10 | 40.000 | - | - | - | 4.000 | 10 | 40.000 |
| PC | 8.000 | 5 | 40.000 | - | - | - | 8.000 | 5 | 40.000 |
| VENT | 500 | 8 | 4.000 | - | - | - | 500 | 8 | 4.000 |
| 05/01 | TV | - | - | - | 4.000 | 3 | 12.000 | 4.000 | 7 | 28.000 |
| PC | - | - | - | 8.000 | 2 | 16.000 | 8.000 | 3 | 24.000 |
| VENT | - | - | - | 500 | 5 | 2.500 | 500 | 3 | 1.500 |
| 10/01 | TV | - | - | - | 4.000 | 5 | 20.000 | 4.000 | 2 | 8.000 |
| PC | 9.000 | 3 | 27.000 | 8.000 | 1 | 8.000 | 8.600 | 5 | 43.000 |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| 20/01 | TV | - | - | - | - | - | - | 4.000 | 2 | 8.000 |
| PC | - | - | - | 8.600 | 3 | 25.800 | 8.600 | 2 | 17.200 |
| VENT | - | - | - | - | - | - | 500 | 3 | 1.500 |
| **TOTAL** |  |  |  | **111.000** |  |  | **84.300** |  |  | **26.700** |

\* Por ser CMP, quando entram os novos PCs, simplesmente somam-se os saldos antigos com os novos e divide-se esta soma pela quantidade de produtos no estoque para encontrar o custo unitário novo (Custo Médio Ponderado - CMP)

A **soma das SAÍDAS** é o Custo de Mercadorias Vendidas (CMV), e a **soma do SALDO** é o valor que deve constar como Estoque no BP.

**MVC-P para o Controle de Estoques CMP**

Texto branco sobre fundo preto

O conteúdo gerado por IA pode estar incorreto.**MODEL:** é onde codificamos as classes conforme modeladas no Diagrama de Classes. Para o trabalho, todas as classes serão codificadas em Python.

**EXEMPLO:**

class Cliente:

def \_\_init\_\_(self, id, cpf, nome, None):

self.id = id

self.cpf = cpf

self.nome = nome

//o id, na verdade, seria preenchido pelo próprio BD/SGBD

//quando se instancia o Cliente, pedidos é iniciado como Nulo, o que ajuda a salvar memória secundária. No cenário que o possível Cliente criou a conta mas NUNCA compra qualquer coisa, a melhor opção é Nulo para a coleção pedidos

//quando o Cliente completa qualquer compra, verifica-se se pedidos é Nulo. Se for, cria-se a coleção pedidos; Se não for, simplesmente adiciona-se o novo pedido à coleção

def get\_id()

return self.id

def set\_id()

self.id = id

def get\_cpf()

return self.cpf

def set\_cpf()

self.cpf = cpf

def get\_nome()

return self.nome

def set\_nome()

self.nome = nome

def get\_pedidos()

return self.pedidos

//nunca se cria um SET para coleções (vetores, listas, set, etc)

def to\_string:

return f{“CLIENTE: self.get\_nome() | CPF: self.get\_id().”}

**OBS:** os métodos **CRUD (criar, pesquisar, atualizar e deletar)** não são criados no **MODEL**, eles são delegados para a camada **CONTROLLER**

**CONTROLLER:** é onde codificamos os métodos CRUD para uma classe criada no MODEL. Para cada Entidade (no caso do nosso trabalho são Cliente, Fornecedor e Produto) no Model será criada uma classe Controller\_<ENTIDADE>, então teremos Controller\_Cliente, Controller\_Fornecedor e Controller\_Produto.

As classes **Controller** não possuem **construtor**, **get** nem **set**, somente as funções CRUD e outras complementares. No código do professor, as funções das classes Controller sempre iniciam uma conexão (com permissão de alteração) com o BD antes do resto da função correr.

Se o código do professor parecer confuso, exclua todos os comentários do professor que o código ficará mais claro.

**EXEMPLO:**

from model.clientes import Cliente

from conexion.oracle\_queries import OracleQueries

class Controller\_Cliente:

def \_\_init\_\_(self):

pass

#---------------------------------------------------------------------------

def pesquisaCliente(…) -> bool:

df\_cliente = oracle.sqlToDataFrame(…)

return df\_cliente.empty

#---------------------------------------------------------------------------

def novoCliente(…) -> Cliente :

oracle = OracleQueries(can\_write=True)

oracle.connect()

cpf = input(“CPF: ”)

if self.pesquisaCliente(oracle, cpf):

nome = input(“Nome: ”)

oracle.write(...)

df\_cliente = oracle.sqlToDataFrame(...)

novoCliente = Cliente(...)

print(novoCliente.toString())

return novoCliente

else:

print(f”O CPF {cpf} já está cadastrado.”)

return None

#---------------------------------------------------------------------------

def atualizaCliente(…) -> Cliente:

oracle = OracleQueries(can\_write=True)

oracle.connect()

cpf = input(“CPF (Novo): ”)

if not self.pesquisaCliente(oracle, cpf):

novo\_nome = input(“Novo nome: ”)

oracle.write(...)

df\_cliente = oracle.sqlToDateFrame(...)

clienteAtualizado = Cliente(...)

print(clienteAtualizado.toString())

return clienteAtualizado

else:

print(f”O CPF {cpf} não existe.”)

return None

#-------------------------------------------------------------------------

def excluirCLiente(self):

oracle = OracleQueries(can\_write=True)

oracle.connect()

cpf = input(“CPF: ”)

if not self.pesquisaCliente(oracle, cpf):

df\_cliente = oracle.sqlToDataFrame(…)

oracle.write(f”delete from clientes where cpf = {cpf}”)

clienteExcluido = Cliente(…)

print(“Cliente removido com sucesso!”)

print(clienteExcluido.toString())

else:

print(f”O CPF {cpf} não existe.”)

#-------------------------------------------------------------------------

def verPedidos() -> String[]:

oracle = OracleQueries(can\_write=False)

oracle.connect()

cpf = input(“CPF: ”)

if self.pesquisaCliente(oracle, cpf):

#aqui, caso o CPF conste no BD, vamos montar uma String com cada pedido

#do Clinte no modelo

#“dd/mm/yyyy – idVenda - Produto(s) – Preço Un. Médio – Quant. - Total”

else:

print(f”O CPF {cpf} não existe.”)

Os Controllers a serem codificados são:

* Cont\_Cliente
* Cont\_Compra
* Cont\_Fornecedor
* Cont\_Produto
* Cont\_Venda

**VIEW:** cada View é uma interface que o Usuário pode requisitar. Em um site de empregos, por exemplo, uma das páginas do site é a “Meu Perfil” onde o Usuário pode preencher dados em falta ou alterar dados previamente preenchidos. A página “Vagas”, onde o usuário pode fazer a busca de oportunidades de emprego, é uma outra View.

A View pode mudar bastante de acordo com o objetivo da mesma. Enquanto a página “Meu Perfil” não vai ter um espaço para iniciar pesquisa de vagas, mas pode ter um espaço para pesquisa geral no site, a página “Vagas” obrigatoriamente terá uma forma de pesquisar por vagas, e, como é comum ver, ela provavelmente terá também o espaço de pesquisa geral no site.

As views, no nosso trabalho, são resultado do sqlToDataFrame do professor, onde um script python será ativado, executando uma pesquisa (query(ies)) SQL e pegando a Relação/Tabela de resposta para gerar uma visualização de fácil compreensão.

O relatório (View) **relatorio\_pedidos.sql**, **relatorio\_pedidos\_por\_fornecedor.sql** e **relatorio\_itens\_pedidos.sql** serão substituídos por:

* **relatorio\_compras.sql** | **relatorio\_vendas.sql**
* **relatorio\_compras\_fornecedor.sql** | **relatorio\_vendas\_cliente.sql**
* **relatorio\_compra\_id.sql** | **relatorio\_venda\_id.sql**

Outra especificidade nossa é necessitar de um relatorio\_estoque.sql, que vai gerar a View de Controle de Estoque. Esta View do Estoque deve sair parecida com:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DATA | ID\_PROD | NOME | EN.PR | EN.QTD | EN.SAL | S.PR | S.QTD | S.SAL | ES.PR | ES.QTD | ES.SAL |
| 01/01/2001 | 1 | A | 10.0 | 100 | 1000.0 | 10.0 | 30 | 300.0 | 10.0 | 70 | 700.0 |

Cada código SQL desses gera um View específica, e o professor concentrou todas as chamadas no arquivo relatorios.py. Os códigos SQL em si estão na pasta SQL do projeto do professor.