Bases de Dados

Versão 1.0

1. Bases de Dados

1.1. Modelo Entidade Relacionamento

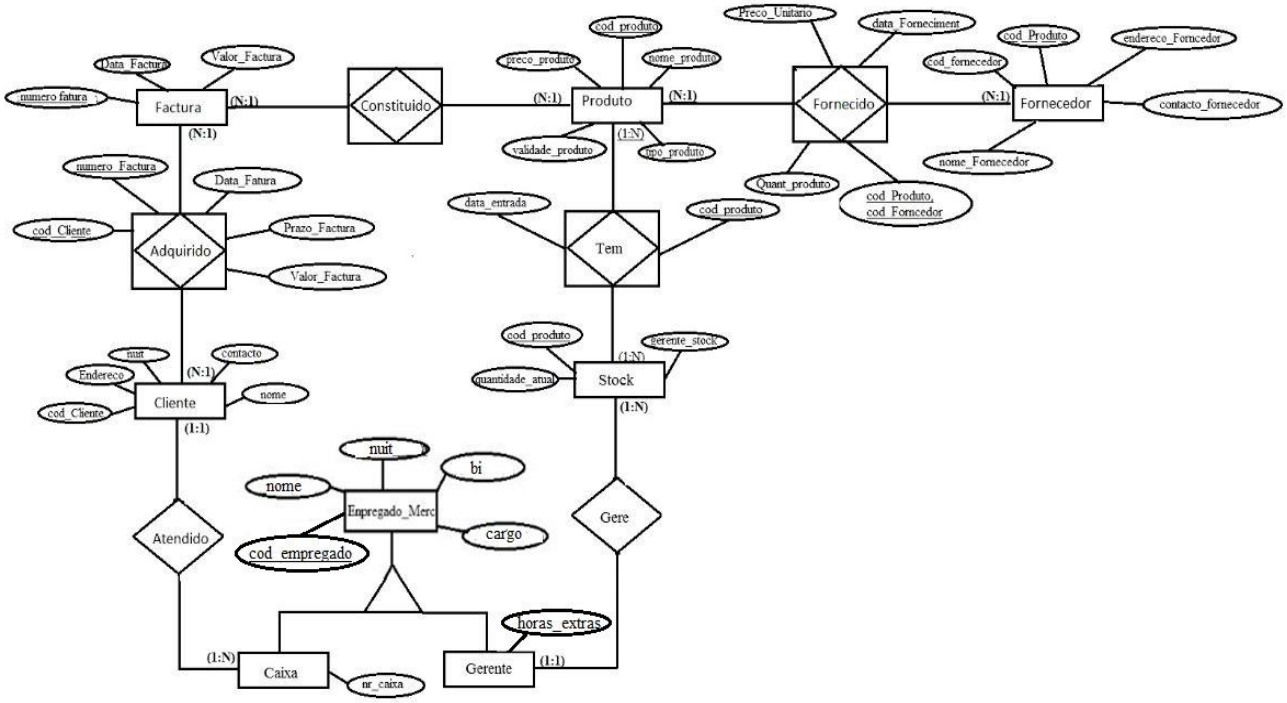


Figura 1 – Modelo Entidade Relacionamento

1.2. Tabela Normalizada

Cliente(cod_Cliente,nome,nuit,contacto,sexo,endereco)

Produto(cod_Produto,nome_Produto,tipo_Produto,preco_Produto,validade_Produto)

Factura(Numero_Factura,Prazo_Factura,Valor_Factura)

Cliente_Factura(numero_Factura,cod_Produto,quant_Compra,tipo_Compra)

Stock (cod_Produto, quantidade)

Stock_Produto (cod_Produto, Data_Entrada, Dada_Saida)

Fornecedor (cod_Forncedor, nome_Fornecedor, endereco_Forncedor)

Forncedor_Produto(cod_Produto,cod_Forncedor,data_Fornecimento,Quantidades,PrecoUnitario)

Empregado(Cod_Empregado, nome, nuit,contacto, sexo, endereco, ocupacao)

1.3. Dicionário de dados

Cliente					
Campo	Chave	Descrição	Tipo	Tamanho	Observação
Cod_Cliente	Primaria	Código do Cliente	Inteiro		
Nome		Nome do Cliente	Caracter	20	
Contacto		Contacto do Cliente	Inteiro	9	
Nuit		NUIT do cliente	Inteiro	9	
Sexo		Sexo do Cliente	Caracter	1	
Endereco		Endereço do Cliente		25	

Produto					
Campo	Chave	Descrição	Tipo	Tamanho	Observação
Cod_Produto	Primaria	Código do Produto	Inteiro		
Nome_Produto		Nome do Produto	Caracter	20	
Tipo_Produto		Tipo de Produto	Caracter	20	
Preco_Produto		Preço do Produto	Decimal	3	
Validade_Produto		Validade do Produto	Data		

Stock					
Campo	Chave	Descrição	Tipo	Tamanho	Observação
Quantidade		Quantidade do Stock	Inteiro	20	
Cod_Produto	Estrangeira	Código do Produto	Inteiro	20	

Stock_Produto					
Campo	Chave	Descrição	Tipo	Tamanho	Observação

Data_entrada		Data de entrada do stock	Date		
Data_saida		Data de saída do stock	Date		

Fornecedor					
Campo	Chave	Descrição	Tipo	Tamanho	Observação
Cod_Fornecedor	Primária	Código da Solicitação	Inteiro		Chave primária da solicitação
Nome_Fornecedor		Nome do Fornecedor	Caracter	20	
Endereco_Fornecedor		Endereço do Fornecedor	Caracter	25	

Fornecedor_Produto					
Campo	Chave	Descrição	Tipo	Tamanho	Observação
Cod_Fornecedor, Cod_Produto	Primária	Código do Fornecedor e do Produto	Inteiro		Chave primária da solicitação
Data_Fornecimento		Data do Fornecimento	Data		
Quantidades		Quantidade de Produtos	Inteiro	20	
Cod_Fornecedor, Cod_Produto	Estrangeira	Número de serei	Inteiro	20	

Empregado

Campo	Chave	Descrição	Tipo	Tamanho	Observação
Cod_Empregado	Primaria	Código do Cliente	Inteiro		
Nome		Nome do Cliente	Caracter	20	
Contacto		Contacto do Cliente	Inteiro	9	
Nuit		NUIT do cliente	Inteiro	9	
Sexo		Sexo do Cliente	Caracter	1	
Endereco		Endereço do Cliente	Caracter	25	
Ocupação		Cargo do empregado	Caracter	25	

1.4. Consultas

i. Lista de Interrogações a Base de Dados

1. Liste o nome do produto ordenado dos mais barato ao mais caro
2. Liste o total das facturas
3. Liste os Produtos com quantidade superior a 100:
4. Liste todos os dados do cliente cuja compra é do tipo à prazo:
5. Consulta que retorna código, nome, validade, e preços de todos os produtos perecíveis.
6. Consulta que retorna código, nome, validade, e preços de todos os produtos cujo a validade e inferior ou igual a 30 dias
7. Consulta que retorna a quantidade atual do stock
8. Consulta que retorna todoss os dados dos clients cujo o prazo da facture e o dia corrente.

ii. Algébra Relacional

1. $\pi_{\text{nome\_Produto}}(\sigma^{(Produto)}_{Order\ by\ (Preco_{Produto})})$

2. $\sigma^{(Factura)}_{Sum(Valor_{Factura})}$

3. $\pi_{\text{Cod\_Produto,nome\_Produto,tipo\_produto,preco\_Produto,validade\_Produto}}(\sigma^{(ProdutoXStock)}_{quantidade>100})$

4. $\pi_{\text{Cod\_Cliente,nome,nuit,contacto,Sexo,Endereco}}(\sigma^{(ProdutoXClienteXCliente\_Produto)}_{tipo_{Compra='a\ prazo'}})$

5. $\pi_{\text{Cod\_Produto,nome\_Produto,preco\_Produto,validade\_Produto}}(\sigma^{(Produto)}_{tipo_{Produto='pericivel'}})$

6. $\pi_{\text{Cod\_Produto,nome\_Produto,preco\_Produto,tipo\_Produto}}(\sigma^{(ProdutoXStock\_Produto)}_{validade\_Produto-Data\_Entrda<=30})$

7. $\pi_{quantidade\_atual}(\sigma^{(Stock)})$

8. $\pi_{*}(\sigma^{(ClienteXFactura\_Cliente)}_{Factura_{Cliente}=getData()})$

iii. Tradução das Consultas para Sql

1. SELECT nome_Produto,preco_Produto

FROM Produto

ORDER by Preco_Produto DESC;

2. SELECT Sum (Valor_Factura) as Total

FROM Factura;

3. SELECT *

FROM Stock, Produto

WHERE quantidade> 100

4.SELECT Cod_Cliente, nome, nuit, contacto, Sexo, Endereco

FROM Cliente, Produto

WHERE tipo_Compra=’A prazo’;

5. SELECT cod_Produto, nome_Produto, validade_Produto, preco_Produto

FROM Produto

WHERE tipo_Produto= ‘perecível’

6. SELECT cod_Produto, nome_Produto, tipo_Produto, preco_Produto

FROM Produto, Stock_Produto

WHERE validade_Produto - Data_Entrada<=30

7. SELECT quantidade_atual

FROM Stock

8. SELECT *

FROM Cliente, Fatura_Cliente

WHERE Fatura_Cliente= getData();

1.5. Triggers

Trigger para actualizar o Stock após a compra de um produto.

Create Trigger act_Stock after insert on cliente_produto

->for each row update stock set quantidade= quantidade -new. Quant_Compra;

Trigger para actualizar o Stock após o cancelamento de uma compra.

Create Trigger d_act_Stock after delete on Cliente_produto

->for each row update stock set quantidade = quantidade +old. Quant_Compra;

Trigger para actualizar o Stock após o fornecimento de um produto.

Create Trigger actu_Stock after insert on Fornecedor_Produto

->for each row update stock set quantidade= quantidade+new.Quantidades;

Trigger para actualizar o Stock após o cancelamento de um fornecimento.

Create Trigger d_actu_Stock after delete on Fornecedor_Produto ‘

->for each row update stock set quantidade= quantidade-old.Quantidades;

Trigger para adicionar o IVA ao preço de um Produto

Create Trigger Preco_prod before insert on Produto

for each row update Produto set Preco_Produto= (0.17*new.Preco_Produto)+new.Preco_Produto;

1.6. Procedure

- Criar Procedimento que lista todos os fornecedores e sua especialidade

```
CREATE PROCEDURE listarFornecedor()

BEGIN

SELECT nome_fornecedor,Especialidade_fornecedor FROM fornecedor;

End;
```

1.7. View

- Criar uma view que mostra o nome o valor a pagar de todos os clientes com compra à prazo

```
CREATE VIEW View_Cliente_Prazo

AS

SELECT nome,(Quant_Compra*Preco_Produto)

FROM Cliente, Produto, Cliente_Produto

WHERE tipo_Compra=’a prazo’

GROUP BY Nome;
```

1.8. Transações

Transações que possam ocorrer na Base de dados durante uma determinada operação:

Registo de compra e actualização de *stock*

T1: "Registo de compras e actualização de *stock*"
1. Buscar produtos existentes no *Stock* 2. Buscar informação da quantidade de cada item 3. Actualizar e gravar produtos no *stock* 4. Buscar saldo actual 5. Actualizar e gravar saldo actual 6. Actualizar e gravar a quantidade de cada item 7. Confirmar

Transação 1 - Registo e actualização de stock

Registo de fornecimento

T2: "Registo de fornecimento"
1. Buscar quantidade de produtos existentes no *Stock* 2. Buscar informação sobre a quantidade de cada item 3. Introduzir o novo produto e actualizar 4. Registrar novos grupos de item 5. Efectuar a gravação 6. Confirmar

Transação 2 - Registo de Fornecimento

Registos da informação visando evitar a perda de informação.

T3: "Registo de fornecedores e clientes"
1. Buscar a informação referente aos fornecedores e clientes 2. Buscar dividas a pagar pelos clientes 3. Actualizar e gravar saldo de dívidas pagas e valor a pagar aos fornecedores 4. Introduzir novos clientes e efectuar a respectiva gravação 5. Confirmar

Transação 3 - Registo de fornecedores e clientes

T4: “Actualização de funcionários”
1. Buscar as informações referentes aos funcionários 2. Verificar a existência ou não do funcionário 3. Introduzir e gravar novos funcionários 4. Actualizar a informação 5. Confirmar

Transação 4 - Actualização de funcionários

Quando o cliente faz uma consulta mas não chega a retirar os produtos (sem saída de produtos)

T5: “Cotação”
1. Buscar os produtos existentes no Stock; 2. Armazenar numa variável factura os produtos solicitados; 3. Imprimir a factura. 4. Confirmar

Transação 5 - Cotação

T_1

Start trans

Read (Stock)

Read (Item)

Stock=stock-ped

Write (Stock)

Read (lucro)

Lucro=lucro+valor.itens

Write (lucro)

Item=item-n.itens

Write (Item)

Commit

T2

Start Trans

Read (Stock)

Read (Item)

Stock=stock+entrada

Write (Stock)

Item=item+n.item

Write (Item)

Commit

T3

Start Trans

Read (Clien

Read (Forn

Cliente= divCli

Fornecedor= divFor

Write (Cliente)

Write (Fornecedor)

Commit

T4

Start Trans

Read (Funcionario)

Funcionario=Funcionario+new.funcionario

Write (Funcionario)

Commit

T5

Start Trans

Read (Cliente)

Read (Compra.Produto)

Factura=Compra.Produto;

Print (Factura)

Commit

1.9. Operações de Concorrência

Analisando duas transações que ocorrem simultâneo:

T_1	T_2
Start trans	**Start Trans**
Read (Stock)	Read (stock)
Read (Item)	Read (Item)
Stock=stock-ped	Stock=stock+entrada
Write (Stock)	Write (Stock)
Read (lucro)	Item=item+n.item
Lucro=lucro+valor.itens	Write (Item)
Write (lucro)	Commit
Item=item-n.itens	
Write (Item)	
Commit	

Após a realização das duas transações encontraremos os seguintes problemas de concorrência:

- **Actualização perdida**: Neste caso as duas transações vão acessar o mesmo dado (*stock*), tendo suas operações intercaladas, tornando o dado incorrecto, pois vamos ter um *stock* após a transação 1 e um outro stock após a transação 2.
- **Análise inconsistente**: Neste caso encontramos uma análise inconsistente primeiramente por lidarmos com operações de agregação. O valor do *stock* na transação 2 vai ser modificado sem tomar em consideração o facto de que o mesmo dado *stock* também está sendo modificado na transação 1, modificando um valor já incluído na agregação, gerando uma análise inconsistente, no momento de *commit* (na qual somente será considerada a última confirmação).
- **Leitura suja**: Neste caso pode ocorrer o presente conflito quando há uma modificação externa como falta de energia, problemas de sistemas e outro tipo de alteração que possa resultar em *rollback*.

T3

Start Trans

Read (Cliente)

Read (Fornecedor)

Cliente= divCli

Fornecedor= divFor

Write (Cliente)

Write (Fornecedor)

Commit

T5

Start Trans

Read (Cliente)

Read (Compra.Produto)

Factura=Compra.Produto;

Print (Factura)

Commit

Após a realização das duas transações encontraremos os seguintes problemas de concorrência:

- **Leitura Suja:** Neste caso podemos ter que a transação 5 (passo a citar como T5), pode procurar um cliente enquanto este ainda esta a ser registado na transação 3 (passo a citar como T3), e quando em T5 não encontrar faz um *rollback* enquanto em T3 já esta registado o cliente, causado assim uma leitura suja.

1.10. Técnicas de resolução de conflitos

Serialização

Conflito entre as transações 1 e 2.

T1: r_1 (stock), r_1 (item), w_1 (stock), w_1 (lucro), w_1 (lucro), w_1 (item)

T2: r_2 (stock), r_2 (item), w_2 (stock), w_2 (item)

S: r_1 (stock), r_1 (item), w_1 (stock), w_1 (lucro), w_1 (lucro), w_1 (item), r_2 (stock), r_2 (item), w_2 (stock), w_2 (item)

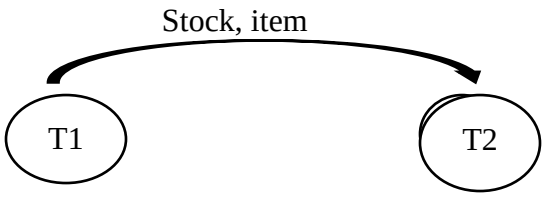
Conflitos:

r_1 (stock) $<_s$ w_2 (stock)

r_1 (item) $<_s$ w_2 (item)

w_1 (stock) $<_s$ w_2 (stock)

w_1 (item) $<_s$ w_2 (item)



T1→T2

É Serializável.

Conflito entre as transações 3 e 5.

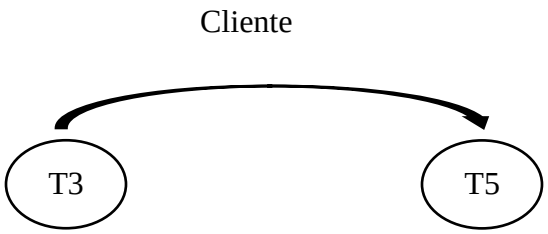
T3: r_3 (cliente), r_3 (fornecedor), w_3 (cliente), w_3 (fornecedor)

T5: r_5 (cliente), r_5 (compra.produto)

S:r_3 (cliente), r_3 (fornecedor), w_3 (cliente), w_3 (fornecedor), r_5 (cliente), r_5 (compra.produto)

Conflito:

W3 (cliente) $<_s$ r_5 (cliente)



T3→T5

É Serializável.

Bloqueios

Apliacação da tecnica de bloqueios entre as transações T1 e T2.

T1	**T2**
Start trans	**Start Trans**
lock_x(Stock)	Lock_x(Stock)
Read (Stock)	…
Lock_x(Item)	
Read (Item)	
Stock=stock-ped	
Write (Stock)	
Unlock(Stock)	
Lock_x(lucro)	Read (Stock)
Read (lucro)	Lock_x (Item)
Lucro=lucro+valor.itens	…

oqueios entre as transações T

T3	**T5**
Start Trans	**Start Trans**
Lock_x (Cliente)	Lock_s (Cliente)
Read (Cliente)	…
Lock_x (Fornecedor)	Read (Item)
Read (Fornecedor)	Stock=stock+entrada
Cliente= divCli	Write (Stock)
Fornecedor= divFor	Unlock(Stock)
Write (Cliente)	Item=item+n.item
Unlock (Cliente)	Write (Item)
Write (Fornecedor)	Unlock(Item)
Unlock (Fornecedor)	Commit
Commit	

Read (Cliente)
Read (Compra**.**Produto)
Factura=Compra.Produto;
Print (Factura)
Commit

1.11. Análise do volume de transações

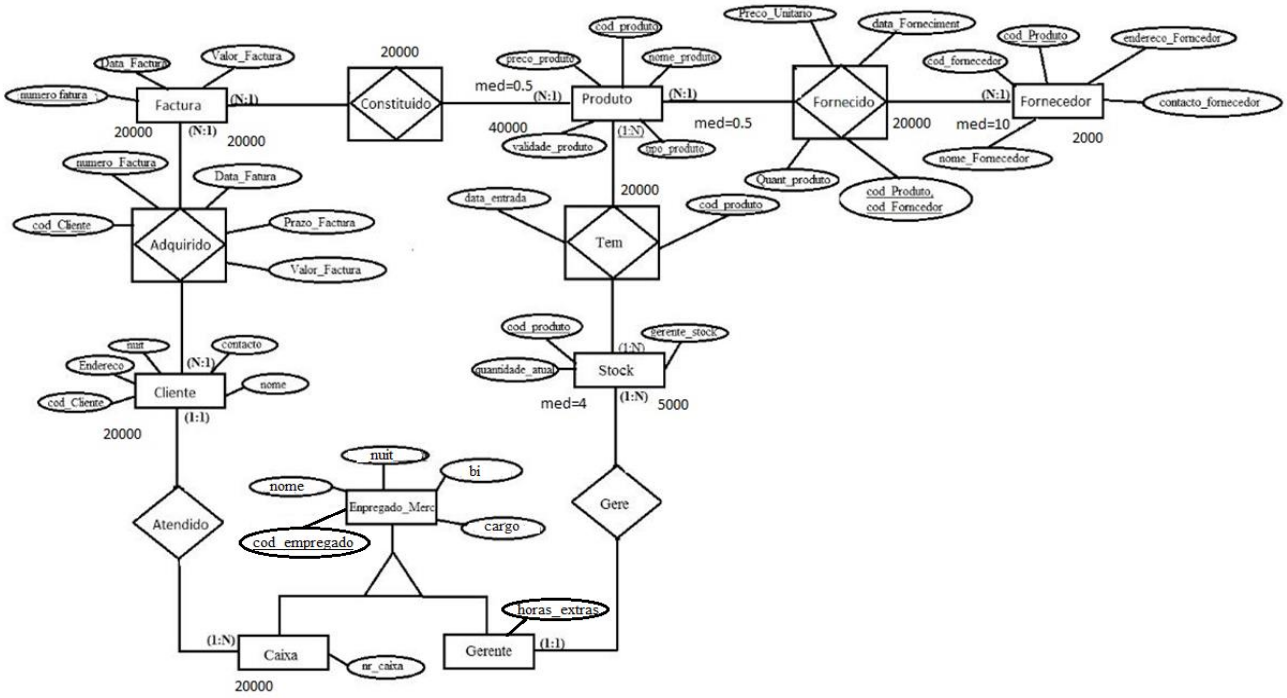


Figura 2 - Modelo entidade relacional e suas cardinalidades

Operações	**Freq**	**Conceito**	**E\R**	**R\W**	**Média de Concorrência**
Cliente compra produto	100 vezes/dia	Cliente Compra Produto	E R E	R W R	100 100x1=100 100x0.5=50
Compra gera factura	100 vezes/dia	Cliente Gera Factura	E R E	R - R	100 100x1=100 100x1=100

Fornecedor fornece produto	1 vez/dia	Fornecedor Fornece Produto	E R E	R - W	1 1x10=10 1x0.5=0.5
Empregado antende um cliente	100 vezes/dia	Empregado atende Cliente	E R E	R W W	100 100x0.5=50 100x0.5=50
Buscar produto no stock	150 vezes/dia	Produto Tem Stock	E R E	R - R	150 150x0.5=75 150x4=600

Tabela 1 - volume de acessos lógicos das operações

1.12. Matriz CRUD

	Funcionalidade/Entidade	**Cliente**	**Produto**	**Factura**	**Stock**	**Empregado**	**Fornecedor**
Comprar	Cliente compra novo produto	R	RDU	R	RU		
	Cliente pede cotação do produto	R	R		R		
Gerar	Cliente solicita factura	R	RDU	R	RU		
	Empregado debita valores			R		RU	
	Registar Clientes	RCU				R	

Atender	Registar Fornecedor					R	RCU
Fornecer	Fornecedor entrega produto		RU		RU		R
	Fornecedor envia cotação					R	R
Ter	Produto consulta o stock		R		R		
	Produto é retirado do stock		RDU		RDU		

Tabela 2- Matriz CRUD

1.13. Desnormalização

4.13.1. Tabelas Normalizadas

Cliente(cod_Cliente,nome,nuit,contacto,sexo,endereco)

Produto(cod_Produto,nome_Produto,tipo_Produto,preco_Produto,validade_Produto)

Stock (cod_Produto, quantidade)

Stock_Produto (cod_Produto, Data_Entrada, Dada_Saida)

Fornecedor (cod_Forncedor, nome_Fornecedor, endereco_Forncedor)

Forncedor_Produto(cod_Produto,cod_Forncedor,data_Fornecimento,Quantidades,PrecoUnitario)

Factura(cod_Cliente, cod_Produto,numero_Factura,Prazo_Factura,Valor_Factura)

4.13.2. Tabelas Dernormalizadas

Imaginando uma situação em que seja preciso constantemente retornar o valor da factura. Pode ser interessante criar um campo req_Factura do pedido na tabela Cliente para que, toda vez que se precisar calcular o valor da factura não seja necessário percorrer seus itens e depois, os produtos, para chegar a este valor.

Produto(cod_Produto,nome_Produto,tipo_Produto,preco_Produto,validade_Produto,
req_Factura)

Factura(cod_Cliente, cod_Produto,numero_Factura,Prazo_Factura,Valor_Factura, req_Factura).

1.14.Particionamento

Tabela cliente

cod_Cliente	nome	Nuit	Contacto	Endereço
1111	Marcello	123526580	820053382	Magoanine “a”
1112	Judelson	1235557280	825152027	3 de fevereiro
…	…	…	…	…

Tabela 3- Cliente não particionada

cod_Cliente	nome
1111	Marcello
1112	Judelson
…	…

Tabela 4- Tabela particionada

NB:Conjunto mais usado

cod_Cliente	Nuit	Contacto	Endereço
1111	123526580	820053382	Magoanine “a”
1112	1235557280	825152027	3 de fevereiro
…	…	…	…

Tabela 5- Conjuto menos Usado

cod_Produto	Nome_Produto	Tipo_Produto	Preco_Produto	Validade_produto
2000	Arroz Ngonhama	Não perecível	1800	01/08/2050
2001	Refresco	Perivel	40	01/02/2017
2002	Bolachas	Perecível	55	01/02/2017
2003	Açúcar	Perecível	80	01/08/2050
…	…	…	…	…

cod_Produto	Nome_Produto	Tipo_Produto	Preco_Produto	Validade_produto
2000	Arroz Ngonhama	Não perecível	1800	01/08/2050
2003	Açúcar	Perecível	80	01/08/2050
…	…	…	…	…

Tabela 6- Tabela particionada

cod_Produto	Nome_Produto	Tipo_Produto	Preco_Produto	Validade_produto
2001	Refresco	Perivel	40	01/02/2017
2002	Bolachas	Perecível	55	01/02/2017
…	…	…	…	…

Tabela 7- Tabela particionada

Tabela Stock

cod_Produto	quantidade_Atual	Gerente_stok
2000	10	Luthermilla
2001	8	Luthermilla
…	…	…

Tabela 8- Tabela Stock não particionada

cod_Produto	quantidade_Atual
2000	10
2001	8
…	…

Tabela 9- Tabela Particionada

cod_Produto	Data_Entrada
2000	01/05/2016
2001	02/06/2016
…	…

Tabela 10- Stock_Produto não particionada

1.15. Índices

O uso de índices pode trazer grandes melhorias para o desempenho do banco de dados. Pensando nisso, estes serão implementados de modo a facilitar e optimizar as consultas a serem feitas na base de dados.

Para perceber melhor como isso acontece: os índices percorrem a estrutura da árvore do índice para localizar os registros, por comparação, extraindo somente aqueles registros necessários para satisfazerem os critérios passados pela consulta.

Os índices aceleram a recuperação dos dados, além disso estes são sempre bem vindos em colunas de grande seletividade, como por exemplo, além da chave primária.

Nessa vertente será criado um índice na tabela produto para a variável nome do produto, que estará sempre presente nas consultas, um outro índice na tabela cliente para o nome do cliente outro dado com alta demanda nas colsultas.

1.16. Implementação

Usando a estrutura de dados BTREE (que é organização indexada caracterizada por uma ordenação segundo os dados armazenados permitindo um acesso rápido sobre um valor exacto ou parte da chave) teremos:

```
CREATE INDEX IndicePorNomeProduto ON Produto

WITH STRUCTURE= BTREE,

KEY=(Nome_Produto)


CREATE INDEX IndicePorNomeCliente ON Cliente

WITH STRUCTURE= BTREE,

KEY=(Nome)
```