Projeto Final
“Minimercado”

Eduardo Palma nº2955

Nuno Melo nº3015

Ricardo Santos nº3016

Docente: Norberto Albino

Unidade Curricular: Base de Dados

Curso: Licenciatura em Bioinformática

Data: 29-06-2020

Índice

Índice de Figuras

Introdução

Pretende-se desenvolver uma base de dados para efetuar a gestão de um minimercado. O minimercado está estruturado da seguinte forma: produtos, categoria de produtos, vendas, clientes, fornecedores, tipo de pagamento, origem dos produtos, colaboradores e as funções que estes desempenham.

No minimercado encontram-se disponíveis diversas categorias de produtos para venda: produtos alimentares, produtos de limpeza, produtos de higiene pessoal e produtos de papelaria. Cada produto vai estar identificado com o código de produto, categoria, código de encomenda, designação e origem. Encomendamos produtos de vários países diferentes, Portugal, México, Espanha e Inglaterra.

Temos disponíveis vários tipos de pagamento: numerário, multibanco, MBWay e cheque.

Ao efetuar-se uma encomenda aos fornecedores, é atribuído um código de fornecedor e das respetivas encomendas, sendo recolhido o seu nome e o telefone.

Para garantir o funcionamento do minimercado existem vários colaboradores, com funções diferentes: reposição dos produtos, inventário, o atendimento ao público e a manutenção do estabelecimento com as condições de higiene necessárias ao bom funcionamento do mesmo (COVID-19).

Cada colaborador contratado, tem de fornecer o seu nome, telefone, NIF, NIB e a sua morada, após o qual lhe será atribuído um código de colaborador.

Ao efetuar-se a venda dos produtos, regista-se a data da venda, o valor monetário e o produto. Desta venda origina-se uma fatura com o NIF e nome do cliente, caso ele assim o deseje.

Modelo Relacional

1FN

Na 1FN, temos os atributos chave, onde os atributos dependem da chave primária.

Venda(**Codigovenda**,CodigoFatura,CodigoProduto,Nif,NumID,Valor, CodigoTipoPag,**NIF**,Nome,**CodigoFatura,**NIF,DataFatua,**CodigoTipoPag**,Tipo Pag,**CodigoOrigem**,País,**CodigoEncomenda**,CodigoFornecedor,Nome, Telefone,**CodigoCategoria**,Designacao,**CodigoProduto**,Designacao, CodigoEncomenda,CodigoCategoria,Preço,CodigoOrigem,**CodigoFunção**, Designacao, **NumID**, Nome, Telefone, CodigoFuncao)

2FN

Na 2FN, temos os atributos da 1FN sem dependências parciais.

Venda (**Codigovenda**,CodigoFatura,CodigoProduto,Nif,NumID,Valor,CodigoTipoPag);

Clientes (**NIF**,Nome);

Fatura (**CodigoFatura,**NIF,DataFatua);

TipoPagamento(**CodigoTipoPag**,TipoPag)

Produtos (**CodigoProduto**,Designacao,CodigoEncomenda,CodigoCategoria,Preço, CodigoOrigem,**CodigoOrigem**,País,**CodigoEncomenda**,CodigoFornecedor, Nome,Telefone,**CodigoCategoria**,Designacao)

Colaboradores(**NumID**,Nome,Telefone,CodigoFuncao,**CodigoFunção**, Designacao)

3FN

Na 3FN, não existem dependências entre os atributos não chave, tem de estar na 2FN e sem dependências transitivas.

Venda(**Codigovenda**,CodigoFatura,CodigoProduto,NIF,NumID,Valor, CodigoTipoPag);

Clientes (**NIF**,Nome);

Fatura (**CodigoFatura,**NIF,DataFatua);

TipoPagamento (**CodigoTipoPag**,TipoPag)

Produtos (**CodigoProduto**,Designacao,CodigoEncomenda,CodigoCategoria,Preço, CodigoOrigem)

Origem (**CodigoOrigem**,País)

Fornecedores (**CodigoEncomenda**,CodigoFornecedor,Nome,Telefone)

Categoria (**CodigoCategoria**,Designacao)

Colaboradores (**NumID**, Nome, Telefone, CodigoFuncao)

Função (**CodigoFunção**, Designacao)

Modelo Entidade Relação

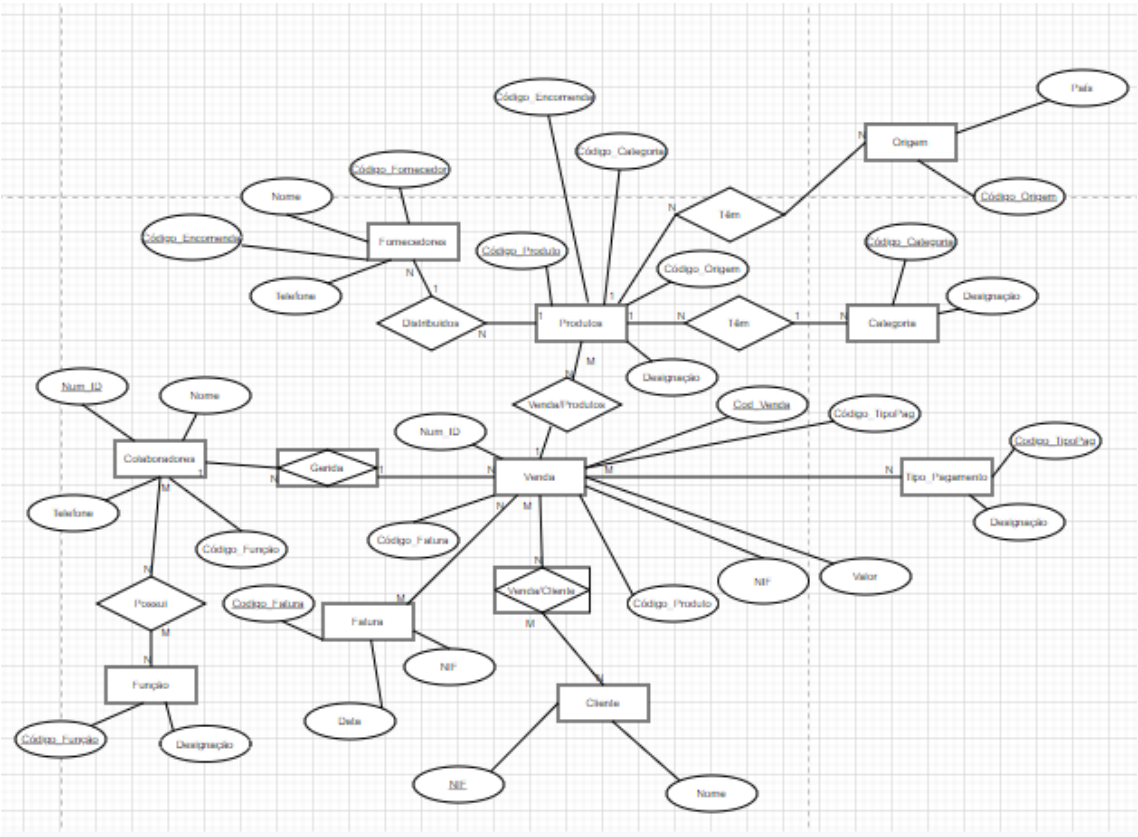

Figura 1 – Modelo Entidade Relação

Este modelo foi enviado também num ficheiro draw.io.

Resumo de opções técnicas e objetos desenvolvidos

Views

A view pode ser definida como uma tabela virtual composta por linhas e colunas de dados vindos de tabelas relacionadas em uma query. As linhas e colunas da view são geradas dinamicamente quando é feita uma referência a ela.

```
CREATE VIEW ClientesVezesLoja
AS
SELECT
    count(DISTINCT V.NIF) AS [Número de Idas],
    C.Nome AS Nome

FROM PF.Venda AS V
INNER JOIN PF.Clientes AS C ON C.NIF = V.NIF
GROUP BY Nome
GO
```

Figura 2 – View 1

Usamos o count para contar o número de idas e depois juntamos a tabela clientes com a tabela venda, como mostra a figura 2.

```
CREATE VIEW VerProdutoPreço
AS
SELECT
    P.Designacao AS Produto,
    P.Preço AS Preço

FROM PF.Produtos AS P
GO
```

Figura 3 – View 2

Usamos o comando select para mostrar o produto e o preço , como mostra a figura 3.

```
CREATE VIEW NrProdutosVenda
AS
SELECT
    V.CodigoFatura AS [Número de Venda],
    count(CodigoVenda) AS [Número de Produtos],
    C.Nome AS Nome

FROM PF.Venda AS V
INNER JOIN PF.Clientes AS C ON C.NIF = V.NIF
GROUP BY CodigoFatura, Nome
GO
```

Figura 4 – View 3

Usamos o count para contar o número de produtos e juntamos a tabela de clientes com a da venda e organizamos por código de fatura e nome, como mostra a figura 4.

```
CREATE VIEW NrProdutosEncomenda
AS
SELECT
    P.CodigoEncomenda AS [Número da Encomenda],
    count(P.CodigoEncomenda) AS [Número de Produtos],
    F.Nome AS Nome
FROM PF.Produtos AS P
INNER JOIN PF.Fornecedores AS F ON F.CodigoEncomenda = P.CodigoEncomenda
GROUP BY Nome, P.CodigoEncomenda
GO
```

Figura 5 – View 4

Usamos o count para contar o número de produtos e juntamos a tabela dos fornecedores à tabela dos produtos, como mostra a figura 5.

```
CREATE VIEW FuncoesColaborador
AS
SELECT
    C.Nome AS Nome,
    F.Designacao AS Função

FROM PF.Funcao AS F
INNER JOIN PF.Colaboradores AS C ON C.CodigoFuncao = F.CodigoFuncao
GO
```

Figura 6 – View 5

Usamos o comando select para ver o nome dos colaboradores e as suas funções e de seguida juntamos a tabela dos colaboradores à da função, como mostra a figura 6.

Triggers

Um Trigger é um conjunto de comandos que é automaticamente executado quando um comando INSERT, DELETE ou UPDATE for executado numa tabela da base de dados.

Os Triggers são utilizados para realizar tarefas relacionadas com validações, restrições de acesso, rotinas de segurança e consistência de dados; normalmente os Triggers são adequados em determinadas situações:

- Mecanismos de validação envolvendo múltiplas tabelas;
- Criação de conteúdo de uma coluna derivada de outras colunas da tabela;
- Realizar análises e atualizações em outras tabelas com base em alterações e/ou inclusões da tabela atual.

A criação de um Trigger envolve duas etapas:

- Um comando SQL que vai disparar o Trigger (INSERT, DELETE, UPDATE);
- A ação que o Trigger vai executar.

```
CREATE TRIGGER PF.[TRG_LogData]  ON PF.Fatura FOR UPDATE
AS
BEGIN
    DECLARE @Data datetime2(7)
    SET @Data = GETDATE()
    INSERT INTO PF.Fatura(DataFatura)
    VALUES (@Data)
END
GO
```

Figura 7 – Trigger 1

Um trigger que dispara quando se cria uma fatura e que vai atualizar a data na fatura, como mostra a figura 7.

```
CREATE TRIGGER PF.[TRG_AdicionaMaisUm] ON PF.Produtos FOR Insert
AS
BEGIN
    DECLARE
    @CodigoProduto int,
    @CodigoEncomenda int,
    @Descricao varchar(30),
    @CodigoCategoria int,
    @CodigoOrigem int,
    @Preço int,
    @NumeroProdutos int

    SELECT
        @CodigoProduto = CodigoProduto
        ,@CodigoEncomenda= CodigoEncomenda
        ,@Descricao = Designacao
        ,@CodigoCategoria = CodigoCategoria
        ,@CodigoOrigem = CodigoOrigem
        ,@Preço = Preço
    FROM INSERTED

    SELECT @NumeroProdutos = COUNT(CodigoProduto) FROM pf.Produtos
    WHERE CodigoProduto = @CodigoProduto AND  @CodigoEncomenda= CodigoEncomenda AND @Descricao = Designacao
        AND @CodigoCategoria = CodigoCategoria  AND @CodigoOrigem = CodigoOrigem AND @Preço = Preço

    UPDATE PF.Produtos
    SET CodigoProduto = @NumeroProdutos + 1
    WHERE CodigoProduto = @CodigoProduto AND  @CodigoEncomenda= CodigoEncomenda AND @Descricao = Designacao
        AND @CodigoCategoria = CodigoCategoria  AND @CodigoOrigem = CodigoOrigem AND @Preço = Preço
END
GO
```

Figura 8 – Trigger 2

Um trigger que é disparado quando se insere um novo produto, usa-se a tabela INSERTED para guardar os valores, começa-se por fazer um count do CodigoProduto que é igual ao @NumeroProdutos e adiciona-se mais um, que é o novo produto, como mostra a figura 8.

```
CREATE TRIGGER PF.TRG__Validar ON PF.Categoria INSTEAD OF INSERT
AS
BEGIN
    /*tenho a informação introduzida na tabela inserted*/
    DECLARE @Designacao varchar(20)

    SELECT @Designacao = Designacao FROM inserted

    /*verificar se o valor de designação é válida*/
    IF @Designacao IN ('Produtos Básicos','Produtos de Limpeza','Produtos Higiene Pessoal',
    'Frutas e Legumes','Congelados','Charcutaria','Bebidas')
    /* se a designação for válida, fazer o insert*/
        INSERT INTO PF.Categoria(Designacao)
            SELECT @Designacao FROM inserted
    ELSE
        /*apresentar o erro */
        PRINT 'A designação inserida não é válida'
END
GO
```

Figura 9 – Trigger 3

Um trigger que valida a @Designacao introduzida na tabela categoria, se não coincidir com uma das pré-definidas, mostra uma mensagem de erro, como mostra a figura 9.

Stored Procedures

Stored Procedure, é um conjunto de comandos em SQL que podem ser executados como uma função. Um Stored Procedure armazena tarefas repetitivas e aceita parâmetros de entrada para que a tarefa seja efetuada de acordo com a necessidade individual.

```
CREATE PROCEDURE PF.Valor
AS
UPDATE PF.Venda
SET Valor = P.Preço
FROM PF.Venda AS V
INNER JOIN PF.Produtos AS P ON P.CodigoProduto = V.CodigoProduto
GO
```

Figura 10 – Stored Procedure 1

Foi criado um Procedure para fazer a atualização do valor de cada item quando se realiza uma venda.

```
CREATE PROCEDURE PF.ValorTotal
AS
    SELECT SUM(Valor) AS [Preço Total], CodigoFatura

    FROM PF.Venda
    GROUP BY CodigoFatura
RETURN 0
GO
```

Figura 11 – Stored Procedure 2

Foi criado um Procedure para o valor final quando se realiza uma venda.

```
CREATE PROCEDURE PF.ProdutoMaisCaro
AS
SELECT MAX(Valor) AS [Preço mais alto], C.Nome , CodigoFatura AS Venda
FROM PF.Venda AS V
INNER JOIN PF.Clientes AS C ON C.NIF = V.NIF
GROUP BY C.Nome, CodigoFatura
GO
```

Figura 12 – Stored Procedure 3

Foi criado um Procedure para saber o valor do produto mais caro da venda de cada cliente.

Diagrama Entidade Relação SQL

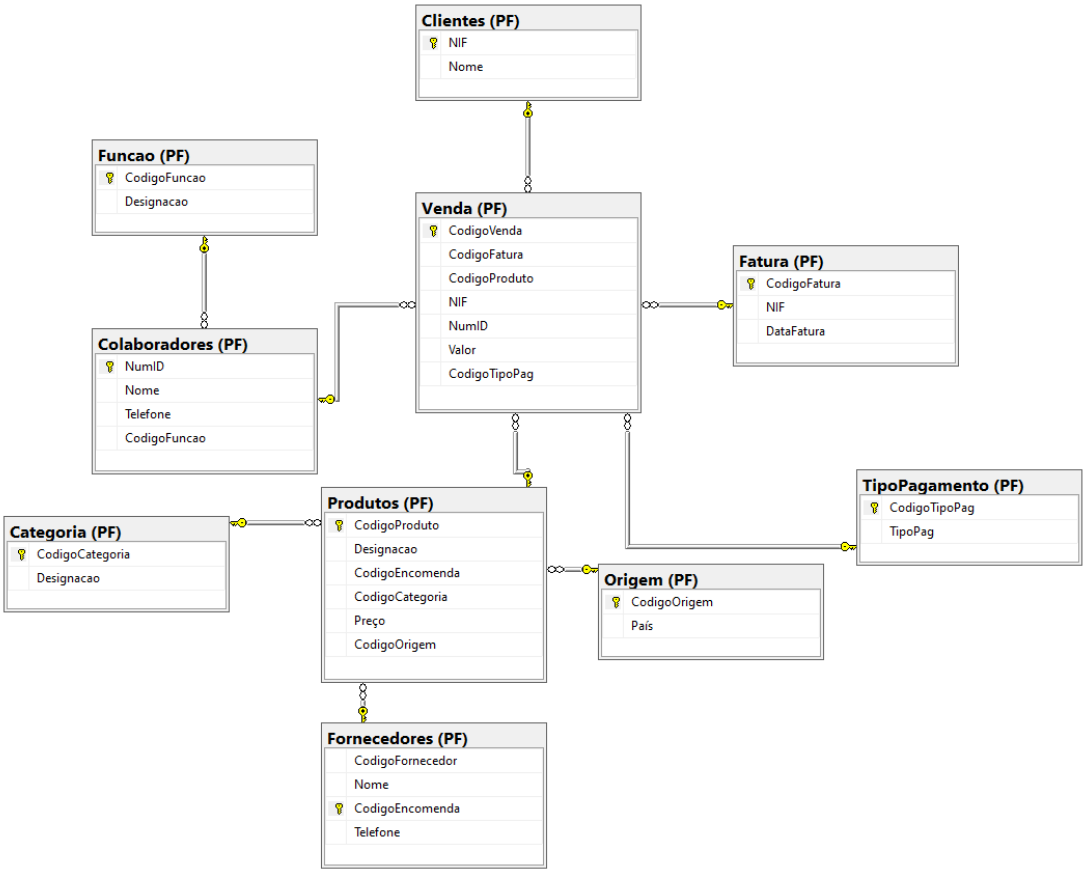

Figura 13 – Diagrama Entidade Relação