

Relatório

BDDAD



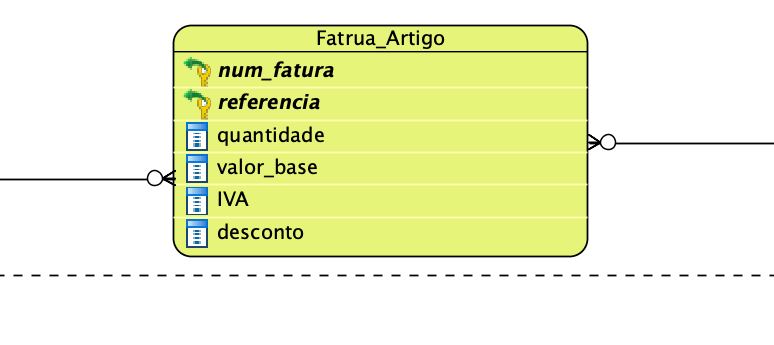
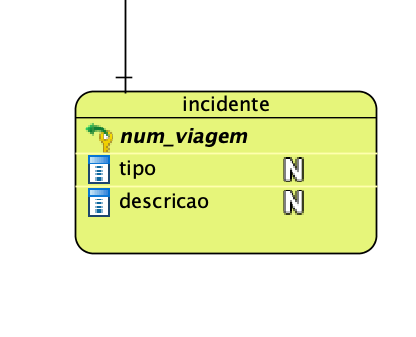
# Turma: 2DK José Mota 1161263

Patrick Timas 1171352

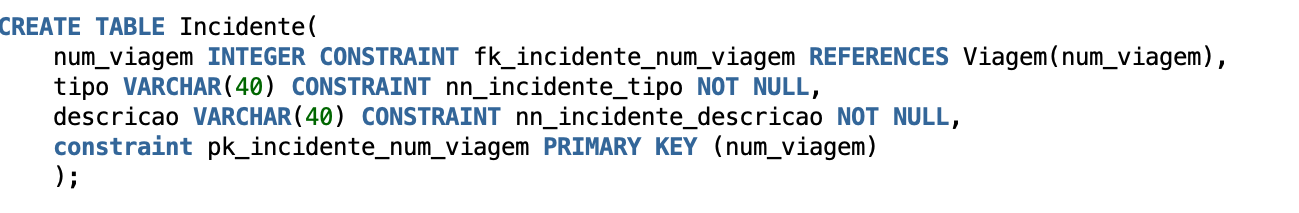
João Flores 1171409

2018/2019

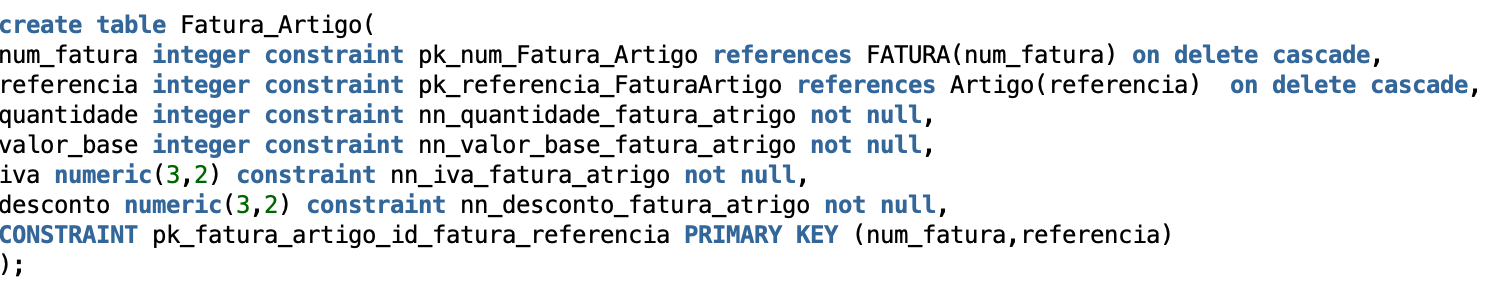
Para esta terceira parte do trabalho, para além de modificação das chaves primarias nas tabelas e adição de chaves compostas, apenas criamos duas tabelas novas: Fatura\_Artigo e Incidente.



A tabela incidente guarda o tipo de incidente e uma breve descrição do sucedido, para uma dada viagem.

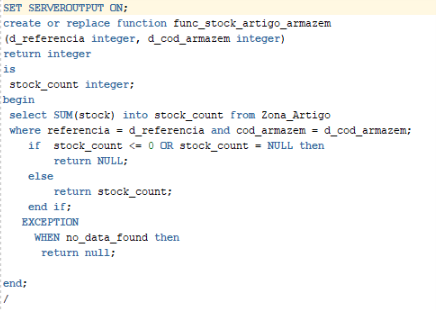


Já para a tabela Fatura\_Artigo, esta funciona como tabela intermediaria entre a fatura e o artigo. Esta contem o número da fatura proveniente da tabela fatura( sendo esta uma primary key) a referencia dos artigos proveniente da tabela artigo (sendo esta uma primary key). Esta tabela é, portanto, contituida por uma chave composta. É também constituída pelo valor base do produto pelo iva e pelo desconto.



Funções

**1. Implementar uma função designada func\_stock\_artigo\_armazem, para retornar a quantidade total de um artigo num determinado armazém. A função tem como parâmetros, o identificador do artigo e o identificador do armazém. No caso de o identificador do artigo ou do armazém não existir na BD, a função deve retornar o valor NULL.**



Para esta função, dado o código do artigo e o código do armazém proveniente, apenas temos de fazer um select count para o scock desse artigo nas zonas do armazém. Para tal através sabendo a referência do produto, o código do armazém conseguimos obter o stock de cada artigo em cada zona do armazém e retornamos esse valor, caso não se encontre o produto e/ou o armazém, é retornado null.

**2. Implementar uma função designada func\_razao\_veiculo\_transporte, para obter a razão entre o número de veículos que efetuaram viagens com o maior número de encomendas e o número de veículos que efetuaram viagens com o menor número de encomendas. O cálculo desta razão deve ser feito para um dado armazém num dado intervalo de tempo. A função tem como parâmetros, o identificador do armazém e o período de tempo (data início e data fim). No caso de o identificador do armazém não existir na BD, a função deve retornar o valor NULL.**



Função para obter a razão entre o número de veículos que efetuaram viagens com o maior número de encomendas e o número de veículos que efetuaram viagens com o menor número de encomendas. Com os parametros do armazem e um intervalo sendo este intervalo uma data inicio e uma data fim.

Quando um dos dados passados por parametro não consta nas tabela a função lança a exception que os dados não existem. Caso o máximo ou o minimo seja zero o utilizador tambem recebe a informação de dados insuficiente, isto acontece por não se puder dividir por zero.

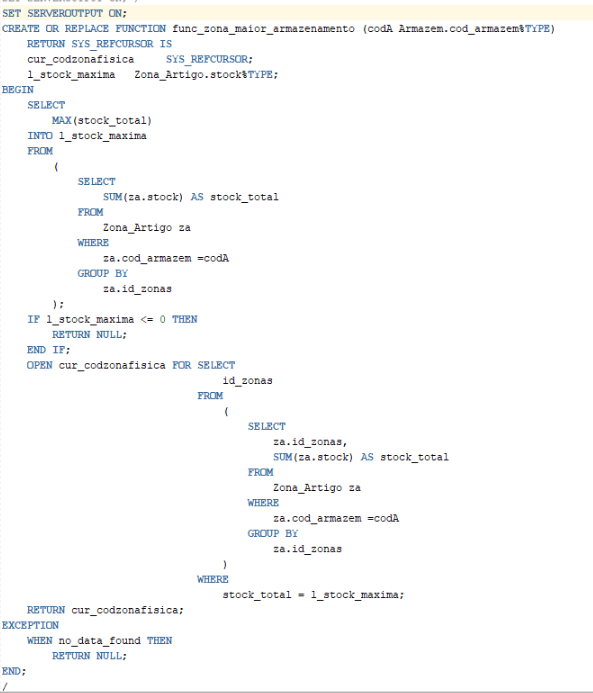
Para obter tanto o máximo como o minimo segue a mesma logica onde se junta as tabelas armazem, viagem e nota encomenda numa só que tenha data entre a data inicio e data fim, distingindo-se os dois select um por invocar o max outra o min.

Ao fazer-se o select count utliza-se o distinct para distinguir as matriculas em cada viagem e o count para as contar.

Após ter o máximo e minimo faz a divisão entre os dois returnando-se o valor obtido.

Para testar o falso um dos parametros não existe e a função retorna uma mensagem ao utilizador como era suposto.

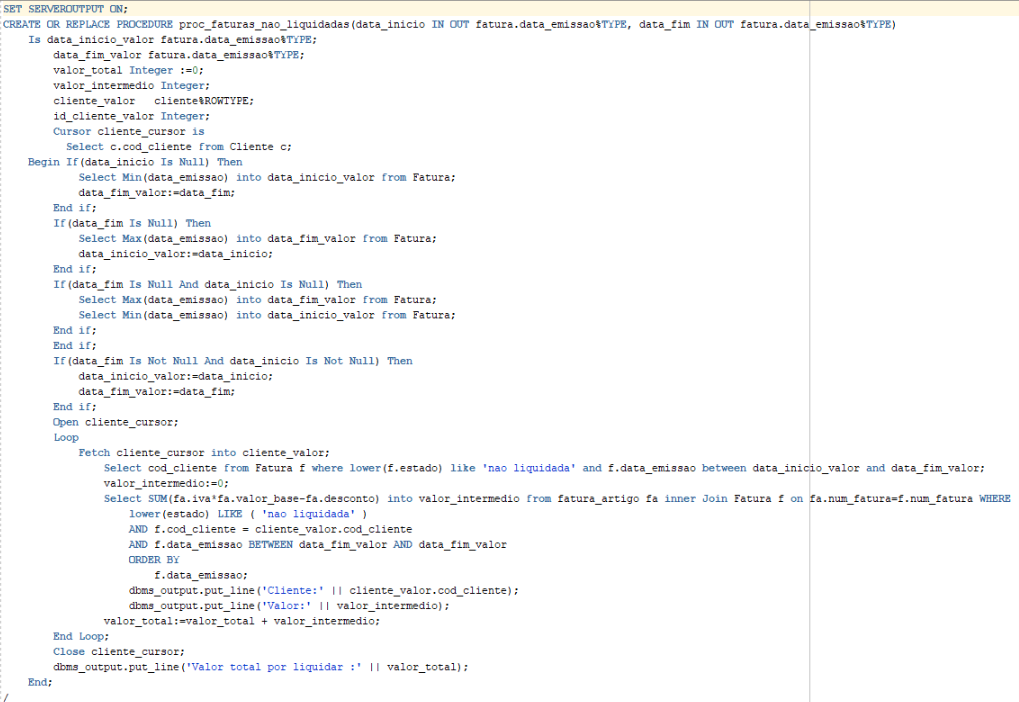
**3. Implementar uma função designada func\_zona\_maior\_armazenamento, para obter um cursor com os identificadores das zonas físicas de um dado armazém, que possuem a maior quantidade de artigos em stock. A função tem como parâmetro o identificador do armazém. No caso de o identificador do armazém não existir na BD ou a maior quantidade de artigos em stock for zero, a função deve retornar o valor NULL.**

****

A função recebe por parâmetro o codigo de armazém e através de um select vai somar o stock de cada artigo de uma zona física e guardar numa variável stock\_total através SUM. Ao fim de fazer a soma para todas as zonas do armazém indicado, usa-se um Select que vai retornar o valor máximo das somas feitas anteriormente através do MAX. O cursor vai fazer o mesmo feito anteriormente, mas no final através da condição WHERE ele compara as somas encontradas na procura recente, com o valor que esta na variável que se determinou anteriormente.

Procedimentos

**4. Implementar um procedimento designado proc\_faturas\_nao\_liquidadas, para realizar uma listagem com as faturas, de um dado período de tempo, que ainda não foram liquidadas. O procedimento tem como parâmetro o período (data de início e de fim) e no caso de não ser indicada uma das datas serão analisadas todas as faturas não liquidadas. Deve ser apresentada informação (identificador, data de faturação e valor total) das faturas em falta de cada cliente, ordenada pela data de faturação, bem como, o subtotal a liquidar por cliente. De igual forma deve ser apresentado o total de todas as faturas, de todos os que clientes, que faltam ser liquidadas à IsepBricolage.**

****

Com o procedure proc\_faturas\_nao\_liquidadas para realizar uma listagem com as faturas, de um dado período de tempo, que ainda não foram liquidadas. Dado que pode ou não ser passados parametros começa-mos por preencher esses valores. Caso não exista data inicio é atribuida a primeira data, caso não exista data fim fica com a ultima data nas listas, caso não seja passada nenhuma data mostra-se para todos e caso passe duas datas as tabelas tem de ser nesse intervalo.

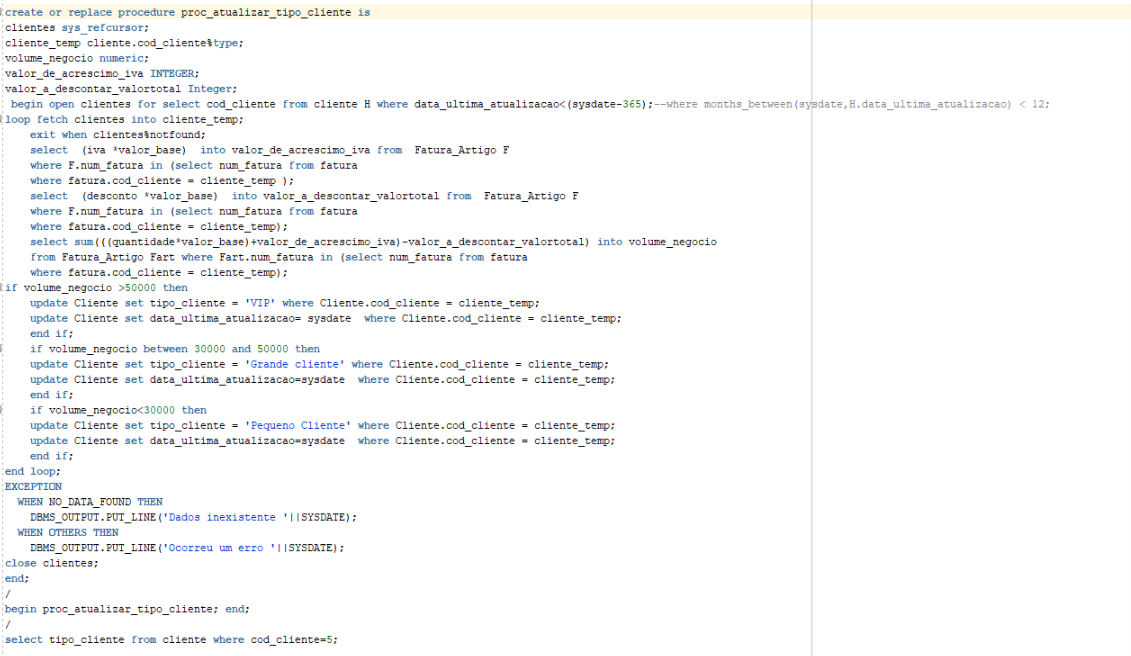
É colocado num cursor todas as chaves dos clientes que tenham faturas no estado ‘nao liquidadas’ e com datas entre o inicio e fim.

São adicionadas a uma variavel intermedia para ser sumada a uma final. Essa variavel intermedia é a soma de : 1+iva para o artigo vezes valor do artigo menos o desconto que estejam dentro dos parametros da data e do estado.

A cada iteração é apresentados o cliente o valor respetivo valor a liquidar.

No fim é apresentado o valor total que representa a soma de todos os valores a liquidar.

**6. Elaborar um procedimento designado proc\_atualizar\_tipo\_cliente, que atualize o valor do tipo de cliente, para todos os clientes, de acordo com o volume de negócio que efetuaram (ver Tabela 1). O período de volume de negócios a ser considerado é o dos 12 meses anteriores. No entanto, só os clientes que foram atualizados há mais de 12 meses é que podem ser atualizados. Se se entender necessário podem ser realizadas alterações ao modelo relacional**

****

Para este procedimento, precisamos de interagir com a classe cliente.

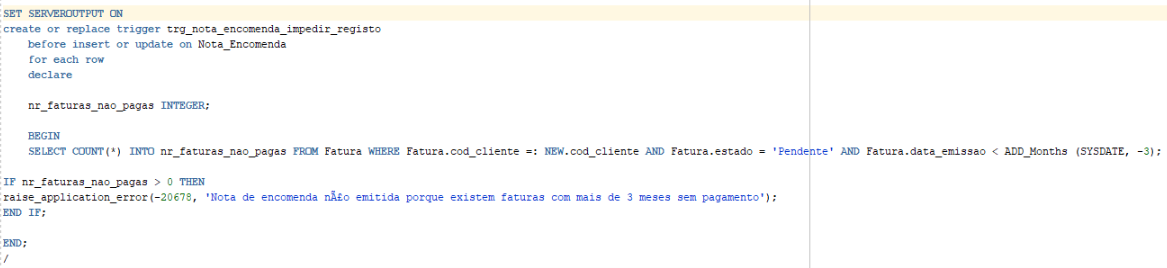
Criamos um cursor que ira conter os códigos dos clientes, cujo a sua classificação como tipo de cliente tenha sido atualizada apenas á mais de 1 ano. Com isto, para cada cliente vamos buscar o id das faturas do cliente em questão, e calculamos o valor de cada encomenda. Para tal ao valor do artigo(valor\_base) somamos o valor com o iva e no fim retiramos a quantia do desconto. No fim multiplicamos pelo pela quantidade do artigo. Consoante o valor obtido, mudamos a classificação do cliente:

* Se o valor for menor do que 30 000, será classificado como pequeno cliente;
* Se o valor estiver compreendido entre 30 000 e 50 000, será classificado como grande cliente;
* Se o valor for maior do que 50 000, será classificado como VIP

Do fim apenas temos de atualizar o tipo de cliente, bem como a data da última atualização, que será a data atual.

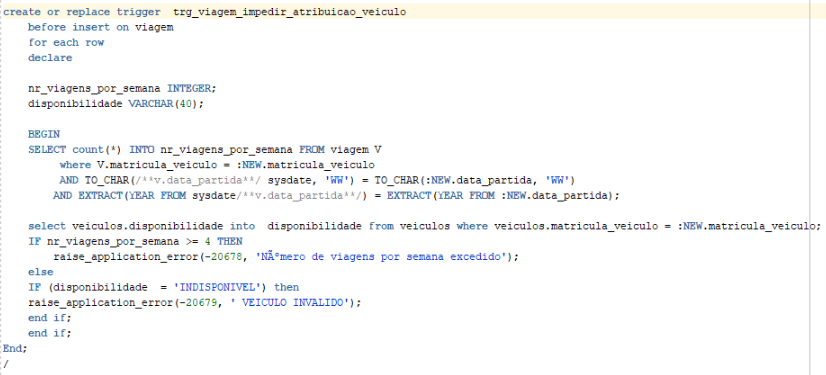
TRIGGERS

**7. Implementar um trigger designado trg\_nota\_encomenda\_impedir\_registo, que impeça que seja registada uma nota de encomenda de um cliente se este tiver faturas por pagar há mais de três meses.**



O trigger “trg\_nota\_encomenda\_impedir\_registo” tem como objetivo impedir o registo de uma nova nota encomenda de um determinado cliente que tem alguma fatura por pagar a mais de 3 messes. Para isto usou-se um Select que vai buscar todos os códigos dos clientes na tabela fatura e ve se é igual ao novo codigo ao cliente que queremos introduzir, se assim for, verifica-se o estado da fatura se e pendente ou pago, se for pendente o codigo verifica se o mesmo está pendente a mais de 3 messes. Se o resultado der positivo a nota encomenda e impedida de ser registada.

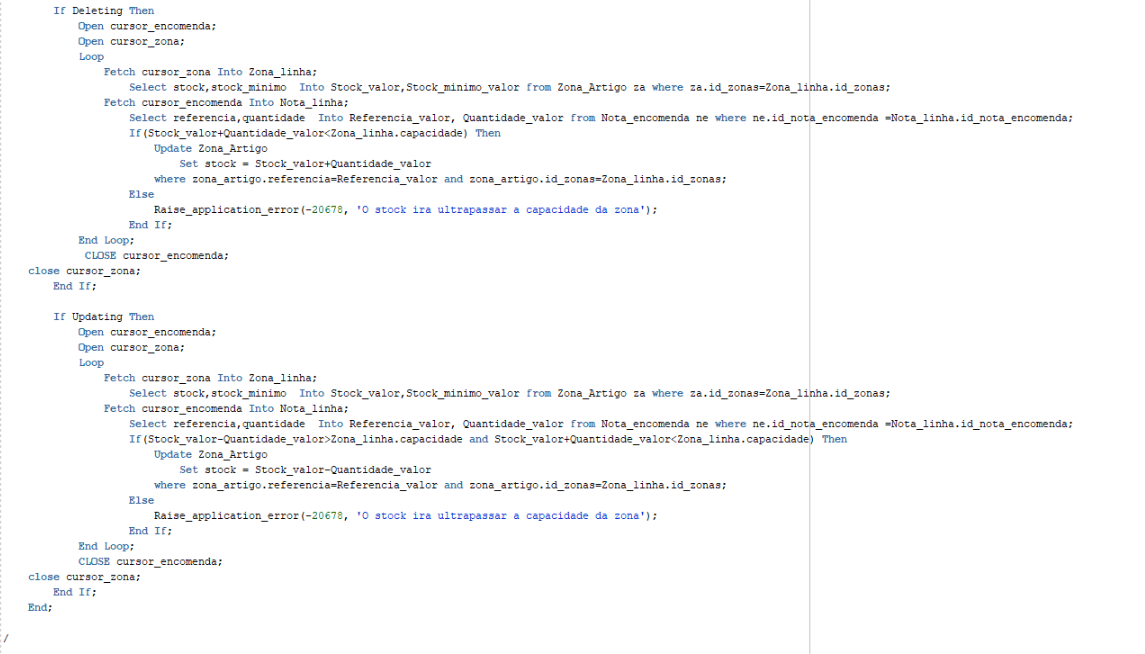
**8. Implementar um trigger designado trg\_viagem\_impedir\_atribuicao\_veiculo, que impeça que seja atribuído um veículo a uma nova viagem, se este já efetuou o número máximo de quatro viagens permitidas por semana ou, se está indisponível.**



Este trigger é disparado quando o veiculo se encontra, por algum motivo, indisponível ou porque no intervalo de 1 semana já realizou mais do que 4 viagens. Para tal para todos os veículos a serem inseridos, irmos contar o numero de viagens realizadas num prazo de uma semana, bem como a sua disponibilidade. Caso tenham sido realizadas mais de 4 viagens e se o veiculo se encontrar indisponível, será impedida a inserção do da atribuição do veiculo em causa para uma nova viagem.

**9. Implementar um trigger designado trg\_guia\_saida\_atualizar\_stock, que garanta que o stock é corretamente atualizado de acordo com a informação das guias de saída, ou seja, quando se cria, altera ou apaga uma linha de uma determinada guia de saída.**





Com o trigger trg\_guia\_saida\_atualizar\_stock pretende-se que garanta que o stock é corretamente atualizado de acordo com a informação das guias de saída, ou seja, quando se cria, altera ou apaga uma linha de uma determinada guia de saída.

Começa-mos por criar um cursor encomenda para guardar os dados necessarios da encomenda, outro cursor para guardar os atributos correspondentes à zona, e variveis para poder manipular os select e insert.

No insert abre-se os cursores e guarda-se nas variaveis stock stock minino e quantidade verifica-se o stock menos a quantidade pedida na encomenda é superior ao stock minino se sim insere-se os dados senão lança-se uma exception. No deleting verifica-mos se o stock mais a

quantidade que vai deixar de ser pedida numa certa encomenda não ultrapassa as capacidades da zona do armazem. No update faz-se ambas as verificações novamente caso falhe uma é lançada uma exception para ser mais facíl a compreensão do erro.