Enunciado - Parte 2

A segunda parte do trabalho prático (TP) de Bases de Dados tem como objetivo a implementação de algumas estruturas em PL/SQL. As estruturas a implementar consistem em funções, procedimentos e *triggers*, sobre a base de dados (BD) desenvolvida para a *SweetDreams*. Esta BD está de acordo com o modelo relacional da Figura 1 e os scripts para a criar estão disponíveis no *moodle*.

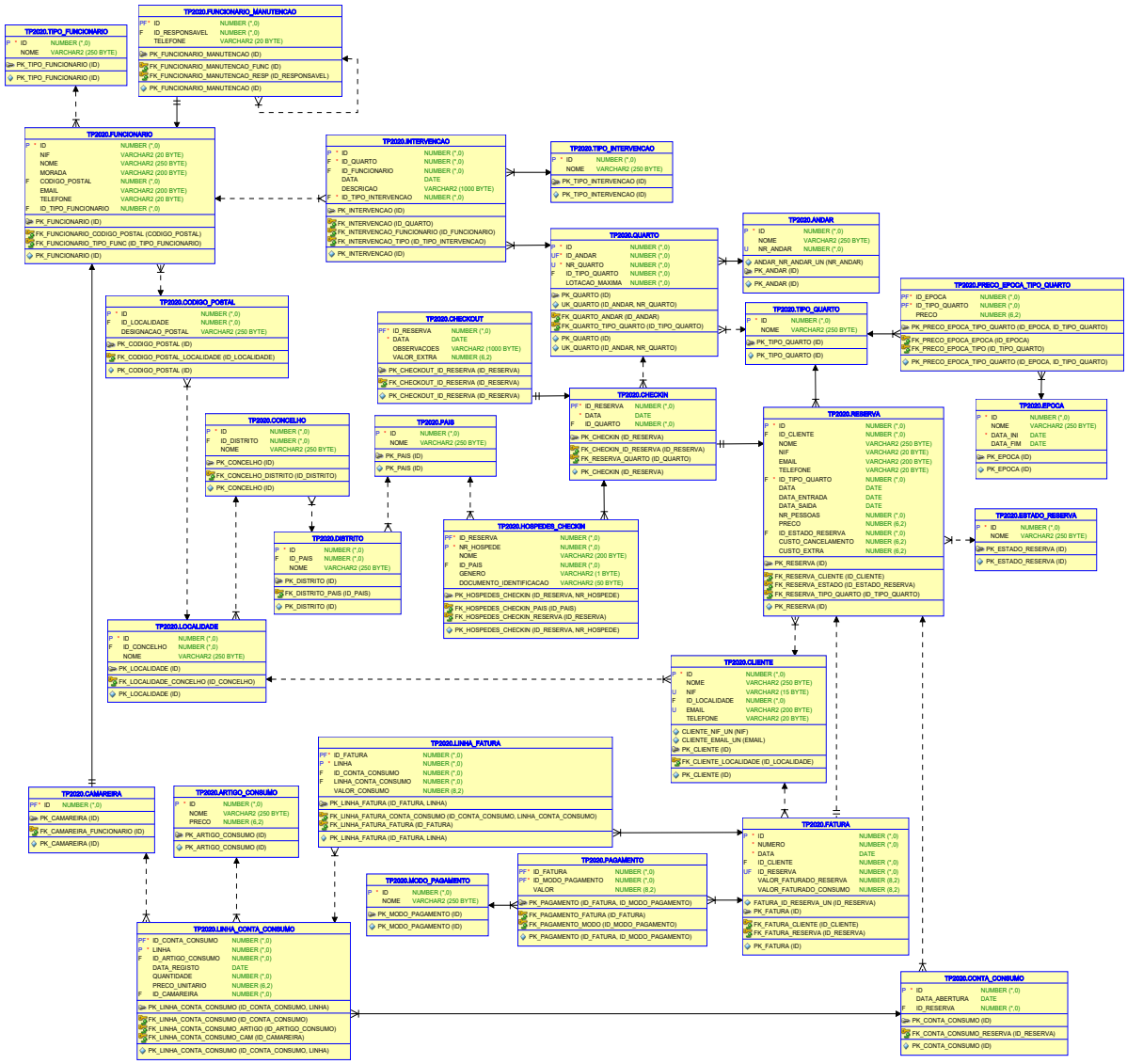

Figura1 - Modelo Relacional da *SweetDreams*

Nota: No moodle, na página da disciplina encontra-se uma imagem do modelo relacional.

Descrição das estruturas a implementar

Aluno A do Grupo

1. Implementar uma função designada **fncGetQuartoReserva** que, dado o ID de uma reserva, retorne o quarto a alocar. Havendo mais que um quarto, pretende-se o que estiver no andar mais baixo e, depois, que tenha tido menos dias de ocupação no último ano (para promover a distribuição da utilização pelos diversos quartos). No caso de o parâmetro ser inválido ou não haver quarto disponível, a função deve retornar o valor NULL através do mecanismo de exceções. O parâmetro é inválido quando: a) é null, b) não existe, c) a reserva já tem um quarto associado, d) o estado da reserva não é o adequado (e.g. está fechada). A função tem que tirar o máximo proveito do código SQL. **Testar** adequadamente a função implementada, através de **blocos anónimos**.

2. Implementar um procedimento designado **prcCheckOut** para efetuar o *check-out* de um quarto, incluindo a geração da fatura. O procedimento deve receber por parâmetro a linha da tabela reserva correspondente à reserva do quarto. Se o parâmetro fornecido for inválido, o procedimento deve levantar uma exceção com uma mensagem apropriada. O procedimento tem que tirar o máximo proveito do código SQL. **Testar** adequadamente o procedimento implementado, através de **blocos anónimos**.

3. Implementar um *trigger* designado **trgEpocasNaoSobrepostas** que garanta que a inserção/alteração de uma época não conduz a sobreposição entre épocas. **Testar** adequadamente o *trigger* implementado.

Aluno B do Grupo

4. Implementar uma função, designada **fncObterRegistoMensalCamareira,** que retorne um cursor com informação relativa aos consumos registados por cada uma das camareiras. A função recebe como parâmetro o mês, e opcionalmente o ano, e retorna um cursor com a seguinte informação: identificador único da camareira, nome da camareira, valor total dos consumos registados, data do primeiro registo de consumo, data do último registo de consumo e quantidade de dias em que não foram registados quaisquer consumos. Caso o valor do ano não seja passado por parâmetro, considere o ano anterior ao ano atual do sistema. A função tem que tirar o máximo proveito do código SQL. **Testar** adequadamente a função implementada, através de **blocos anónimos**.

5. Implementar um procedimento designado **prcAtualizarBonusCamareiras** que permita atualizar o valor de um bónus às camareiras de acordo com valor do total dos consumos registados num dado mês de um dado ano. O valor do parâmetro para o mês é obrigatório, mas o do ano é opcional, devendo, por omissão, ser considerado o ano atual

do sistema. O valor do bónus deve ser registado na tabela de camareiras e será determinado de acordo com o descrito na tabela 1. O procedimento tem que tirar o máximo proveito do código SQL. **Testar** adequadamente o procedimento implementado, através de **blocos anónimos**.

Tabela 1 - Bónus a atribuir

Critério	Bónus
>1000	15% do valor
<= 1000 e >=500	10% do valor
<500 e >100	5% do valor
<=100	sem bónus

Indique todas as alterações que têm que ser feita no modelo relacional.

6. Implementar um *trigger* designado **trgCorrigirAlteracaoBonus** para impedir que o valor do bónus das camareiras possa ser alterado sem garantir o seguinte: o bónus não pode ser diminuído e, no caso de ser aumentado, o aumento seja no máximo de 50%. **Testar** adequadamente o *trigger* implementado.

Aluno C do Grupo

7. Implementar uma função designada **fncDisponibilidadeReserva** que permita verificar a disponibilidade para uma possível reserva. A função deve receber, por parâmetro, o tipo de quarto, a data pretendida, a duração (dias), o número de pessoas e tem de retornar um valor booleano, true ou false. Se um dos parâmetros fornecidos for inválido, a função deve retornar o valor Null, usando o mecanismo de exceções. A função tem que tirar o máximo proveito do código SQL. **Testar** adequadamente a função implementada, através de **blocos anónimos**.
8. Implementar um procedimento designado **prcRegistarReserva** que permita registar uma reserva. O procedimento deve ter dois tipos de parâmetros de entrada: obrigatórios e opcionais. Os parâmetros obrigatórios são: o tipo de quarto, a data de entrada, a data de saída e o número de pessoas. Os parâmetros opcionais são: o id do cliente, o nome do cliente, o NIF, o telefone e o email. Considerar que quando se especifica o parâmetro id do cliente, não se pode especificar o nome, o NIF, o telefone nem o email e, quando não se especifica o parâmetro id do cliente, o nome torna-se obrigatório e o NIF, o telefone e o email são opcionais. O procedimento tem que tirar o máximo proveito do código SQL. **Testar** adequadamente o procedimento implementado, através de **blocos anónimos**.
9. Implementar um *trigger* designado **trgAtualizaCliente**, que permita atualizar o cliente na reserva assim que defina o seu id. **Testar** adequadamente o *trigger* implementado.

Informações sobre a entrega

Data limite para entrega: 15 de dezembro até às 23:59.

Observações:

O nome do ficheiro zip a submeter no Moodle deverá ter o seguinte formato:

TurmaXX_GrupoN_Parte2.zip onde,

XX - Representa a turma (e.g., TurmaDE)

N – Número do grupo (e.g., Grupo9)

Tenha em atenção que o ficheiro zip a ser entregue deve conter os seguintes ficheiros:

- um script para cada uma das questões implementadas, incluindo o código dos testes;
- apresentar o comprovativo de cada output dos testes realizados (screenshot dos resultados);
- conter o modelo relacional em formato pdf. Caso o modelo relacional tenha sido alterado, deverá indicar e justificar convenientemente todas as alterações efetuadas ao modelo.