***Sistemas de Informação II***

Turma LI52D | Inverno 2017/2018



**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

***Relatório 1ºTrabalho***

***Sistema de Informação II***

**Filipe Fé 42141 | Rui Lima xxxxx**

**Docente:** Engenheiro Nuno Datia

# Descrição

Neste trabalho era pedido a realização de um modelo EA que desse suporte para a criação de um sistema de informação da empresa Glampinho que permitisse gerir os seus parques de campismo de luxo, respeitando as entidades apresentadas no enunciado juntamente com as restrições de integridade também nele referidas. Foi também pedido a criação de código T-SQL que ilustrasse as ‘’queries’’ pedidas em cada umas das alíneas e por fim testes que comprovassem o bom funcionamento das mesmas.

# Modelo Relacional

**Simples:**

* ParqueCampismo(nome [nvarchar(30)], morada[nvarchar(50)], estrelas[tinyint in(1, 2, 3, 4, 5)], email[nvarchar(30)])
* Extra(id[int], descrição[nvarchar(30)], preçoDia[int], associado[varchar(10) in(‘alojamento’, ‘pessoa’)])
  + O atributo associado indica se o extra é facturado por alojamento ou pessoa
* Hóspede(NIF[int], nome[nvarchar(30)], morada[nvarchar(50)], email[nvarchar(30)], númeroIdentificação[int])
* Estada(id[int], dataInício[DateTime], dataFim[DateTime])
  + data de início tem de ser menor que a data de fim

**Fracas:**

* Alojamento(nomeParque[nvarchar(30)], localização[nvarchar(30)], nome[nvarchar(30)], descrição[nvarchar(30)], preçoBase[int], númeroMáximoPessoas[tinyInt], tipoAlojamento[varchar(8) in(‘bungalow’ , ‘tenda’)])
  + nomeParque é chave estrangeira para ParqueCampismo.nome.
  + nome é chave candidata
* Bungalow(nomeParque[nvarchar(30)], localização[nvarchar(30)], tipologia[char(2)])
  + O par nomeParque-localização é chave estrangeira para Alojamento
* Tenda(nomeParque[nvarchar(30)], localização[nvarchar(30)], área[int])
  + O par nomeParque-localização é chave estrangeira para Alojamento
* Actividades(nomeParque[nvarchar(30)], númeroSequencial[int], nome[nvarchar(30)], descrição[nvarchar(30)], lotaçãoMáxima[int], preçoParticipante[int], dataRealização[DateTime])
  + nomeParque é chave estrangeira para ParqueCampismo.nome
* Factura(identificadorEstada[int], id [int], nomeHóspede[nvarchar(30)], NIFHóspede[int)
  + - identificadorEstada é uma chave estrangeira para Estada.id
    - NIFHóspede é uma chave estrangeira para Hóspede.NIF
    - nomeHóspede tem de ser o nome do hóspede associado ao NIFHóspede.
    - Telefones(nomeParque[nvarchar(30)], telefone[int])
  + nomeParque é chave estrangeira para ParqueCampismo.nome
  + Item(idEstada[int], idFactura[int], linha[int], quantidade[int], preço[int], descrição[nvarchar(30)])
  + O par idEstada-idFactura é chave estrangeira para Factura(idEstada, id)
  + quantidade indica quanto vezes foi esse item usufruído (número de pessoas \* número de dias)
  + preço indica o total a pagar para esse item, já calculado tendo em conta a quantidade
  + descrição é a descrição correspondente a esse item

**Associações:**

* Paga(nomeParque[nvarchar(30)], númeroSequencial[int], NIF[int])
  + O par nomeParque-númeroSequencial é chave estrangeira para Actividades
  + NIF é chave estrangeira para Hóspede.
* HóspedeEstada(NIF[int], id[int], hóspede[true,false])
  + NIF é chave estrangeira para Hóspede.NIF
  + id é chave estrangeira para Estada.id
  + hóspede identifica se o hóspede associado à estada é o responsável ou não.
* EstadaExtra(estadaId[int], extraId[int])
  + - estadaId é chave estrangeira para estada.id
    - extraId é chave estrangeira para extra.id
* AlojamentoEstada(nomeParque[nvarchar(30), localização[nvarchar(30)], id[int])
  + O par nomeParque-localização é chave estrangeira para Alojamento
  + id é chave estrangeira para Estada.id
* AlojamentoExtra(nomeParque[nvarchar(30)], localização[nvarchar(30)], id[int])
  + O par nomeParque-localização é chave estrangeira para Alojamento
  + id é chave estrangeira para Extra.id

# Restrições de integridade

# Alojamento é caracterizado por um nome único e por uma localização representada por um conjunto caracteres alfanuméricos únicos dentro de cada parque.

# Inicialmente foi definido como chave primária o conjunto nomeParque,nome,localização. Para respeitar a 3FN , foi retirado a atributo nome da chave primária de Alojamento, visto que basta o par localização,nome para identificar um Alojamento dentro do Parque de Campismo. Dessa forma o atributo nome passou a ser chave candidata pois, por ser único, também ele consegue identificar o alojamento dentro do parque.

# Qualquer alteração de preços de alojamento e de extras posterior a uma reserva ou início de estada não irá alterar o valor a pagar pelos hóspedes.

A nossa solução para esta restrição foi atribuir às relações entre entidades que tenham preços associados e a estada um atributo preço que especifica o preço definido na altura da reserva.

# Um hóspede, quando criado, tem de estar associado a uma estada num determinado Alojamento do parque.

É garantido que no momento a seguir à inserção do hóspede, este é associado a uma estada já existente se não for o responsável ou caso seja responsável, é criada a estada na altura.

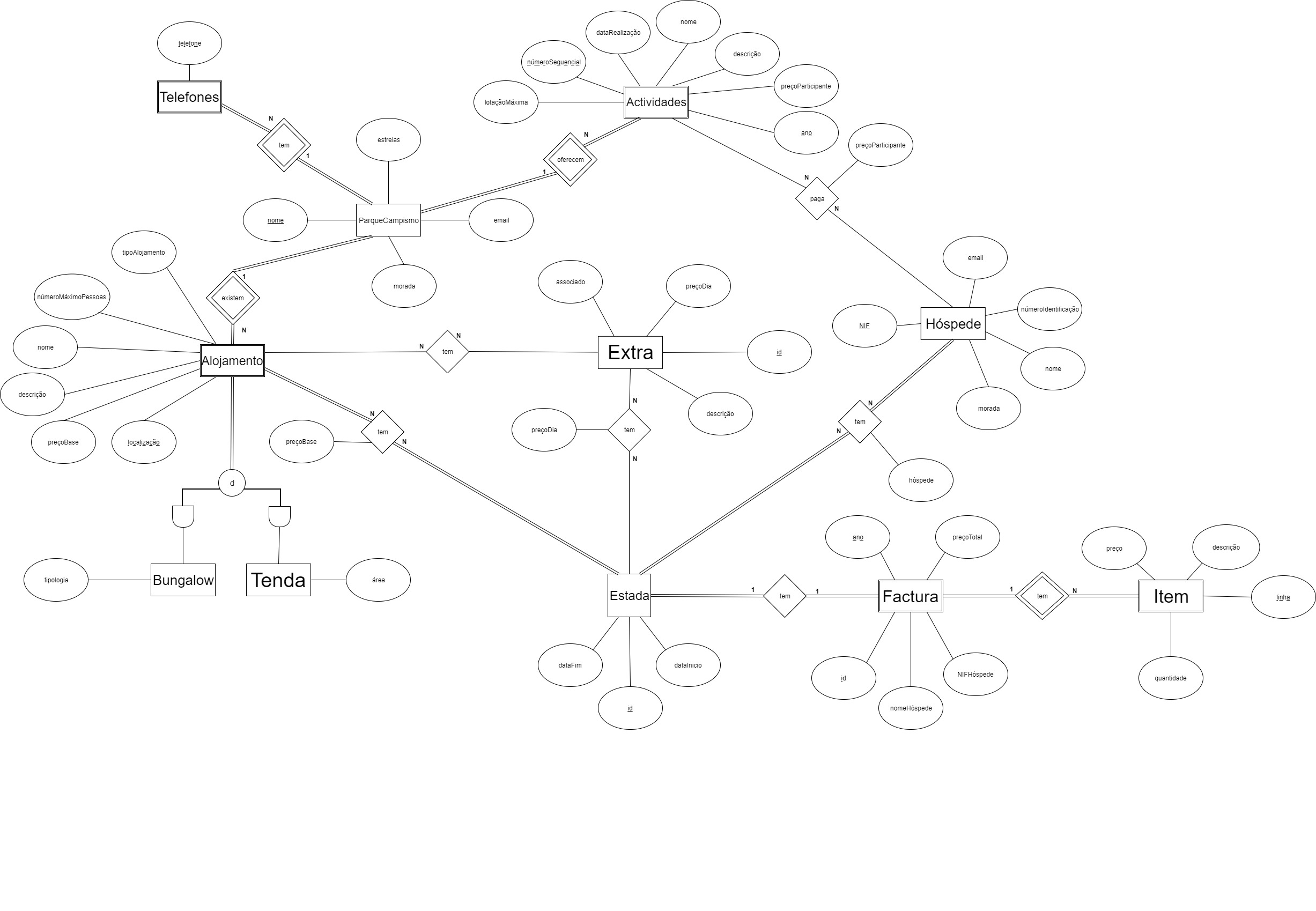
# Uma estada no parque fica associada a um hóspede responsável, e tem de ter associados um ou mais alojamentos e uma ou mais pessoas.

* ***Os mesmos hóspedes podem ficar alojados diversas vezes no mesmo alojamento, desde que em períodos diferentes.***
* ***Os extras para cada estada são definidos no momento do registo no sistema da estada.***

Desta forma, não é possível associar um extra novo a meio de uma estada. Por outro lado facilita a emissão da fatura para qualquer estada que tenha extras associados.

* ***A data de fim de uma estada é especificada no início de uma reserva.***
* ***Consequentemente, a fatura é emitida depois de criação da estada ser feita, tendo a ela associada todos os extras, atividades e alojamento usados na estada.***
* ***Uma fatura não pode ser apagada do sistema se ainda estiver entidades associadas à mesma ( hóspede responsável ,alojamentos, extras e atividades) ???***

Figura 1 – Modelo EA



# Resposta às alíneas do trabalho

1. ***Criação do modelo físico.***

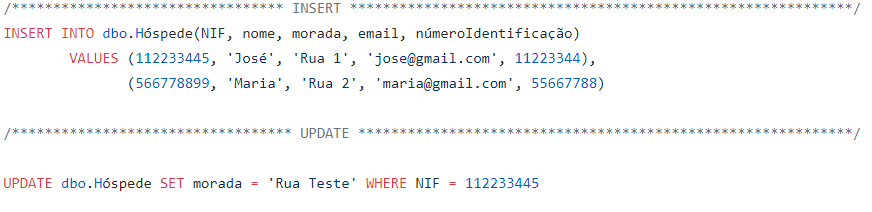
Foi realizado um script com a criação de todas as entidades necessárias, inclusive as associações entre elas, respeitando as restrições de integridade que são possíveis implementar em código SQL.

1. ***Remover o modelo físico.***

Foi realizado um script com os drops de todas as tabelas do nosso modelo de dados, com o cuidado de não apagar uma tabela que tenha dependência de terceiras.

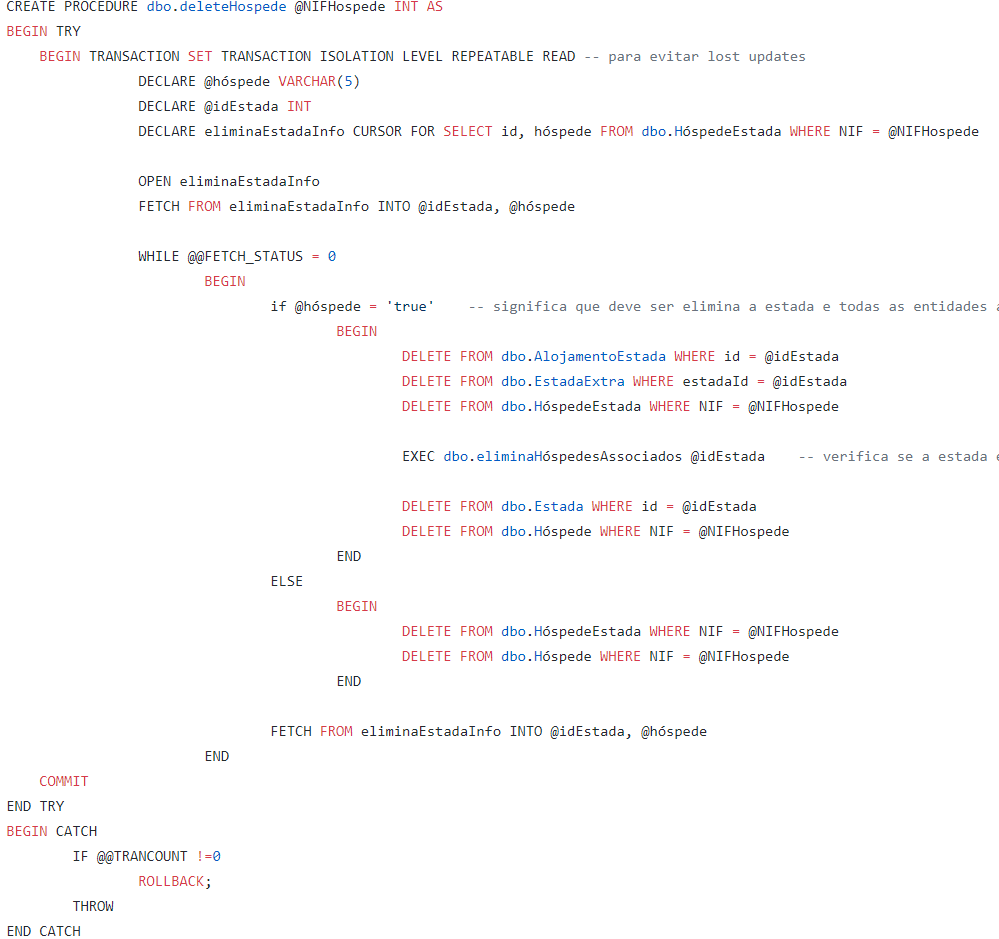
1. ***Inserir, remover e atualizar informação de um hóspede.***

Não foi necessário a criação de um procedimento armazenado para inserir e atualizar a informação de um hóspede, visto que trata apenas de uma instrução de INSERT/UPDATE na respetiva entidade.



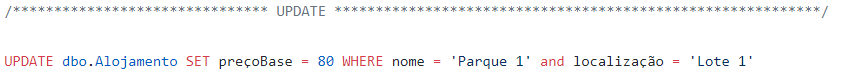
Para apagar um hóspede é necessário ter o cuidado de verificar se o hóspede em causa é o responsável pela estada. Se for, é necessário eliminar a informação relativa a esse hóspede nas tabelas que representam associações entre ele e a estada. Consequentemente a estada é também ela apagada do sistema para garantir o cumprimento da restrição de integridade que afirma que qualquer estada tem de ter um hóspede responsável associado. Dessa forma, todos os hóspedes que estavam alojados na mesma estada do responsável têm de ser apagados do sistema. Para isso é chamado o procedimento armazenado dbo.eliminaHóspedesAssociados .

O nível de isolamento desta transação foi definido como REPEATABLE READ de forma a evitar lost updates.



1. ***Inserir, remover e atualizar informação de um alojamento num parque.***

Não foi necessário a criação de um procedimento armazenado para atualizar a informação de um alojamento, visto que trata apenas de uma instrução de UPDATE na respetiva entidade.

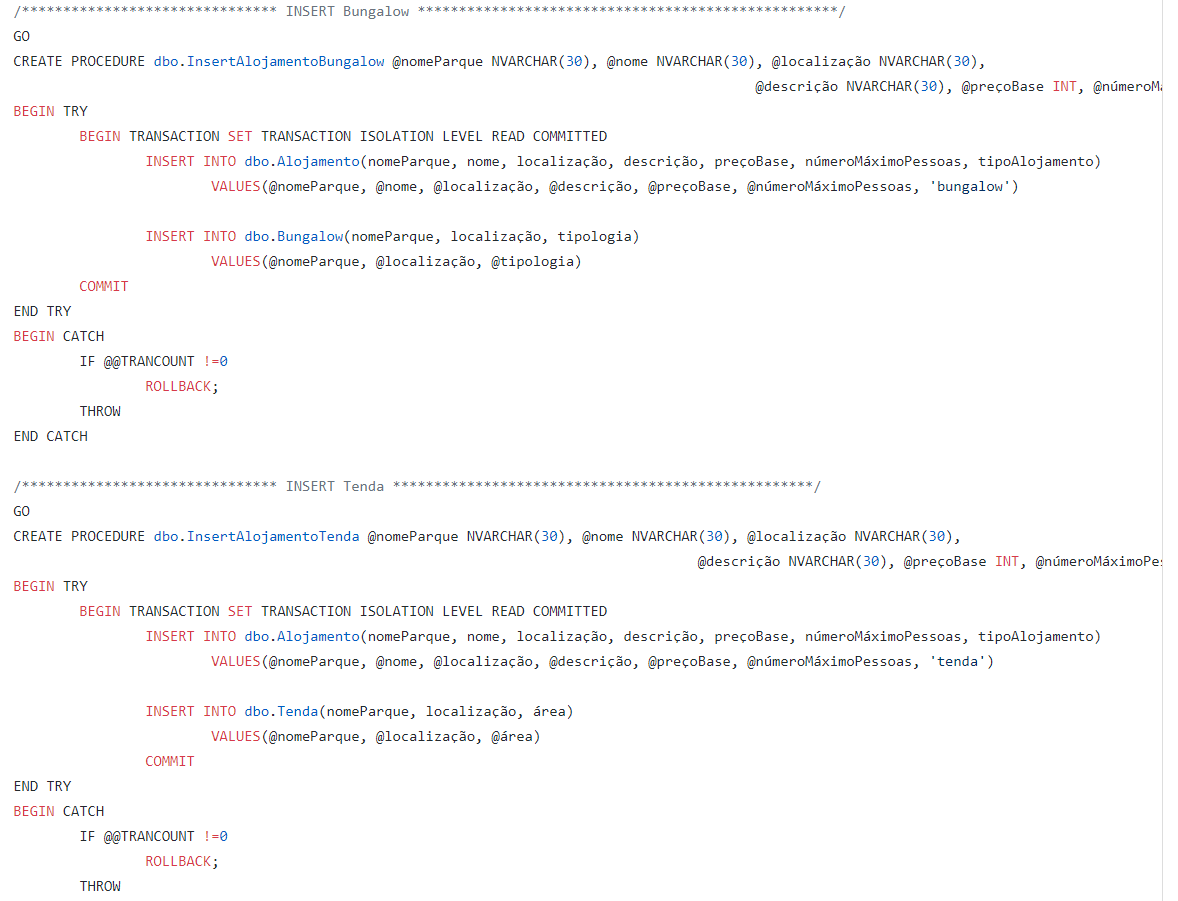


Para a inserção de um alojamento há que ter atenção os seguintes aspetos:

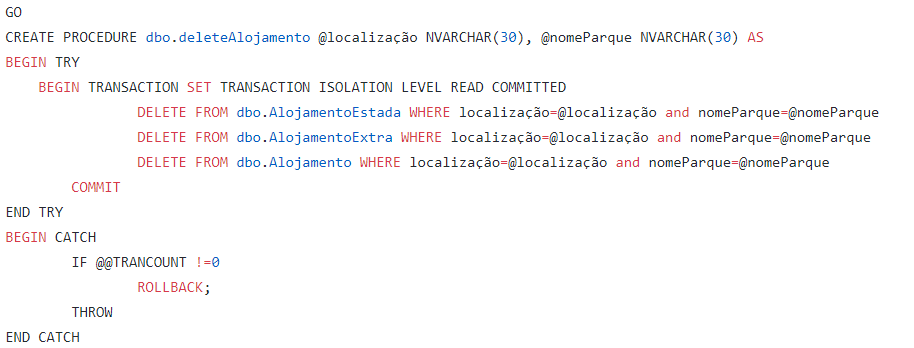
* Verificar qual o tipo de Alojamento a ser criado;
* Inserir na tabela correspondente ao tipo de Alojamento referido ( Tenda ou Bungalow );
* Adicionar o atributo área ou tipologia, consoante o tipo;
* Criar a entidade Alojamento respetiva.

Por essa razão foram criados dois procedimentos armazenados com o objetivo de distinguir o tipo de alojamento e dessa forma inserir na entidade respetiva. Ao invés de escolher um único procedimento armazenado que tratasse de todos dos aspetos referidos em cima e que recebesse como parâmetro os dois atributos área e tipologia, sendo que há partida um deles viria com o valor NULL, optamos por subdividi-lo em dois, para tornar o código mais legível.

Foi definido o nível de isolamento da transação para READ COMMITED porque apenas se tratam de inserções na base de dados e para garantir que se entretanto alguma transação for realizar uma leitura, o que está a ler foi commited na base de dados.

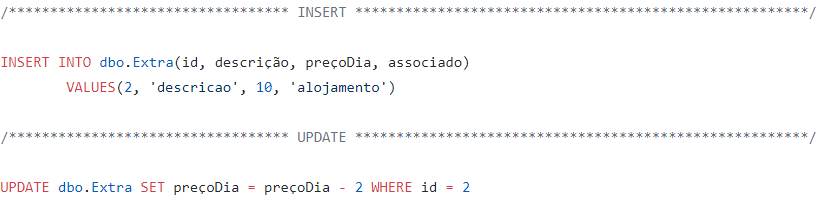


Para apagar um Alojamento é necessário apagá-lo primeiramente das tabelas que representam as associações entre outras entidades e o alojamento com o nome e nome do Parque passados como parâmetro, e por fim apagar da tabela Alojamento.

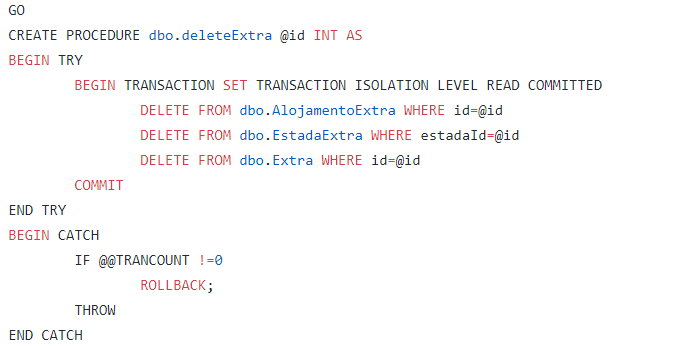


1. ***Inserir, remover e atualizar informação de um extra de alojamento.***

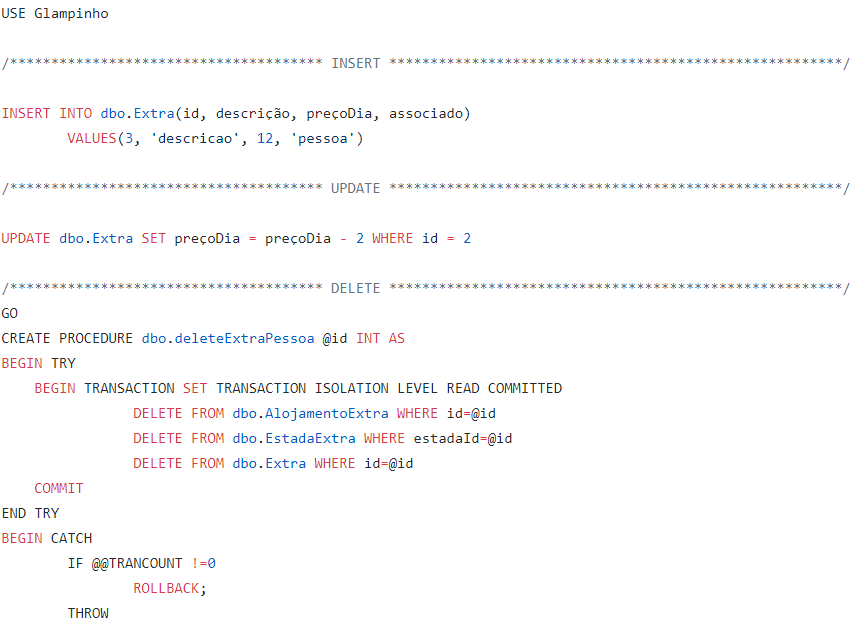
A inserção e atualização da informação de um extra , quer seja do tipo pessoal ou do tipo de alojamento é realizado da mesma forma: foi criado na entidade Extra um atributo “associado” que permite distinguir precisamente qual o tipo de extra a que nos estamos a referir. Posto isto, para inserir/atualizar é apenas necessário realizar uma instrução INSERT/UPDATE com o atributo associado especificado e os restantes que serão inseridos/atualizados.



Para remover é necessário primeiramente remover das entidades que se associam com Extra (Alojamento e Estada) os tuplos que tenham o mesmo id de Extra passado como parâmetro no procedimento armazenado de delete, e por fim apagar da tabela Extra.

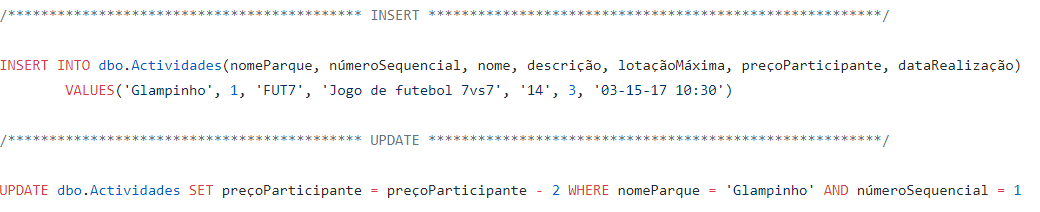


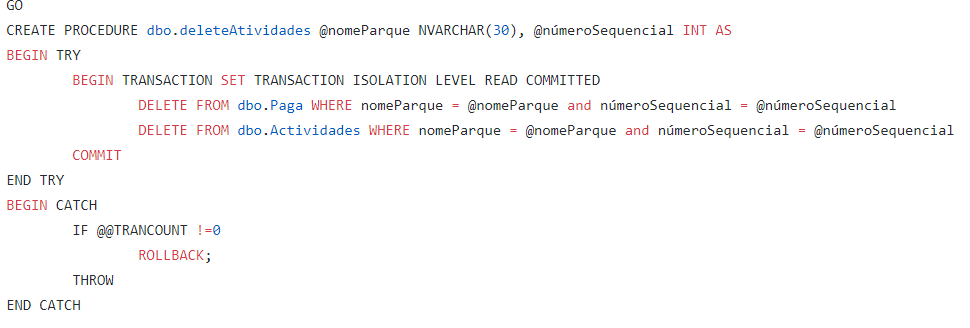
1. ***Inserir, remover e atualizar informação de um extra pessoal***.



1. ***Inserir, remover e atualizar informação de uma atividade.***

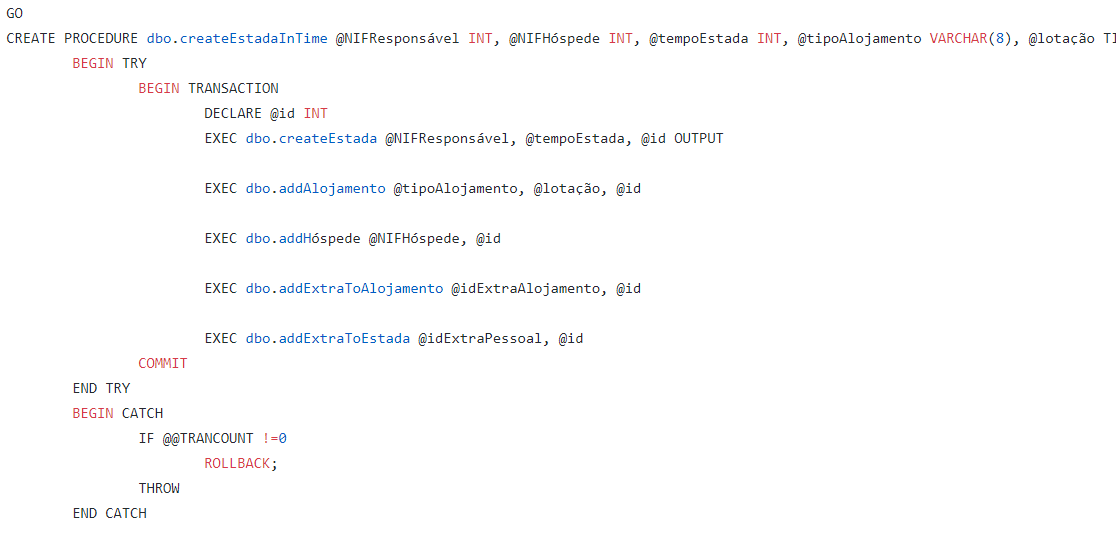
Não foi necessário a criação de um procedimento armazenado para inserir e atualizar a informação de um hóspede, visto que trata apenas de uma instrução de INSERT/UPDATE na respetiva entidade.



Para remover a informação da atividade é preciso remover primeiro da tabela Paga e só posteriormente remover da tabela Atividade. 

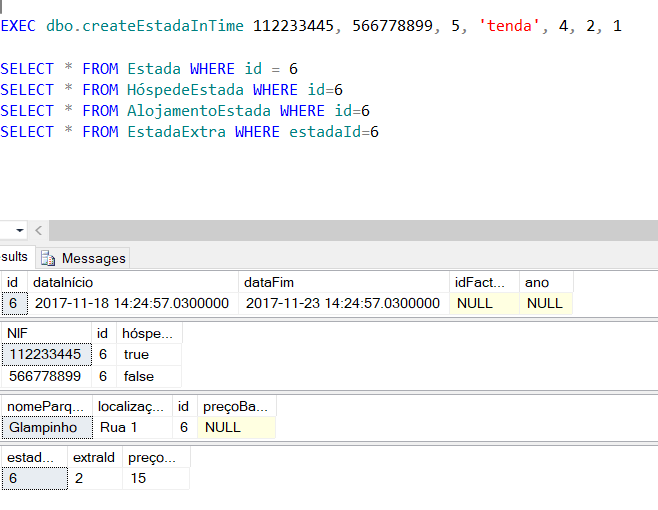
1. ***Criar uma estada para um dado período de tempo. Este processamento deve ser dividido nos seguintes sub-processamentos.***

A criação da estada foi dividida nos seguintes procedimentos:

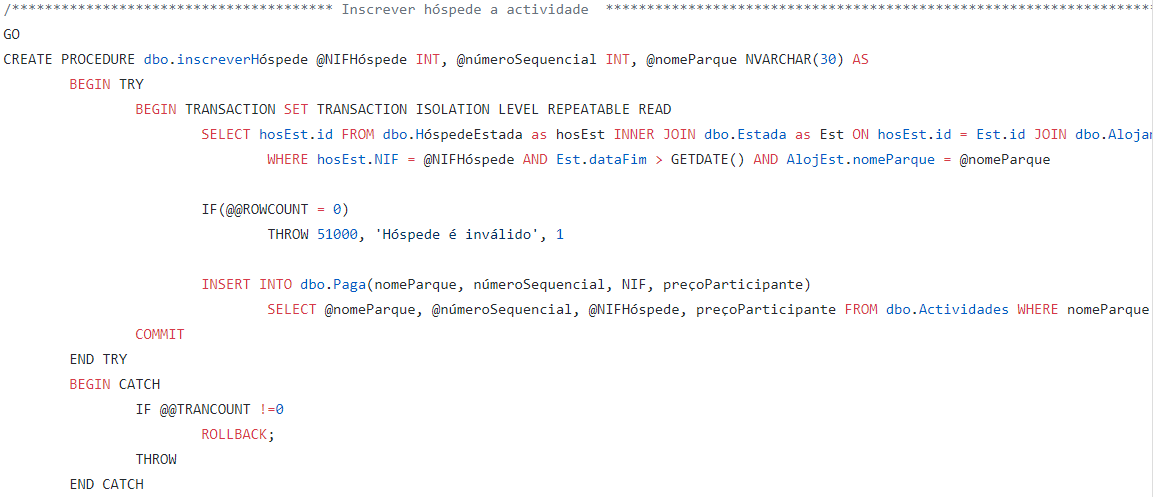


1. Criar uma estada , recebendo como parâmetro o NIF do responsável e a duração da estada. É inserido na tabela estada a data atual como data de inicio da estada, e soma-se a duração à data atual de forma a definir a data de fim. É verificado se já existe algum hóspede responsável pela estada em causa.
2. Adicionar um alojamento à esta criada anteriormente, desde que obedece às condições passadas como parâmetro, que neste caso é a lotação. Verificar se existem alojamento disponíveis para poder ser feita a reserva. Se sim, então é adicionado na altura os Extras desejados para aquela estada e é permitido avançar para o procedimento seguinte.
3. É adicionado um hóspede à estada criada.
4. Adicionados os extras para aquele tipo de Alojamento com o preço da altura.
5. Por fim é adicionado um extra pessoal e a transação é feita com sucesso.

***RESULTADO ESPERADO: ( nota: já existiam 5 estadas na base de dados, daí o id gerado ser o 6 )***

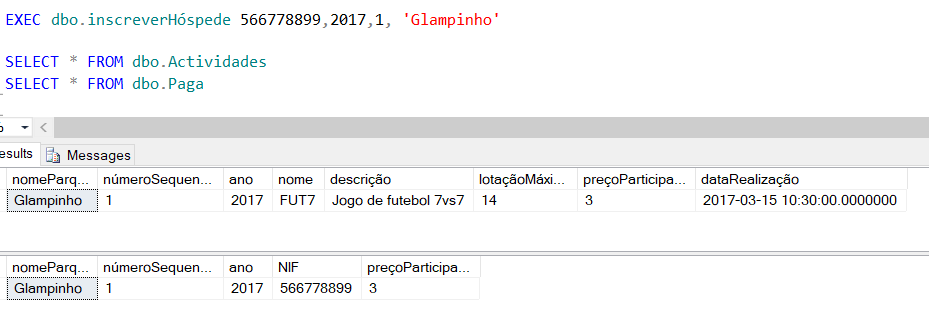


1. ***Inscrever um hóspede na atividade***

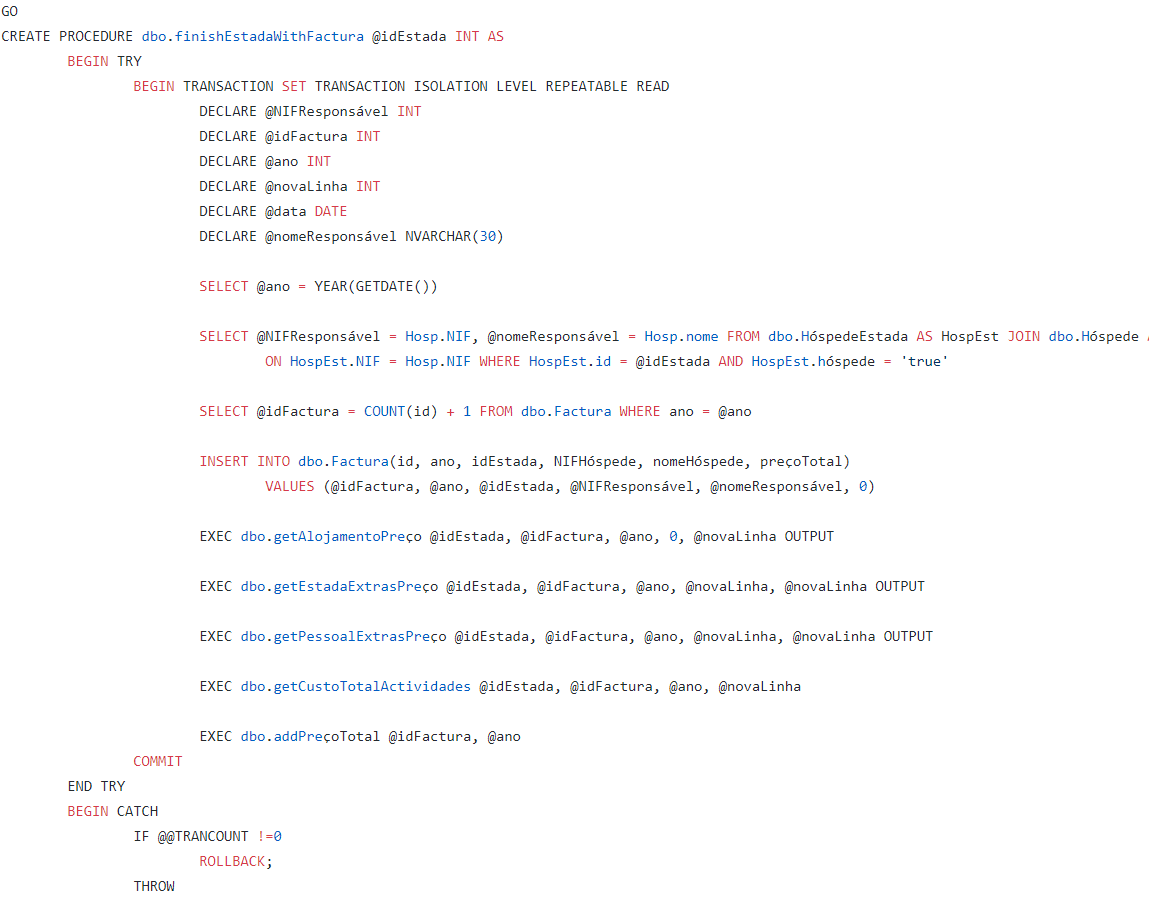


É feita a verificação se o hóspede que deseja inscrever-se numa atividade tem as condições necessárias para esse efeito, nomeadamente se o hóspede está alojado no parque passado como parâmetro. Para isso é verificado em todas as estadas daquele parque se existe algum cliente hospedado num alojamento com o NIF passado como parâmetro e se sim, verificar se a estada ainda está a decorrer. Caso o resultado seja positivo, então basta inserir na tabela Paga a informação de que determinado hóspede realizou a atividade com númeroSequencial X , sendo que o preço da altura era Y.

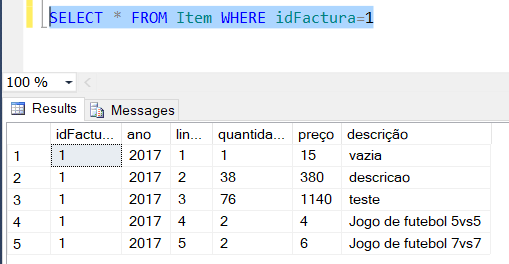
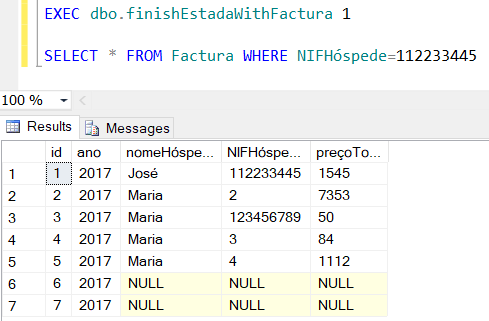
***RESULTADO ESPERADO:***



1. ***Proceder ao pagamento devido por uma estada, com emissão da respetiva fatura;***



***RESULTADO ESPERADO: (nota: pagamento da fatura da estada 1)***



????? EXPLICAR

1. ***Enviar emails a todos os hóspedes responsáveis por estadas que se irão iniciar dentro de um dado período temporal. Os emails dever ser enviados usando o procedimento armazenado SendMail que recebe o NIF do cliente e o texto da mensagem a enviar.***



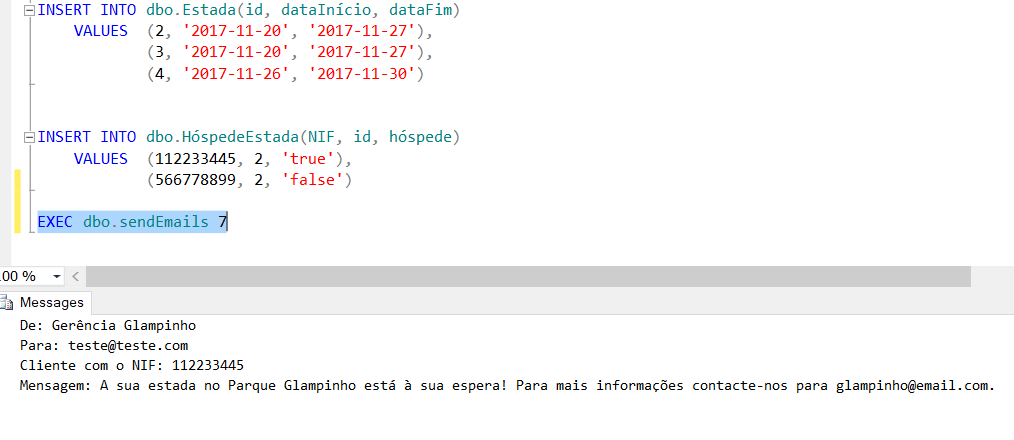
Foi criado o procedimento armazenado SendEmails que recebe um período temporal. Este parâmetro referido indica os dias que faltam para se iniciar as estadas dos hóspedes em causa.

Para essa verificação é feito um SELECT à base de dados na Estada de forma a ir buscar os clientes que se enquadrem entro de esse período temporal*(* ***Est.dataInício <= DATEADD(DAY, @periodoTemporal, GETDATE()*)** ).

Posteriormente é aberto um cursor para percorrer a tabela daí resultante de forma a enviar para cada um desses clientes o email desejado, através do chamamento do procedimento SendEmail.

O nível de isolamento é definido como REPEATABLE READ de forma a prevenir que entretanto ocorra alguma alteração na tabela fatura que modifique o valor da média.

***RESULTADO ESPERADO:***



1. ***Listar todas as atividades com lugares disponíveis para um intervalo de datas especificado;***

Foi criada a função listAtividades que é responsável por retornar uma tabela que contem apenas as atividades que se encontram disponíveis e dentro do intervalo passado como parâmetro.

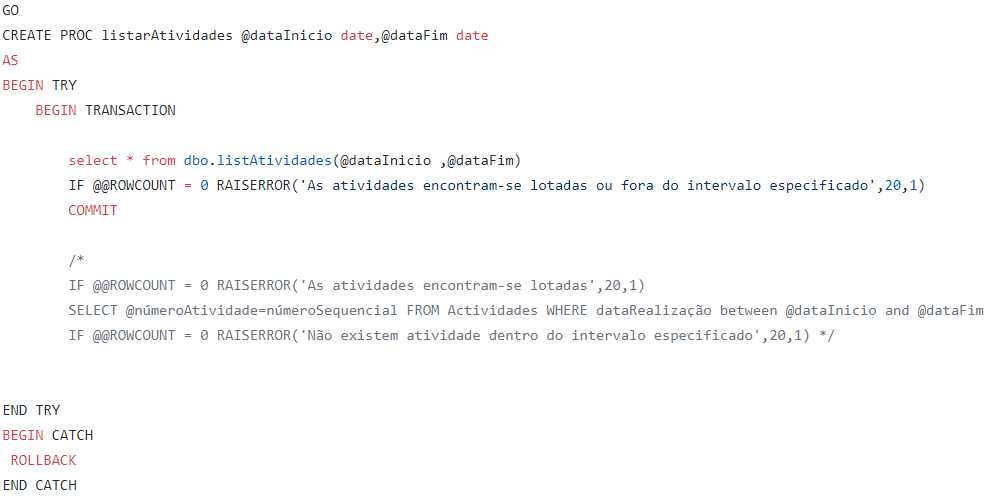
Para isso é feito um SELECT à base de dados na tabela Atividades de forma a ir buscar as atividades que tem uma data de realização dentro desse intervalo. De seguida é preciso contar o número de participantes dessas atividades de forma a garantir que têm lugares disponíveis para a eventual inserção de um cliente nessa mesma atividade.



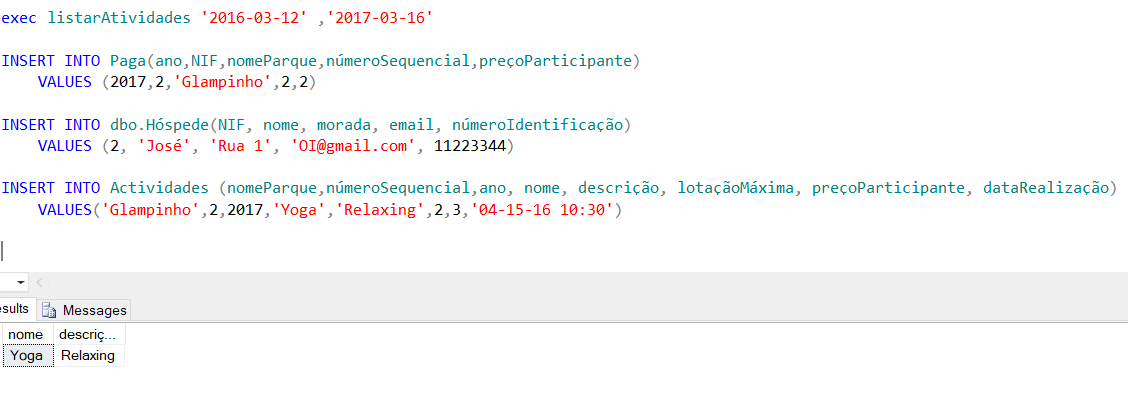
O nível de isolamento do procedimento é REPEATABLE READ apenas para prevenir que seja inserido alguma atividade enquanto a transação não tenha acabado. Dessa forma não corremos o risco de o resultado ter sido modificado

Por fim foi criado um procedimento armazenado que chama essa função.

O nível de isolamento deste procedimento armazenado é REPEATABLE READ para se ter a garantia que durante o intervalo entre primeiro select e o segundo não vá haver alterações naquilo que é apresentado, visto que entretanto podeira ocorrer uma inserção de um hóspede na atividade selecionada e perante essa situação estaríamos a perder um eventual update.



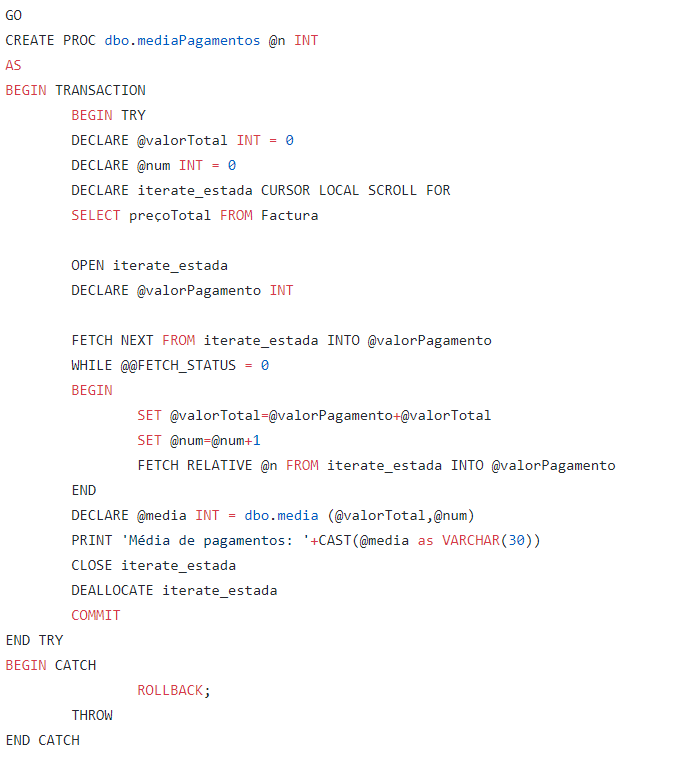
***RESULTADO ESPERADO:***



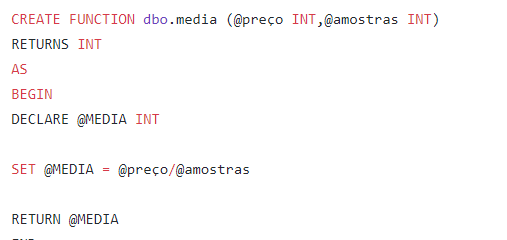
1. ***Obter a média dos pagamentos realizados num dado ano, calculada com um intervalo de amostragem especificado;***

Foi criado um procedimento onde é aberto um cursor que tem como objetivo iterar de N em N sobre a tabela Fatura, sendo esse N recebido como parâmetro. Para isso foi utilizado uma propriedade do cursor denominada RELATIVE que permite avançar para a “N rows” relativamente à coluna selecionada. É assumido que nunca é passado como parâmetro um N igual ou superior ao numero de amostras total na tabela Fatura. Foram também declaradas 3 variáveis:

* Uma para ir guardando o valor total dos pagamentos selecionados;
* Outra para guardar o valor atual do pagamento da Fatura selecionada.
* Uma variável para ir guardando o número de amostras selecionadas.

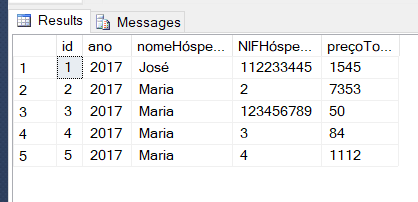


Depois de calculado o valor total de pagamentos é chamada a função media que tem como objetivo calcular a média de pagamentos.

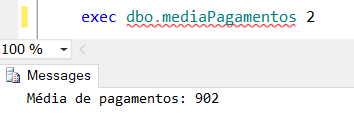


***RESULTADO ESPERADO:***

Faturas:



Média:



***(1545+50+1112)/3=902***

1. ***Criar a vista bungalows que permita executar as instruções SQL SELECT, INSERT, DELETE e UPDATE apenas sobre a parte dos alojamentos correspondente a bungalows. A vista deve produzir todas as colunas de um alojamento bungalow e todas as colunas de um parque. Os comandos INSERT, DELETE e UPDATE apenas alteram os dados relativos ao alojamento e não os relativos aos parques.***

Foram criados triggers para os INSERTS E DELETES visto que como a vista Bungalows é resultante da junção de várias tabelas, quando se insere/apaga algo da tabela é preciso inserir/eliminar também das tabelas que a vista depende. Por essa razão um trigger INSTEAD OF é útil na medida quem que quando alguém tentar alterar diretamente através da vista, é realizada a operação que queremos especificada no trigger.

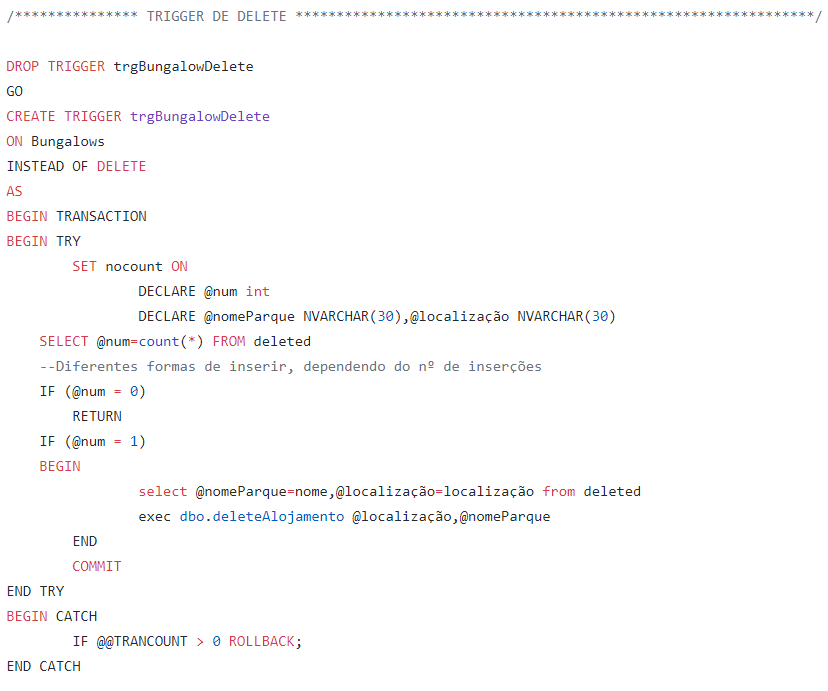
Ambos os triggers seguem a mesma lógica:

* Se não foi feito nenhum insert/delete, então o trigger não realiza nenhuma operaçãp;
* Caso tenha sido feito apenas um insert/delete é chamado o procedimento armazenado correspondente para inserir/apagar da tabela Alojamento e Bungalow.
* Se forem feitos mais do que um insert ao mesmo tempo, é preciso abrir um cursor para percorrer a tabela inserted. A cada iteração será inserido nas tabelas Alojamento e Bungalow os tuplos que o cursor iterou da tabela inserted, e assim sucessivamente até não existirem mais tuplos a inserir

***TRIGGER DE INSERT***



***TRIGGER DE DELETE***



Para o caso do update, como é dito no enunciado que apenas é alterado a informação relativa aos alojamentos Bungalows, o único atributo que pode ser mudado é a tipologia. Como esse atributo apenas aparece na tabela Bungalow, é possível realizar um update sem a utilização de um trigger, visto que apenas vai alterar o valor numa única tabela.

***EXEMPLO:***

