**Ficha 3 - Introdução ao PL/SQL**

Base de Dados NoSQL

**Exercício 1.**

Criar uma view denominada ‘tit\_reagge’ que devolva sempre todos os títulos cujo género é ‘Reagge’.

create view TIT\_REAGGE as

    select TITULO from TITULO

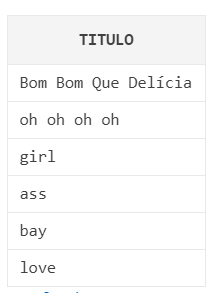
    inner join GENERO on GENERO.ID\_GENERO=TITULO.ID\_GENERO

    where GENERO.NOME = 'Reagge'

1. Introduzir um novo título cujo estilo seja 'Reagge' e verifique a sua inserção recorrendo à view acima criada.

insert into TITULO values (50, 'Bom Bom Que Delícia', 10, to\_date('10-01 2010', 'dd-mm-yyyy'), 2, 4, 5, 3);

Ao fazermos uma seleção de todos as linhas da *view* **TIT\_REAGGE** conseguimos ratificar a inserção desta nova entrada.



**Exercício 2.**

Acrescentar uma coluna no título que é o valor comercial do mesmo denominada val\_comercial que é um número inteiro de 3 dígitos.

alter table TITULO add VAL\_COMERCIAL number(3)

1. Colocar o val\_comercial igual ao preço de aquisição.

update TITULO set VAL\_COMERCIAL = PRECO



Através da tabela acima anexada podemos confirmar a adição de uma coluna e da sua igualdade com o **PRECO** do título em si.

1. Criar um procedimento que atualize o valor comercial de referência de título com o valor médio dos títulos num dado ano acrescido de uma percentagem (decimal), ano e percentagem são passados como parâmetro.

create or replace procedure AtualizaValor(ano number, percentagem number)

is

    media number;

begin

    update TITULO

        set VAL\_COMERCIAL = (select median(PRECO) from TITULO where extract(year from DTA\_COMPRA) = ano) \* (1+percentagem)

        where extract(year from DTA\_COMPRA) = ano;

end AtualizaValor;

1. Atualize os títulos de 2010 com uma percentagem de 20%.

call AtualizaValor(2010, 0.2)

1. Liste o título, preço de compra, valor comercial e a diferença entre o preço de aquisição e o valor comercial atual de todos os títulos cuja diferença seja maior que 0.

select TITULO, PRECO, VAL\_COMERCIAL, VAL\_COMERCIALPRECO as DIFERENCA

from TITULO

     where VAL\_COMERCIAL-PRECO > 0

**Exercício 3.**

Criar uma função lógica que devolva o preço de um título que receba como parâmetro o ‘id\_titulo’.

create or replace function DA\_PRECO (ID\_TIT number)

return integer as RESULTADO integer:=0;

begin

        select PRECO into RESULTADO from TITULO

            where ID\_TITULO = ID\_TIT;

return RESULTADO;

end;

1. Listar todos os títulos (id, titulo, preço) cujo preço seja 10 utilizando uma função lógica da\_preco(titulo) criada.

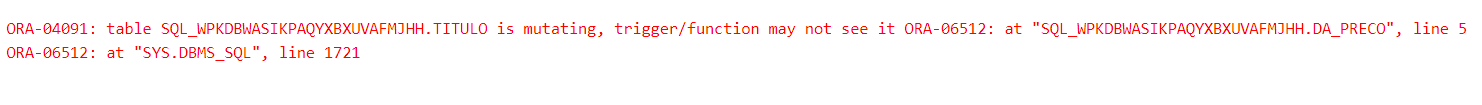
select ID\_TITULO, TITULO, PRECO from TITULO where DA\_PRECO(ID\_TITULO)=10



Testemunha-se que a função lógica criada para obter o preço de um título funciona, facilitando assim a procura e escrita da *query*.

1. Remover todos os títulos cujo preço seja 10 utilizando uma função lógica da\_preco(titulo) criada.

delete from TITULO where DA\_PRECO(ID\_TITULO) = 10



Através de algumas pesquisar acerca deste problema, a ideia com que fiquei foi que tal acontece quando se está a tentar aplicar uma *query* sobre os dados de uma tabela ao mesmo tempo que a estamos a tentar atualizar.

Na alínea anterior este erro não existiu, tendo em conta que apenas estávamos a selecionar informação da tabela **TITULO**. Neste caso, estamos a tentar obter os títulos cujo preço é de 10, ao mesmo tempo que tentamos eliminar os mesmos. Isto faz com que a tabela **TITULO** seja denominada como *mutating* – em processo de mutação.

**Exercício 4.**

Criar uma nova tabela que guarde o total gasto até agora e o valor comercial total da coleção.

create table GASTO\_TOTAL (

    ID\_TOTAL\_GASTO number not null constraint GASTO\_TOTAL\_PK primary key,

    VAL\_ATUAL\_TOTAL number not null,

    VAL\_COMERCIAL\_TOTAL number not null

);

1. Inserir na tabela o valor atual gasto e o valor comercial de toda a coleção.

insert into GASTO\_TOTAL values(1, 0, 0);

update GASTO\_TOTAL

    set VAL\_ATUAL\_TOTAL = (select sum(PRECO) from TITULO),

        VAL\_COMERCIAL\_TOTAL = (select sum(VAL\_COMERCIAL) from TITULO)

    where ID\_TOTAL\_GASTO = 1;

Insere-se um valor inicial nulo tanto para o valor atual gasto como para o valor comercial de toda a coleção, para se poder efetuar um *update* sobre a tabela que se encontra inicialmente vazia.

1. Criar uma sequência que sirva para incrementar o id dos títulos quando um novo é inserido. A sequencia deverá ser denominada ‘titulo\_sq’. Deverá ter em consideração que o valor inicial deverá considerar os títulos já inseridos.

create sequence TITULO\_SQ

    increment by 1

    start with 51

;

create or replace trigger ID\_TITULO\_TRIG

    before

        insert on TITULO

    for each row

begin

    if :NEW.ID\_TITULO is null then

        select TITULO\_SQ.NEXTVAL

        into :NEW.ID\_TITULO

        from DUAL;

    end if;

end;

A *sequence* criada garante que ao ser inserido um novo título com o valor do **ID\_TITULO** a NULL, este será gerado automaticamente com base no valor atual de títulos já existentes. Dado que já existem 50 títulos na coleção, a sequência deve começar com um valor de 51, sendo incrementado sempre 1 ao **ID\_TITULO** do próximo inserido.

Além desta sequência, um *trigger* parece ser necessário, na medida em que garante que este processo de atualização realmente aconteça.

1. Criar um trigger denominado ‘valor\_total’ que consiga atualizar a tabela criada em [4] com os valores quando é feita uma atualização na tabela título. Seja a adição (*insert*) de um novo título, a remoção (*delete*) ou a atualização (*update*) do preço ou valor comercial.

create or replace trigger valor\_total

    after

    insert or update or delete

    on TITULO

begin

    if inserting then

        update GASTO\_TOTAL

        set VAL\_ATUAL\_TOTAL = (select sum(PRECO) from TITULO),

            VAL\_COMERCIAL\_TOTAL = (select sum(VAL\_COMERCIAL) from TITULO)

        where ID\_TOTAL\_GASTO = 1;

    end if;

    if updating then

        update GASTO\_TOTAL

        set VAL\_ATUAL\_TOTAL = (select sum(PRECO) from TITULO),

            VAL\_COMERCIAL\_TOTAL = (select sum(VAL\_COMERCIAL) from TITULO)

        where ID\_TOTAL\_GASTO = 1;

    end if;

    if deleting then

        update GASTO\_TOTAL

        set VAL\_ATUAL\_TOTAL = (select sum(PRECO) from TITULO),

            VAL\_COMERCIAL\_TOTAL = (select sum(VAL\_COMERCIAL) from TITULO)

        where ID\_TOTAL\_GASTO = 1;

    end if;

end;

1. Testar o trigger anterior e a sequencia anterior inserindo um novo título: 'Enjoy Triggers', preço: 20, data: 22-02-2000, editora: 'Vevo', suporte: 'Spotify', género: 'Reagge', autor: 'Big Nelo' e o valor comercial de 22.

insert into TITULO

    values (NULL, 'Enjoy Triggers', 20, to\_date('22-02-2000', 'dd-mm-yyyy'), 4, 6, 5, 6, 22);

1. Testar o trigger atualizando o preço de aquisição do título: 'get get get get' para 30.

update TITULO set PRECO=30 where TITULO='get get get get';

1. Testar o trigger removendo o título: 'go head'.

delete from MUSICA where ID\_TITULO=15;

delete from REVIEW where ID\_TITULO=15;

delete from TITULO where TITULO='go head';

Com as imagens abaixo conseguimos provar que o exercício 4 funciona e responde às funcionalidades pedidas, conectando perfeitamente a tabela **GASTO\_TOTAL** com os novos pedidos construídos para a tabela **TITULO**.

Não só se percebe que a sequência funciona para um **ID\_TITULO** a NULL, criando automaticamente um novo ID, como se valida o facto de o *trigger*, que é responsável por atualizar a tabela **GASTO\_TOTAL**, estar a funcionar como suposto – o novo título tem um preço de 20 (513+20 = 533).

