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'
```

a) 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.

TITULO
Bom Bom Que Delícia
oh oh oh oh
girl
ass
bay
love

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)
```

a) Colocar o val_comercial igual ao preço de aquisição.

```
update TITULO set VAL_COMERCIAL = PRECO
```

ID_TITULO	TITULO	PRECO	DTA_COMPRA	ID_EDITORA	ID_SUPORTE	ID_GENERO	ID_AUTOR	VAL_COMERCIAL
50	Bom Bom Que Delícia	10	10-JAN-10	2	4	5	3	10
1	oh oh	10	01-FEB-10	1	4	2	1	10
2	oh oh oh	10	10-JAN-10	2	4	3	3	10
3	wit	10	12-MAR-00	3	5	4	5	10
4	oh oh oh oh	10	27-APR-10	4	6	5	6	10
5	imma	12	12-MAY-10	5	5	1	8	12

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

b) 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 extra
ct(year from DTA_COMPRA) = ano) * (1+percentagem)
        where extract(year from DTA_COMPRA) = ano;

end AtualizaValor;
```

c) Atualize os títulos de 2010 com uma percentagem de 20%.

```
call AtualizaValor(2010, 0.2)
```

d) 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;
```

a) 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
```

ID_TITULO	TITULO	PRECO
50	Bom Bom Que Delícia	10
1	oh oh	10
2	oh oh oh	10

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

b) 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
```

```
KUVAFMJHH.TITULO is mutating, trigger/function may not see it
```

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
);
```

a) 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.

b) 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.

c) 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;
```

d) 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);
```

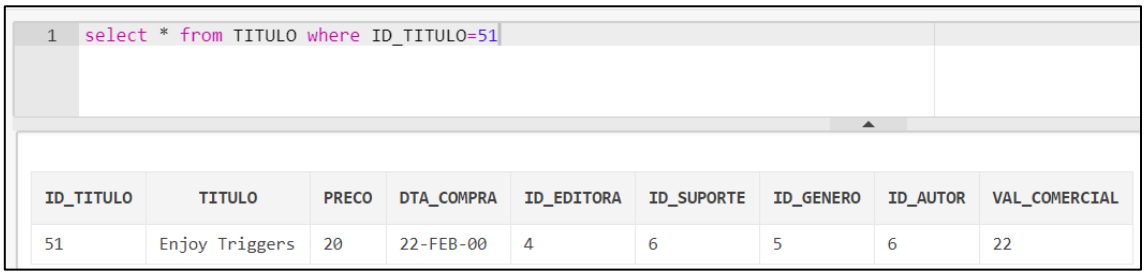
e) Testar o trigger atualizando o preço de aquisição do título: 'get get get get' para 30.

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

f) 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';
```

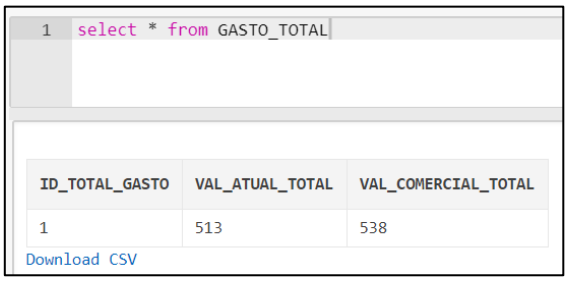
Com as imagens abaixo conseguimos provar que o exercício 4 funciona e responde às funcionalidades pedidas, conectando perfeitamente a tabela **GASTO_TOTAL** com os novos pedidos construídos para a tabela **TITULO**.



```
1  select * from TITULO where ID_TITULO=51
```

ID_TITULO	TITULO	PRECO	DTA_COMPRA	ID_EDITORA	ID_SUPORTE	ID_GENERO	ID_AUTOR	VAL_COMERCIAL
51	Enjoy Triggers	20	22-FEB-00	4	6	5	6	22

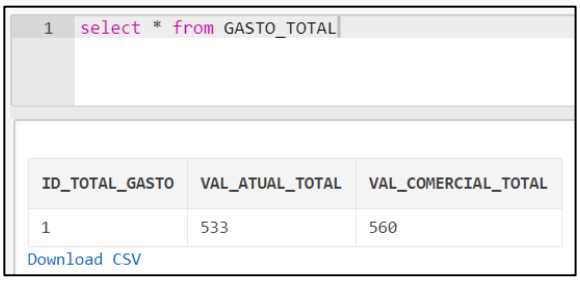
Não só se percebe que a sequência funciona para um **ID_TITULO** a NULL, criando automaticamente um novo ID, como se valida o facto de o *trigger*, que é responsável por atualizar a tabela **GASTO_TOTAL**, estar a funcionar como suposto – o novo título tem um preço de 20 (513+20 = 533).



```
1  select * from GASTO_TOTAL
```

ID_TOTAL_GASTO	VAL_ATUAL_TOTAL	VAL_COMERCIAL_TOTAL
1	513	538

Download CSV



```
1  select * from GASTO_TOTAL
```

ID_TOTAL_GASTO	VAL_ATUAL_TOTAL	VAL_COMERCIAL_TOTAL
1	533	560

Download CSV