

Licenciatura Engenharia Informática

Ciência de Dados

Trabalho MSSQL

2º Semestre

Executado por

Hugo Lopes 2349

Bernardo Silva 2348

Orientado por

Ricardo Ferreira

Entregue em

22/04/2022

# Índice

1. Índice 2

2. Índice Figuras 4

3. Introdução 5

4. Raciocínio do Banco de dados 6

5. ENUM 8

6. Mascaras 9

7. Modelo entidade associação 10

8. Modelo relacional 11

9. Trigger 14

10. Conclusão 16

# Índice Figuras

[Figura 1-Modelo entidade associação 9](file:///D:\Download\Banco_de_dados.docx#_Toc101530789)

[Figura 1-Modelo entidade associação 9](file:///D:\Download\Banco_de_dados.docx#_Toc101530790)

[Figura 2-Modelo relacional (Tabelas "Utilizador", "Morada", "UtilizadoresDaConta") 11](file:///D:\Download\Banco_de_dados.docx#_Toc101530791)

[Figura 2-Modelo relacional (Tabelas "Utilizador", "Morada", "UtilizadoresDaConta") 11](file:///D:\Download\Banco_de_dados.docx#_Toc101530792)

[Figura 3-Modelo relacional (Tabelas "DividaDeUtilizador", "Divida", "Movimentos") 12](file:///D:\Download\Banco_de_dados.docx#_Toc101530793)

[Figura 3-Modelo relacional (Tabelas "DividaDeUtilizador", "Divida", "Movimentos") 12](file:///D:\Download\Banco_de_dados.docx#_Toc101530794)

[Figura 4-Modelo relacional (Tabelas "Perms", "UserPerms") 12](file:///D:\Download\Banco_de_dados.docx#_Toc101530795)

[Figura 4-Modelo relacional (Tabelas "Perms", "UserPerms") 12](file:///D:\Download\Banco_de_dados.docx#_Toc101530796)

[Figura 5-Modelo relacional (Tabelas "Conta") 13](file:///D:\Download\Banco_de_dados.docx#_Toc101530797)

# Introdução

Este trabalho foi realizado com o âmbito de solidificar os nossos conhecimentos na linguagem SQL e como parâmetro de avaliação na disciplina Ciência de dados, nele vamos explicar como fizemos um banco de dados para uma aplicação de um banco financeiro e explicar qual foi o nosso raciocínio para construir do mesmo.

# Raciocínio do Banco de dados

Começamos por anotar quais são os pontos principais que o banco deve ter, um banco tem Clientes, Trabalhadores, Contas e Movimentos entre contas.

De seguida começamos a pensar como transformar estes pontos em tabelas, bom começamos por criar uma tabela para os Clientes e trabalhadores, mas questionamo-nos os trabalhadores e os utilizadores não vão ter vários campos iguais? Com o objetivo de evitar redundância criamos apenas uma tabela chamada Utilizador. Durante a criação dos campos da mesma deparámo-nos com outro problema, vários utilizadores podem ter a mesma morada, ou seja, mais redundância, por isso criamos uma nova tabela chamada morada.

Depois de criar a tabela Utilizadores criamos a tabela Contas, nela iriamos ter os seguintes pontos principais, descrição da conta, o registo dos movimentos, quais são os seus utilizadores, as suas funções na conta e o montante que a mesma possui. Durante o desenvolvimento dela percebemos que iriamos ter uma ligação muitos para muitos com os Utilizadores, um utilizador pode ter muitas contas e uma conta pode ter vários utilizadores, além disso decidimos colocar nesta sub tabela as funções que cada user tem na conta.

Agora que criamos a tabela “Conta” vamos criar uma nova tabela para guardar os movimentos da conta, nela iriamos ter a conta remente, a conta destinatária, o utilizador que fez esse mesmo movimento, a data do movimento e o montante desse mesmo movimento.

Neste ponto chegamos a uma grande questão, como iriam funcionar os empréstimos/dividas dos clientes neste banco de dados? Bom pensamos em várias formas, depois de grande discussão de ideias achamos que o melhor método era criar uma nova tabela para a divida que iria ter uma ligação muito para muitos, ou seja, uma sub tabela para os utilizadores, porque vários utilizadores podem ter a mesma divida e uma divida pode ter vários utilizadores, a divida. Além disso tinha mos que criar um trigger para quando houvesse um novo movimento trigger o montante da divida tinha que ir diminuindo.

Mas o nosso banco de dados tinha uma última falha, não havia distinção entre trabalhadores e clientes, para isso criamos uma nova tabela, mas ainda encontramos outro tipo de problema, existem funcionários com as mais variadas funções, para os conseguirmos distinguir criamos outra tabela com as várias funções e uma sub tabela entre esta duas, porque elas têm uma ligação muito para muitos. Alem disso ainda tínhamos de guardar de alguma forma os movimentos em que os funcionários foram pagos os seus salários então criamos uma tabela com apenas com o identificador do movimento em que o funcionário foi pago e o identificador do funcionário, por fim, tínhamos que fazer um sistema para registar as horas de entrada e saída dos funcionários, então criamos uma nova tabela com a hora de início e fim que o trabalhador começou a trabalhar

# ENUM

Durante o desenvolvimento do projeto notamos que haveria campos deveriam ser ENUM. Nunca escrevemos nenhum código que os envolvesse, mas após uma breve pesquisa na documentação sobre a linguagem conseguimos perceber como fazer.

Quando criamos um novo campo devemos escrever no final CHECK( NomeDoCampo IN (‘Opeção1’, Opeção2’, …))

Exemplo: Distrito varchar(10) NOT NULL CHECK (Distrito IN('Lisboa', 'Porto', 'Faro'))

# Mascaras

Quando chegámos á parte de fazer mascaras surgiu um grande problema, a nossa única experiência com mascaras era na interface gráfica do Microsoft Access, não fazíamos ideia de como as escrever em código, decidimos então ver tutoriais no Youtube, eram muito confusos, então fomos ler a documentação do SQL no site da Microsoft, rapidamente percebemos como escrever o código.

No caso do campo de email já existe uma máscara pré-definida, então basta escrever o seguinte código MASKED WITH (FUNCTION = 'email()')

Enquanto nos outros campo tivemos de fazer uma máscara personalizada, com a seguinte sintaxe MASKED WITH (FUNCTION = 'partial(prefix,[padding],suffix)')

Exemplo pratico: Codigo\_Post Varchar(8) MASKED WITH (FUNCTION = 'partial(4,"xxxx-xxx",3)') Not Null,

Aqui temos o exemplo do código postal, na prática o que nos queremos é um resultado deste género: 2635-298. Para isso no prefix colocamos 4, ou seja, os primeiros quatro dígitos vão poder ser alterados, no sufix devemos colocar 3, para os últimos três dígitos poderem ser alterados, por fim no padding devemos colocar quadro letras ‘x’ que se vão referir na prática ao prefix, um ‘- ‘e mais três letras ‘x’ que estão se a referir ao sufix na prática ficando ‘xxxx-xxx’, ou seja, ‘2635-298’.

# Modelo entidade associação

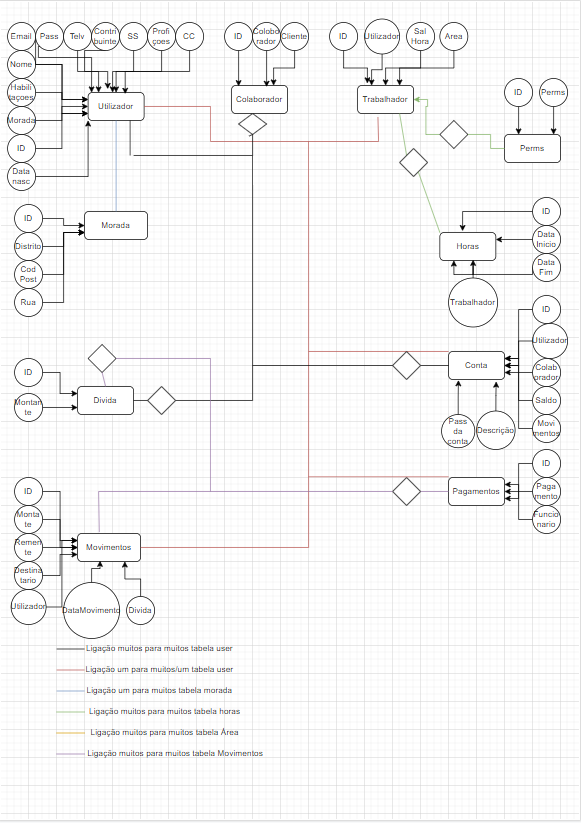
Este é o modelo entidade associação, aqui podemos ter uma visão geral das ligações, das tabelas e os seus campos nesta base de dados.

Figura -Modelo entidade associação

Figura -Modelo entidade associação

# Modelo relacional

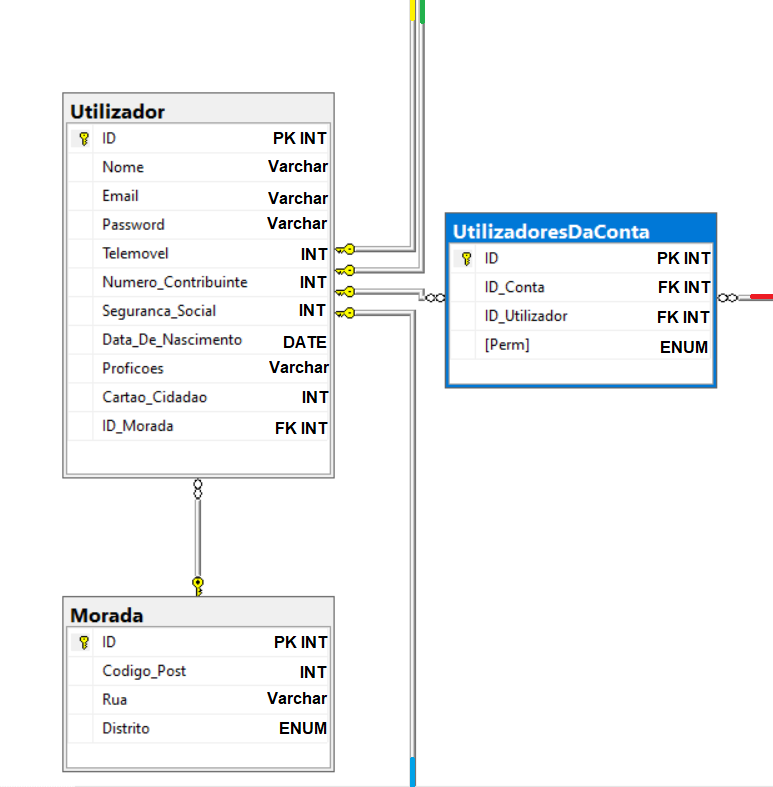
O Modelo relacional permite-nos saber quais a chaves primarias, estrangeiras e associações de cada tabela.

Figura -Modelo relacional (Tabelas "Utilizador", "Morada", "UtilizadoresDaConta")

Figura -Modelo relacional (Tabelas "Utilizador", "Morada", "UtilizadoresDaConta")

Na tabela “Utilizador” temos como chave primária o “ID”, como chave estrangeira o “ID\_Morada”, que faz a ligação desta tabela à tabela “Morada”, e o resto dos campos com informações a cerca do utilizador.

A tabela “UtilizadoresDaConta” é uma sub tabela criada porque a ligação entre a tabela “Utilizador” e “Conta” era uma ligação de muitos para muitos, e para evitar isto foi criada esta sub tabela, ela também vai ser a tabela do cartão, o número de cartão vai ser o ID, o utilizador do cartão vai ser o utilizador que corresponde ao “ID\_utilizador” e o pin da conta vai ser o campo “PassDaConta” na tabela “Conta”.

Na tabela “Morada” a chave primária é o campo “ID”, e o resto dos campos são informações da morada.

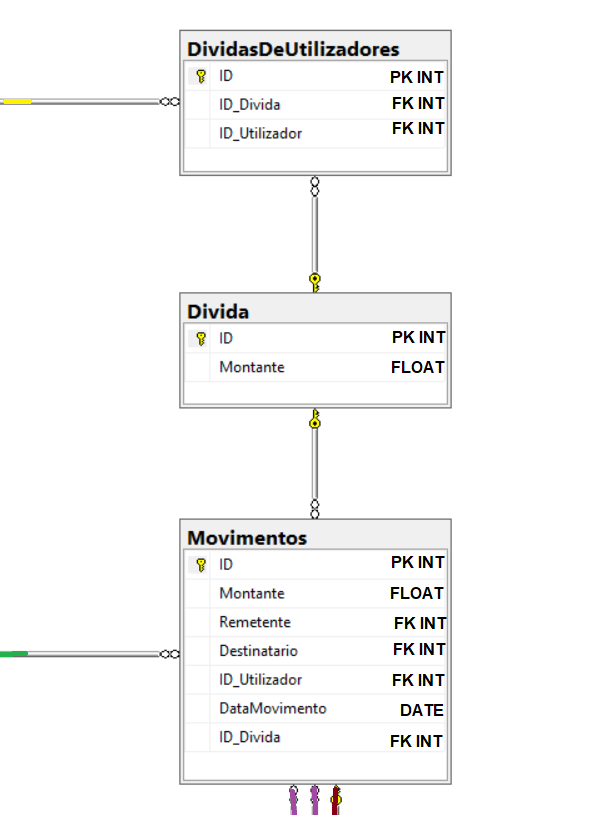
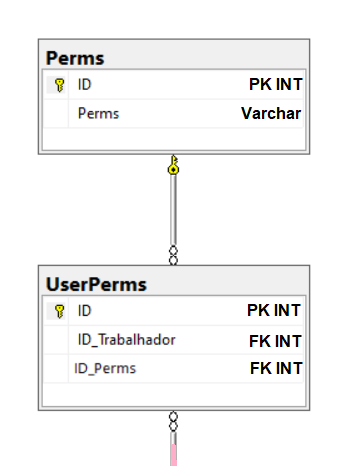
Aqui temos 2 tabelas e uma sub tabela, “DividaDeUtilizadores” que faz a ligação entre a tabela “Divida” e “Utilizador”, para saber quais a dividas de cada utilizador.

Figura -Modelo relacional (Tabelas "DividaDeUtilizador", "Divida", "Movimentos")

Figura -Modelo relacional (Tabelas "DividaDeUtilizador", "Divida", "Movimentos")

Figura -Modelo relacional (Tabelas "Perms", "UserPerms")

Figura -Modelo relacional (Tabelas "Perms", "UserPerms")

A tabela movimentos temos como chave primária a campo “ID”, temos também as informações necessárias para realizar um movimento, e as chaves estrangeiras “ID\_Utilizador” e “ID\_Divida”.

A tabela “Perms” é uma tabela que nos dá informação de quais permissões existem, e com a ajuda da sub tabela “UserPerms” podemos saber ou dar permissões a cada trabalhador.

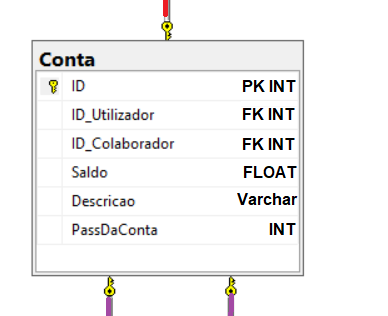
Nesta última tabela temos uma chave primária, “ID”, e duas chaves estrangeiras “ID\_Utilizador” e “ID\_Colaborador”, e os outros campos são as informações da conta.

Figura -Modelo relacional (Tabelas "Conta")

# Trigger

Decidi-mos usar triggers para atualizar os campos da nossa base de dados, apesar de nunca termos utilizado este comando decidi-mos que era a melhor maneira para atomizar a nossa base de dados. Com alguma pesquisa consegui-mos perceber como usar e como implementar na nossa base de dados.

CREATE TRIGGER Update\_Divida

ON Movimentos

FOR INSERT

AS

BEGIN

DECLARE @Divida INT,

@VALOR INT

SELECT @Divida = ID\_Divida, @VALOR = Montante from inserted

UPDATE Divida SET Montante = Montante - @VALOR

WHERE ID = @Divida

END

GO

Este trigger permite que assim que surge um movimento com o objetivo de abater o valor da divida, a divida na tabela “Divida” é atualizada.

CREATE TRIGGER Transferencia

ON Movimentos

FOR INSERT

AS

BEGIN

DECLARE @Remetente INT,

@Destinatario INT,

@VALOR INT

SELECT @Remetente = Remetente,@Destinatario = Destinatario,@VALOR = Montante from inserted

UPDATE Conta SET Saldo = Saldo - @VALOR

WHERE ID = @Remetente

UPDATE Conta SET Saldo = Saldo + @VALOR

WHERE ID = @Destinatario

END

GO

Este trigger permite que as transferências entre utilizadores atualizem os saldos das contas de ambos.

# Conclusão

Com este projeto solidificamos a nossa visão na maneira de criar uma base de dados apenas com a informações base das tabelas que eram necessários para a construção da mesma.

Começamos por desenhar a base de dados no papel e de seguida fazer os diagramas através do modelo entidade associação e modelo relacional, inicialmente menosprezamos estes diagramas, mas á medida que íamos fazendo a base de dados percebemos que eram essenciais para a correção de bugs, discussão de ideias e melhorias. Isto ajudou na criação da base de dados no Microsoft SQL Server, pois apenas tivemos de escrever o código e não tivemos de pensar muito em tabelas ou campos que estavam a mais. Decididamente que vamos utilizar este modelo em futuros projetos.

Encontramos algumas dificuldades, uma delas era a utilização de triggers, mascaras e enum, pois todas as vezes quem tínhamos usado MySQL nunca usamos estes mecanismos, estas dificuldades conseguiram ser superadas através de tutoriais do Youtube e documentação oficial da linguagem. Outra das dificuldades foi a utilização do IDE, pois como nunca o tínhamos usados, havia uma regularidade de bugs no programa que tínhamos que resolver como por exemplo reiniciar o mesmo, ir a lista de serviços do Windows para ligar o servidor de SQL, entre outros.

Na nossa visão este projeto está bastante completo e de fácil escalabilidade, mas uma das melhorias que possam ser feitas são adicionar mais dados aos enumerables, e ligar a base de dados a uma aplicação.