

BASE DE DADOS



Resolução de um
Exame

Teóricas/Práticas
Ano Lectivo 2018/2019

3. Considere o seguinte esquema de uma base de dados relacional, que tem como objetivo a gestão do registo de cruzeiros efetuados por empresas.

Porto = { CodPorto, NomePorto, Pais }

Cruzeiro = { NrCruzeiro, NomeCruzeiro, Empresa, Preco }

Itinerario = { NrCruzeiro, CodPorto, NrParagem }

GuiaTuristico = { CodGuia, Nome, dataNascimento }

ServicoGuia = { Data, CodGuia, NrCruzeiro, CodPorto }

Escreva um procedimento em PL/SQL que receba como parâmetros de entrada um itinerário (assumido como existente) e uma data. O procedimento deve inserir os dados na tabela ServicoGuia atribuindo o primeiro guia turístico livre por ordem alfabética (assume-se que não há guias turísticos com o mesmo nome). O procedimento deve lançar uma exceção caso não exista nenhum guia turístico disponível para essa data.

```
create procedure proc_3_1(p_nrcruzeiro itinerario.nrcruzeiro%type, p_codigporto
itinerario.codigporto%type, p_data date) is
cursor c is
    select codguia from guia where codguia not in (select codguia from service_guia where
data = p_data) order by nome;
begin
    open c;
    fetch c into r;
    if c%found then
        insert into servico_guia(data, codguia, nrcruzeiro, codigporto) values (p_data,
r.codguia, p_nrcruzeiro, p_codigporto);
    else
        raise_application_error(-20000, '...');
    end if;
    close c;
end;
```

Escreva uma função em PL/SQL que receba como parâmetro de entrada o código de um determinado guia e retorne a maior diferença de dias entre duas datas consecutivas do respetivo Serviço de Guia (tabela ServicoGuia). A função deverá retornar NULL no caso do parâmetro fornecido ser NULL ou no caso de um código de guia não existente. **Deverá usar cursores explícitos.**

```
create or replace FUNCTION func_maior_diferenca(p_cod_guia servico_guia.cod_guia%TYPE)
RETURN NUMBER
AS
CURSOR c IS SELECT DISTINCT data FROM servico_guia WHERE cod_guia=p_cod_guia
ORDER BY 1 DESC;
v_data_1 DATE;
v_data_2 DATE;
v_dif    NUMBER :=0;
v_dif_max NUMBER :=0;
v_cont   NUMBER;
```

continua

```
BEGIN
OPEN c
FETCH c INTO v_data_1;
LOOP
  FETCH c INTO v_data_2;
  EXIT WHEN c%NOTFOUND;
  v_dif := v_data_1 - v_data_2;
  IF v_dif > v_dif_max THEN
    v_dif_max := v_dif;
  END IF;
  v_data_2 := v_data_1;
END LOOP;
v_cont := c%ROWCOUNT;
CLOSE c;
IF v_cont>1 THEN
  RETURN v_dif_max;
END IF;
RETURN NULL;
END;
```

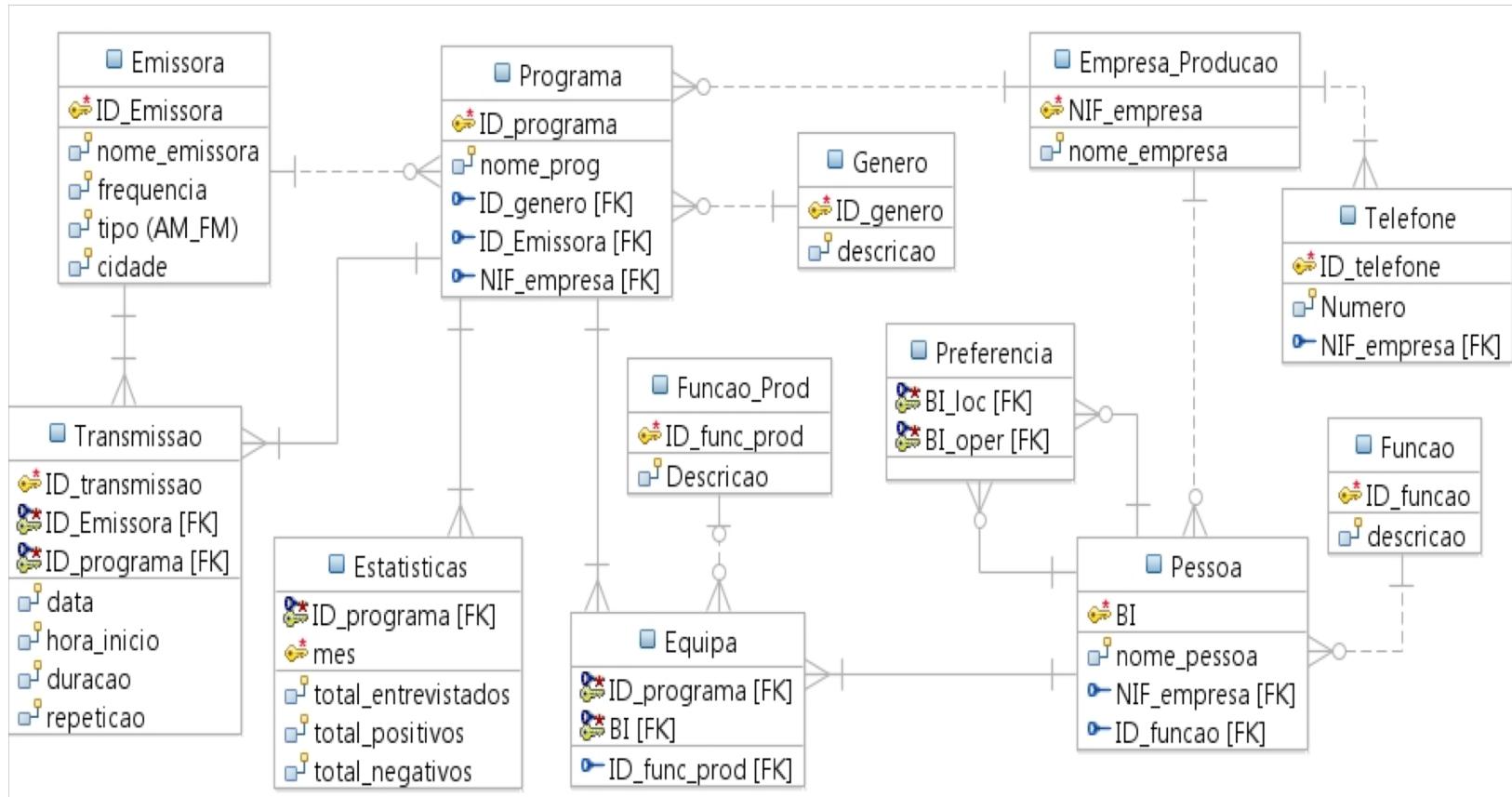
Escreva um trigger em PLSQL que não permita a atualização do preço de cruzeiros que não façam escala em portos da Grécia e cujo preço seja inferior a 1000.

```
CREATE OR REPLACE TRIGGER trg_imped_atual_preco_cruzeiro
BEFORE UPDATE OF preco ON cruzeiro
FOR EACH ROW WHEN (OLD.preco<1000)
DECLARE
cont NUMBER;
BEGIN
SELECT COUNT(*) INTO cont
FROM itinerario I, porto P
WHERE I.cod_porto=P.cod_porto AND I.nr_cruzeiro=:OLD.nr_cruzeiro AND
UPPER(P.pais)='GRECIA';
IF (cont=0) THEN
RAISE_APPLICATION_ERROR(-20100, 'AtualizaÁ,o do preÁo do cruzeiro ' || :OLD.nome_cruzeiro || ' È
inv·lida.');
END IF;
END;
```

2. Considere o seguinte problema:

Uma empresa decidiu construir uma base de dados para armazenar informação sobre a programação de várias emissoras de rádio com base nos seguintes requisitos.

1. Existe um conjunto de emissoras de rádio, caracterizadas por: nome, frequência de transmissão, tipo de transmissão (AM-FM) e cidade de localização.
2. Cada emissora de rádio, emite vários programas que são identificados por um nome e um género (jornalístico, musical, desporto, cultural, etc).
3. Os programas podem ser transmitidos por diferentes emissoras de rádio.
4. Cada emissora de rádio pode emitir cada programa mais do que uma vez. Para cada emissão regista-se: a data, a hora de início, a duração e se é uma repetição ou não.
5. Sobre cada programa sabe-se que é produzido por um único consórcio.
6. Cada consórcio é composto por uma das emissoras de rádio que o transmite e uma empresa de produção.
7. Das empresas de produção conhece-se o seu número de contribuinte, nome, e os seus telefones.
8. Em cada empresa de produção trabalham pessoas relacionadas diretamente com os programas, das quais se conhece o número do bilhete de identidade e o seu nome.
9. Estas pessoas podem ser locutores, operadores ou produtores.
10. As empresas de produção exigem exclusividade aos seus funcionários.
11. Para cada programa interessa registar quem são os seus locutores, os produtores e os operadores.
12. Os produtores podem exercer a função de produção jornalística ou comercial em cada programa que trabalham.
13. Existem locutores que têm preferência em trabalhar com alguns operadores.
14. Cada programa tem pelo menos um operador, um locutor e um produtor, mas pode ter várias pessoas para cada função.
15. Ainda relativamente aos programas emitidos interessa saber, mensalmente, o número total de entrevistados, o número total de pessoas que gostam do programa e o número total de pessoas que não gostam do programa.



Escreva um comando SQL que permita listar o nome das emissoras de rádio que emitiram todos os programas da empresa de produção “XPTO”.

Worksheet Query Builder

```
1 select e.nome_emissora from emissora e
2 where not exists (select * from programa p, empresa_producao ep
3   where p.nif_empresa = ep.nif_empresa and ep.nome_empresa = 'XPTO'
4   and p.id_programa not in (select p.id_programa
5     from transmissao t, programa p, empresa_producao ep
6       where t.id_programa = p.id_programa and p.nif_empresa = ep.nif_empresa
7       and ep.nome_empresa = 'XPTO'
8       and t.id_emissora = e.id_emissora));
9
```

Script Output x

Ou:

select e.nome_emissora from emissora e

where not exists(select p.id_programa

 from programa p, empresa_producao ep

 where p.nif_empresa = ep.nif_empresa and ep.nome_empresa = 'XPTO'

minus

 select p.id_programa

 from transmissao t, programa p, empresa_producao ep

 where t.id_programa = p.id_programa and p.nif_empresa = ep.nif_empresa

 and ep.nome_empresa = 'XPTO'

 and t.id_emissora = e.id_emissora);