TREINAMENTO PL/SQL

Curso EAD PL/SQL
Apostila com os comandos das vídeoaulas

Utilização dos comandos PL/SQL
Apostila criada por Grimaldo Oliveira
JULHO/2014

Aula 01 – comandos

- Criação do usuário/esquema -> aluno

```
CREATE USER aluno
IDENTIFIED BY  aluno
DEFAULT TABLESPACE Users
TEMPORARY TABLESPACE Temp;
```

- Alteração da senha do usuário -> aluno

```
ALTER USER aluno IDENTIFIED BY  treinamento;
```

- Concedendo o privilégio DBA ao usuário -> aluno

```
GRANT DBA TO  aluno;
```

- Criação da Tabela TB_TREINAMENTO

```
CREATE TABLE TB_TREINAMENTO
  (
    "ID" NUMBER(5) NOT NULL,
    "NOME" varchar2(10),
    "SEXO" char(1)
  );
```

- Inserindo dados na tabela TB_TREINAMENTO

```
INSERT INTO TB_TREINAMENTO (ID,NOME) VALUES (1,'GRIMALDO');
```

- Criação de constraints na Tabela TB_TREINAMENTO

```
ALTER TABLE "TB_TREINAMENTO" ADD CONSTRAINT
"CK__TREINA_SEXO" CHECK (Sexo in ('M','F')) ENABLE;

ALTER TABLE "TB_TREINAMENTO" ADD CONSTRAINT
"PK_TREINAMENTO" PRIMARY KEY ("ID") ENABLE;
```

- Renomeando constraints na Tabela TB_TREINAMENTO

```
ALTER TABLE TB_TREINAMENTO
   RENAME CONSTRAINT SYS_C004075 TO dname_unikey;

ALTER TABLE "TB_TREINAMENTO" DROP CONSTRAINT dname_unikey;
```

- Eliminando a Tabela TB_TREINAMENTO

```
DROP TABLE TB_TREINAMENTO;
```

Aula 02 – comandos

• Atualiza a tablea TB_LIVRO o campo qtde_estoque para 300 onde o campo titulo é igual a 'Banco de Dados'

UPDATE tb_livro SET qtde_estoque=300 WHERE TITULO='Banco de Dados'

• Elimina os registros da tabela TB_FUNCIONARIO onde o campo nome é igual a 'João'

DELETE FROM TB_FUNCIONARIO WHERE NOME='João'

• Exibe todos os campos da tabela TB_FUNCIONARIO

SELECT * FROM TB_FUNCIONARIO

• Exibe os campos nome e sexo da tabela TB_FUNCIONARIO

SELECT NOME,SEXO FROM TB_FUNCIONARIO

• Exibe os campos sexo e nome da tabela TB_FUNCIONARIO

SELECT SEXO,NOME FROM TB_FUNCIONARIO

• Exibe os campos nome e sexo da tabela TB_FUNCIONARIO ordenados de forma ascendente pelo campo nome

SELECT NOME,SEXO FROM TB_FUNCIONARIO ORDER BY NOME

• Exibe os campos nome e sexo da tabela TB_FUNCIONARIO ordenados de forma descendente pelo campo nome

SELECT NOME,SEXO FROM TB_FUNCIONARIO ORDER BY NOME DESC

• Exibe todos os campos da tabela TB_EDITORA onde o campo endereco é nulo, ou seja, ausência de qualquer dado

SELECT * FROM TB_EDITORA WHERE ENDERECO IS NULL

• Exibe todos os campos da tabela TB_EDITORA onde o campo endereco não é nulo, ou seja, há a existência de dados nos registros do campo endereco

SELECT * FROM TB_EDITORA WHERE ENDERECO IS NOT NULL

• Elimina a constraint FK_LIVRO_EDITORA da tabela TB_LIVRO

ALTER TABLE TB_LIVRO DROP CONSTRAINT FK_LIVRO_EDITORA

- Adiciona a constraint FK_LIVRO_EDITORA na tabela TB_LIVRO fazendo uma referência a tabela TB_EDITORA, desta forma está sendo criada uma FOREIGN KEY (chave estrangeira) para que exista um relacionamento entre as tabelas.

ALTER TABLE TB_LIVRO ADD CONSTRAINTS "FK_LIVRO_EDITORA" FOREIGN KEY ("ID_EDITORA") REFERENCES "ALUNO"."TB_EDITORA" ("ID_EDITORA") ENABLE

- Desabilitando a constraint FK_LIVRO_EDITORA da tabela TB_LIVRO

ALTER TABLE TB_LIVRO DISABLE CONSTRAINTS FK_LIVRO_EDITORA

- Habilitando a constraint FK_LIVRO_EDITORA da tabela TB_LIVRO

ALTER TABLE TB_LIVRO ENABLE CONSTRAINTS FK_LIVRO_EDITORA

- Exibindo os dados de duas tabelas TB_LIVRO e TB_EDITORA através do comando JOIN, que permite a junção entre tabelas

```
SELECT livro.titulo,
      editora.descricao
FROM TB_LIVRO  livro
       JOIN
      TB_EDITORA editora ON (livro.id_editora=editora.id_editora)
```

- Exibindo os dados de duas tabelas TB_LIVRO e TB_EDITORA através do comando LEFT JOIN, que permite a junção entre tabelas, desta vez, serão exibidos todos os dados da tabela principal(TB_LIVRO) relacionados ou não com a tabela TB_EDITORA

```
SELECT livro.titulo,
      editora.descricao
FROM TB_LIVRO  livro
      LEFT JOIN
      TB_EDITORA editora ON (livro.id_editora=editora.id_editora)
```

- Exibindo os dados de duas tabelas TB_LIVRO e TB_EDITORA através do comando RIGHT JOIN, que permite a junção entre tabelas, desta vez, serão exibidos todos os dados da tabela secundária(TB_EDITORA) relacionados ou não com a tabela TB_LIVRO, é o contrário do comando LEFT JOIN

```
SELECT livro.titulo,
    editora.descricao
FROM TB_LIVRO  livro
    RIGHT JOIN
    TB_EDITORA editora ON (livro.id_editora=editora.id_editora)
```

• Exibindo os dados de duas tabelas TB_LIVRO e TB_EDITORA através do comando FULL JOIN, que permite a junção entre tabelas, desta vez, serão exibidos todos os dados de ambas as tabelas tabela principal(TB_LIVRO) relacionados ou não com a tabela TB_EDITORA.

```
SELECT livro.titulo,
    editora.descricao
FROM TB_LIVRO  livro
    FULL JOIN
    TB_EDITORA editora ON (livro.id_editora=editora.id_editora)
```

Aula 03

- Exibe o sexo dos autores , mostrando o conteúdo não repetido.

```
SELECT  DISTINCT sexo FROM TB_AUTOR
```

- Exibe todos os autores que começam pelas iniciais ‘Jo’

```
SELECT  * FROM TB_AUTOR
where SUBSTR(nome,1,2)='Jo'
```

- Exibe o nome em maiúsculo

```
SELECT UPPER(nome) FROM TB_AUTOR
```

- Exibe o nome em minúsculo

```
SELECT LOWER(nome) FROM TB_AUTOR
```

- Exibe o nome armazenado no banco de dados e em maiúsculo dos autores que começam coma letra ‘p’

```
SELECT nome, UPPER(nome) as maiusculo FROM TB_AUTOR
WHERE SUBSTR(nome,1,1)='p'
```

- Exibe todos os livros que possuem a palavra ‘DE’ em qualquer parte do título do livro

```
SELECT  * FROM TB_LIVRO
where  UPPER(titulo) LIKE '%DE%'
```

- Exibição do campo sexo através do comando CASE, que exibe em duas situações ‘M’ para Masculino e ‘F’ para Feminino, caso contrário ‘Sexo inválido’

```
SELECT nome, CASE  sexo
  WHEN 'M' THEN 'Masculino'
  WHEN 'F' THEN 'Feminino'
  ELSE 'Sexo inválido'
END sexo , sexo
FROM TB_AUTOR
```

- Exibe o mês de Nascimento

```
SELECT  * FROM TB_AUTOR
WHERE EXTRACT(MONTH FROM DATA_NASCIMENTO)=05
```

- Exibe o dia de Nascimento

```
SELECT  * FROM TB_AUTOR
WHERE EXTRACT(DAY FROM DATA_NASCIMENTO)=10
```

- Exibe o ano de Nascimento

```
SELECT  * FROM TB_AUTOR
WHERE EXTRACT(YEAR FROM DATA_NASCIMENTO)=1980
```

- Exibe o nome, dia, mês e ano de nascimento dos autores que começam pela letra 'P'

```
SELECT  nome, EXTRACT(DAY FROM DATA_NASCIMENTO) AS DIA,
      EXTRACT(MONTH FROM DATA_NASCIMENTO) AS MES,
      EXTRACT(YEAR FROM DATA_NASCIMENTO) AS ANO
FROM TB_AUTOR
WHERE UPPER(SUBSTR(NOME,1,1))='P'
```

Aula 04

- Conta a quantidade de Livros que possuem o título Banco de Dados

```
SELECT count(*) quantidade_autores
FROM TB_LIVRO l
JOIN
TB_LIVRO_AUTOR la ON (l.id_livro=la.id_livro)
WHERE
UPPER(l.titulo) = 'BANCO DE DADOS'
```

- Exibe a projeção de valores agregados para o campo preço, utilizando os comandos SUM, AVG, MAX,MIN

```
SELECT
SUM(l.preco) soma_total,
TO_CHAR(AVG(l.preco),'9999.99')  media_preco,
MAX(preco) maior_preco,
MIN(preco) menor_preco
FROM
TB_LIVRO l
```

- Exibe a projeção de valores agregados contando a quantidade de livros em relação a editora que ele pertence.

```
SELECT
ed.descricao,
COUNT(*) quantidade_livros
FROM TB_LIVRO l
JOIN
TB_EDITORA ed on ( l.id_editora=ed.id_editora)
GROUP BY ed.descricao
```

- Exibe SUM e AVG, de acordo com a editora que publicou o livro

```
SELECT
ed.descricao,
SUM(preco) soma_total,
AVG(preco) media_preco
FROM TB_LIVRO l
JOIN
TB_EDITORA ed on ( l.id_editora=ed.id_editora)
GROUP BY ed.descricao
```

- Exibe a quantidade de livros por título apenas os títulos que possuírem mais de 2 livros entre os dados, este filtro é graças a cláusula HAVING.

```
SELECT
l.titulo,
count(*) quantidade_livros
FROM TB_LIVRO l
JOIN
TB_LIVRO_AUTOR la on ( l.id_livro=la.id_livro)
GROUP BY l.titulo
HAVING
  COUNT(*)>1
```

- Faz a união entre suas tabelas pelos campos nome e sexo

```
SELECT
   nome,
   sexo
 FROM
    TB_AUTOR
UNION   [ALL]
SELECT
   nome,
   sexo
 FROM
    TB_FUNCIONARIO
```

- Apresenta o nome dos autores que **são** funcionários da editora.

```
SELECT
 nome
FROM
 TB_AUTOR
INTERSECT
SELECT
 nome
FROM
  TB_FUNCIONARIO
```

- Apresenta o nome dos autores que **não são** funcionários da editora.

```
SELECT
 nome
FROM
 TB_AUTOR
MINUS
SELECT
 nome
FROM
  TB_FUNCIONARIO
```

Aula 05

- Apresenta Os títulos e preços dos livros com preços maior ou igual a média geral dos livros à venda.

```
SELECT
  L.titulo,
  L.preco
FROM
 TB_LIVRO L
WHERE
   L.preco >=
    (
      SELECT
         AVG(I.preco)
      FROM
        TB_LIVRO I
    )
```

- Apresenta o nome dos autores que tiveram a publicação de algum livro na editora.

```
SELECT
  AU.nome
FROM
   TB_AUTOR AU
WHERE
   AU.id_autor IN
     (
       SELECT
        LA.id_autor
       FROM
         TB_LIVRO_AUTOR LA
     )
```

- Apresenta o nome dos autores que NÃO SÃO autores do livro BANCO DE DADOS.

```
SELECT
  A.nome
FROM
 TB_AUTOR A
 WHERE
   A.id_autor NOT IN
     (
```

```
    SELECT
      LA.id_autor
    FROM
      TB_LIVRO_AUTOR LA
    JOIN
      TB_LIVRO L ON (LA.id_livro=L.id_livro)
    WHERE
       UPPER(L.titulo) = 'BANCO DE DADOS'
  )
```

- Apresenta o nome dos autores que tiveram a publicação de algum livro na editora.

```
SELECT
  A.nome
FROM
 TB_AUTOR A
 WHERE
   EXISTS
   (
     SELECT *
     FROM
      TB_LIVRO_AUTOR LA
     WHERE
      LA.id_autor=A.id_autor
   )
```

- Apresenta as siglas da lotação com a tabela hierarquizada, através do comando START WITH ... CONNECT BY.

```
SELECT
  sigla,
  level,
  sys_connect_by_path(sigla,',') caminho
FROM
  TB_LOTACAO
START WITH id_lotacao_pai is null
CONNECT BY NOCYCLE PRIOR id_lotacao=id_lotacao_pai
```

- Apresenta os títulos e nomes dos autores com suas publicações

```
CREATE OR REPLACE FORCE VIEW
AS
  SELECT
    L.titulo,
    A.nome
  FROM
     TB_AUTOR A
```

```
JOIN
   TB_LIVRO_AUTOR LA    ON ( A.id_autor=la.id_autor)
JOIN
  TB_LIVRO L  ON (L.id_livro = LA.id_livro)
```

Aula 06

- Escrevendo blocos anônimos, mostrando o resultado na tela

```
SET SERVEROUTPUT ON
DECLARE

vsalario number;
vpercaumento number;
vtotalsal number;

BEGIN

  vsalario:=2500.00;
  vpercaumento:=30/100;
  vtotalsal := vsalario + (vsalario*vpercaumento);
  DBMS_OUTPUT.PUT_line (' O novo salário é de: '||vtotalsal);
END;
```

- Declarando variáveis e constantes dentro de blocos anônimos

```
SET SERVEROUTPUT ON
DECLARE

vsalario number :=4500;
vsituacao boolean;
vpercaumento number :=30/100;
vtotalsal number;

BEGIN

  vsituacao:=TRUE;

  IF vsituacao THEN
     vtotalsal := vsalario + (vsalario*vpercaumento);
     DBMS_OUTPUT.PUT_line (' O novo salário é de: '||vtotalsal);
  ELSE
     DBMS_OUTPUT.PUT_line (' Não houve aumento!!! '||vtotalsal);
  END IF;
END;
```

- Manipulando variáveis data e string

```
SET SERVEROUTPUT ON
DECLARE
```

```
vdata_pagamento date :='02/01/2011';


BEGIN
   DBMS_OUTPUT.PUT_line (' A data do pagamento é : '||vdata_pagamento);
   -- Acrescenta 10 dias a data de pagamento
   vdata_pagamento:=vdata_pagamento+10;
   DBMS_OUTPUT.PUT_line (' A data do pagamento mais 10 dias é de :
'||vdata_pagamento);
   -- Data do sistema
   DBMS_OUTPUT.PUT_line (' A data do sistema é : '||sysdate);
   -- data entre o pagamento e do sistema
   DBMS_OUTPUT.PUT_line (' A quantidade de dias entre hoje e o pagamento :
'||(sysdate - vdata_pagamento));
   -- data entre o pagamento e do sistema ajustado
   DBMS_OUTPUT.PUT_line (' A quantidade de dias entre hoje e o pagamento-
ajustado : '||floor(sysdate - vdata_pagamento));
END;
```

- Utilizando identificadores de blocos anônimos

```
SET SERVEROUTPUT ON
<<EXTERNO>>
DECLARE

vsalario number :=4500;
vsituacao boolean;
vpercaumento number :=30/100;
vtotalsal number;
BEGIN

 vsituacao:=TRUE;

 IF vsituacao THEN
    vtotalsal := vsalario + (vsalario*vpercaumento);
    DBMS_OUTPUT.PUT_line (' O novo salário é de: '||vtotalsal);
 ELSE
    DBMS_OUTPUT.PUT_line (' Não houve aumento!!! '||vtotalsal);
 END IF;
  <<INTERNO>>
   DECLARE
     Vtotalsal NUMBER :=5000;
   BEGIN
    DBMS_OUTPUT.PUT_line (' O salário externo é de : '||externo.vtotalsal);
    DBMS_OUTPUT.PUT_line (' O salário local é de : '||vtotalsal);
   END;
END;
```

- Utilizando comando condicional (IF THEN e ELSE)

```
SET SERVEROUTPUT ON
DECLARE

vsalario number :=3500;
vsituacao boolean;
vpercaumento number :=30/100;
vtotalsal number;

BEGIN

  vsituacao:=FALSE;

  IF vsituacao THEN
      vtotalsal := vsalario + (vsalario*vpercaumento);
      DBMS_OUTPUT.PUT_line (' O novo salário com a situacao TRUE é de:
'||vtotalsal);
  ELSE
     IF vsalario > 4000 THEN
        vtotalsal := vsalario ;
     ELSE
        vtotalsal := vsalario - 1000;
     END IF;
     DBMS_OUTPUT.PUT_line (' o salário com a situacao FALSE é de : '||vtotalsal);
  END IF;
END;
```

Aula 07

- Utilizando o comando condicional (CASE)

```
SET SERVEROUTPUT ON
DECLARE
  vtotalquant number;
 BEGIN
  SELECT  sum(quantidade) INTO vtotalquant
  FROM TB_ITENS_PEDIDO;
  CASE
     WHEN vtotalquant <=200 THEN
      DBMS_OUTPUT.PUT_line (' O estoque esta proximo do mínimo: '||vtotalquant) ;
     WHEN vtotalquant <=300 THEN
      DBMS_OUTPUT.PUT_line (' O estoque esta completo: '||vtotalquant) ;
     ELSE
      DBMS_OUTPUT.PUT_line (' O estoque esta wm excesso: '||vtotalquant);
   END CASE;
 END;
```

- Utilizando comando de iteração (LOOP)

```
SET SERVEROUTPUT ON
DECLARE
  vrepeticao number :=0 ;
 BEGIN
    LOOP
       vrepeticao:=vrepeticao+1;
      IF vrepeticao >5 THEN
           EXIT;
      END IF;
      DBMS_OUTPUT.PUT_line (' IMPRESSAO : '||vrepeticao);
    END LOOP;
 END;
```

- Utilizando comando de iteração (WHILE)

```
SET SERVEROUTPUT ON
DECLARE
  vrepeticao number :=0 ;
 BEGIN
    WHILE vrepeticao<5 LOOP
       vrepeticao:=vrepeticao+1;
       DBMS_OUTPUT.PUT_line (' IMPRESSAO : '||vrepeticao);
    END LOOP;
 END;
```

- Utilizando comando de iteração (LOOP-FOR)

```
SET SERVEROUTPUT ON
DECLARE
  vrepeticao number :=0 ;
 BEGIN
    FOR vrepeticao IN 1..5 LOOP
        DBMS_OUTPUT.PUT_line (' IMPRESSAO : '||vrepeticao);
    END LOOP;
 END;
```

- Interagindo comandos de banco de dados com bloco anônimo

```
SET SERVEROUTPUT ON
DECLARE
 vid_autor INT;
 vnome varchar(50);
 vsexo char(01);
BEGIN
   vid_autor:=1;
    SELECT
      nome,
      sexo
    INTO
      vnome,
      vsexo
    FROM
      TB_AUTOR
    WHERE
      ID_AUTOR= vid_autor;

  DBMS_OUTPUT.PUT_line (' Nome e sexo do autor: '||vnome||' --- '||vsexo);
END;
```

- Manipulando valores nulos

```
SET SERVEROUTPUT ON
DECLARE
 vid_autor INT;
 vid_autor2 INT;
 vnome varchar(50);
 vsexo char(01);
BEGIN
   vid_autor:=1;
   vid_autor2:= 23;
   vid_autor:=nvl(vid_autor2,vid_autor);
```

```
   SELECT
     nome,
     sexo
   INTO
     vnome,
     vsexo
   FROM
     TB_AUTOR
   WHERE
     ID_AUTOR= vid_autor;

  DBMS_OUTPUT.PUT_line (' Nome e sexo do autor: '||vnome||' --- '||vsexo);
END;
```

Aula 08

- Atribuição de variáveis com o tipo %type

```
set serveroutput on
declare
vid_autor  TB_AUTOR.id_autor%TYPE;
vnome  TB_AUTOR.nome%TYPE;
vsexo  TB_AUTOR.sexo%TYPE;

BEGIN
vid_autor:=1;

SELECT
nome, sexo
INTO
vnome,vsexo
FROM
TB_AUTOR
WHERE
id_autor=vid_autor;
DBMS_OUTPUT.PUT_LINE('Nome e sexo do Autor: '||vnome||'   '||vsexo);
END;
```

- Atribuição de variáveis com o tipo %rowtype

```
set serveroutput on
declare
vregautor  TB_AUTOR%ROWTYPE;

BEGIN
vregautor.id_autor:=1;

SELECT
nome, sexo
INTO
vregautor.nome,
vregautor.sexo
FROM
TB_AUTOR
WHERE
id_autor=vregautor.id_autor;
DBMS_OUTPUT.PUT_LINE('Nome e sexo do Autor: '||vregautor.nome||'
'||vregautor.sexo);
END;
```

- Utilizando o ROWID

```
set serveroutput on
declare
vrowid UROWID;

BEGIN

SELECT
ROWID
INTO
vrowid
FROM
TB_EDITORA
WHERE
UPPER(descricao) = 'CAMPUS';
DBMS_OUTPUT.PUT_LINE('O endereco da editora eh: '||vrowid);
END;
```

- Trabalhando com cursores implícitos. Reajustar os livros em 5% e verificar a quantidade de registros que foram afetados.

```
set serveroutput on
BEGIN
UPDATE
TB_LIVRO
SET preco = preco * 1.05
WHERE
preco > = 10;

IF (SQL%NOTFOUND) THEN
   DBMS_OUTPUT.PUT_LINE('Nao houve livro reajustado!!!');
ELSE
  DBMS_OUTPUT.PUT_LINE('A quantidade de livros reajustados foi de:
'||SQL%ROWCOUNT);
END IF;

END;
```

- Trabalhando com cursores explícitos. Apresenta o título e preço registro a registro de forma seqüencial.

```
set serveroutput on
DECLARE
vreglivros TB_LIVRO%ROWTYPE;

CURSOR clivros IS
SELECT
 L.*
FROM
  TB_LIVRO L
  JOIN
TB_EDITORA E ON (L.id_editora=E.id_editora)
WHERE
UPPER(E.descricao)='CAMPUS';
BEGIN
OPEN clivros;
LOOP
FETCH clivros INTO vreglivros;
EXIT WHEN clivros%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE('Titulos e Precos dos livros: '||vreglivros.titulo||' ,
'||vreglivros.preco);
END LOOP;
CLOSE clivros;
END;
```

- Cursores explícitos utilizando o FOR IN

```
set serveroutput on
DECLARE
vreglivros TB_LIVRO%ROWTYPE;

CURSOR clivros IS
SELECT
 L.*
FROM
  TB_LIVRO L
  JOIN
TB_EDITORA E ON (L.id_editora=E.id_editora)
WHERE
UPPER(E.descricao)='CAMPUS';
BEGIN
FOR vreglivros IN clivros
LOOP
  DBMS_OUTPUT.PUT_LINE('Titulos e Precos dos livros: '||vreglivros.titulo||' ,
'||vreglivros.preco);
```

```
END LOOP;
END;
```

- Cursores implícito utilizando o FOR IN

```
set serveroutput on
DECLARE
vreglivros TB_LIVRO%ROWTYPE;

BEGIN
FOR vreglivros IN (SELECT
 L.*
FROM
  TB_LIVRO L
  JOIN
TB_EDITORA E ON (L.id_editora=E.id_editora)
WHERE
UPPER(E.descricao)='CAMPUS'
)
LOOP
  DBMS_OUTPUT.PUT_LINE('Titulos e Precos dos livros: '||vreglivros.titulo||' ,
'||vreglivros.preco);
END LOOP;
END;
```

- Bloqueando registros em um cursos para posterior UPDATE ou DELETE.

```
set serveroutput on
DECLARE
vpreco TB_LIVRO.preco%TYPE;
vdescricao TB_EDITORA.descricao%TYPE;
vpercajuste number;

CURSOR clivros IS
SELECT
 L.preco,
 UPPER(E.descricao)
FROM
  TB_LIVRO L
  JOIN
TB_EDITORA E ON (L.id_editora=E.id_editora)
FOR UPDATE OF L.preco;
BEGIN
OPEN clivros;
LOOP
FETCH clivros INTO vpreco,vdescricao;
EXIT WHEN clivros%NOTFOUND;
```

```
  IF vdescricao='CAMPUS' THEN
    vpercajuste:=5;
  ELSE
    vpercajuste:=10;
  END IF;
  UPDATE TB_LIVRO SEt preco=preco + (preco*vpercajuste /100) WHERE
CURRENT OF clivros;

END LOOP;
CLOSE clivros;
END;
```

Aula 09

- Tratando erros internos do Oracle (EXCEPTION)

```
set serveroutput on
DECLARE
vvalor1 NUMBER :=100;
vvalor2 NUMBER :=0;

BEGIN
vvalor1 := vvalor1 / vvalor2;

EXCEPTION
  WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE (' Valor 2 não pode ser zero');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (' Erro não identificado ocorreu');
END;
```

- Tratando outros erros internos do Oracle (EXCEPTION)

```
set serveroutput on
DECLARE
vvalor number;

BEGIN
--INSERT INTO TB_EDITORA VALUES
(SQ_EDITORA.NEXTVAL,'TESTE2','S/N');

SELECT  preco  INTO vvalor FROM TB_LIVRO;
WHERE ID_LIVRO IN (1,2);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE (' A consulta não retornou nenhum registro');
   WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE (' A consulta retornou mais de um registro');
   WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (' Erro nao identificado ocorreu');
    ROLLBACK;
END;
```

- Tratando erros internos do Oracle (EXCEPTION) , interagindo com comandos de banco de dados com tratamento de exceção.

```
set serveroutput on
DECLARE
vnome varchar(50);
vsexo char(01);
BEGIN
```

```
   BEGIN
     SELECT  nome, sexo  INTO vnome, vsexo FROM TB_AUTOR WHERE id_autor
in (2,3);

     EXCEPTION
       WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE (' A consulta não retornou nenhum registro');
       WHEN TOO_MANY_ROWS THEN
         DBMS_OUTPUT.PUT_LINE (' A consulta retornou mais de um registro');

   END;
   DBMS_OUTPUT.PUT_LINE('Nome e sexo do autor :'||vnome||'--'||vsexo);
END;
```

- Criando exceções de usuários

```
set serveroutput on
DECLARE
vcodigo NUMBER ;
verro varchar2(64);
vdata date :='01.10.2010';
DATA_INVALIDA EXCEPTION;

BEGIN

  --vcodigo:=100/0;
  IF vdata < SYSDATE  THEN
    RAISE DATA_INVALIDA;
  END IF;
  EXCEPTION
   WHEN DATA_INVALIDA THEN
      DBMS_OUTPUT.PUT_LINE('Data invalida!!!');
   WHEN OTHERS THEN
     vcodigo:=SQLCODE;
     verro:=SUBSTR(SQLERRM,1,64);
      DBMS_OUTPUT.PUT_LINE('Erro!!!  '||vcodigo||' :' || verro);
END;
```

- Enviando mensagens de erro para a aplicação

```
set serveroutput on
DECLARE
vdata date :='01.10.2010';
BEGIN

  IF vdata < SYSDATE  THEN
    RAISE_APPLICATION_ERROR (-20100,'Data anterior a data do servidor');
```

```
  END IF;

END;
```

Aula 10

- Chamando uma procedure e manipulando parâmetros (IN,OUT,IN,OUT)

```
CREATE OR REPLACE PROCEDURE SP_INICIO
(
pparam1  IN number,
pparam2 in out number,
pparam3 out number
)
AS
BEGIN
DBMS_OUTPUT.PUT_LINE(' Valores dentro da procedure(1): ' ||pparam1 || '---' ||
pparam2 ||'---' || pparam3);
--pparam1:= pparam1 * 1.5; --Não é possível modificar parametro IN
pparam2:=pparam1 *2;
pparam3 := pparam1*3;
DBMS_OUTPUT.PUT_LINE(' Valores dentro da procedure(2): ' ||pparam1 || '---' ||
pparam2 ||'---' || pparam3);
END;

--- Chamando a procedure criado no bloco anonimo
set serveroutput on
DECLARE
vparam1 number :=500;
vparam2 number :=600;
vparam3 number :=200;
BEGIN
SP_INICIO(vparam1,vparam2,vparam3);
DBMS_OUTPUT.PUT_LINE(' Valores fora da procedure: ' ||vparam1 || '---' || vparam2
||'---' || vparam3);
END;
```

- Chamando uma procedure através de outra procedure

```
CREATE OR REPLACE PROCEDURE SP_AREA_RETANGULO
(
pbase number,
paltura number,
parea OUT number
)
AS
BEGIN
parea:=pbase*paltura;
END;
```

```
CREATE OR REPLACE PROCEDURE SP_CALCULA_AREA
AS
VAREA NUMBER;
BEGIN
sp_area_RETANGULO(2,4,varea);
dbms_output.put_line('A area da figura é: ' || VAREA);
END;

set serveroutput on
EXEC SP_CALCULA_AREA;
```

- Criando uma função que calcula a área de um triangulo (IN,OUT,IN,OUT)

```
CREATE OR REPLACE FUNCTION FC_CALCULA_AREA_TRIANGULO
(
pbase number,
paltura number default 4
)
RETURN number
AS
varea number;
BEGIN
varea := pbase * paltura;
RETURN varea;
END;

SELECT FC_CALCULA_AREA_TRIANGULO(2,5) FROM DUAL;
SELECT FC_CALCULA_AREA_TRIANGULO(2) FROM DUAL;
```

- Chamando uma função através de uma procedure

```
CREATE OR REPLACE PROCEDURE SP_CALCULA_AREA2
AS
varea number;
BEGIN
VAREA:= FC_CALCULA_AREA_TRIANGULO(2,3);
DBMS_OUTPUT.PUT_LINE('A area da figura é:   ' || varea);
END;

SET SERVEROUTPUT ON
EXEC SP_CALCULA_AREA2;
```

- Chamando uma função para permitir que o preço de um livro seja exibido com o seu valor em dobro

```
CREATE OR REPLACE FUNCTION FC_ELEVA_PRECO
(
ppreco number
)
RETURN number
AS
BEGIN
RETURN ppreco*ppreco;
END;

SELECT titulo,preco, FC_ELEVA_PRECO(preco) FROM tb_livro
```

Aula 11

- Criando uma trigger(gatilho) Não permite o cadastramento de autores menores que 16 anos

```
CREATE OR REPLACE TRIGGER TR_MENOR
BEFORE INSERT OR UPDATE
ON TB_AUTOR
FOR EACH ROW

DECLARE vidade number;
BEGIN
  vidade:= extract (YEAR FROM SYSDATE) - extract (YEAR FROM
:NEW.data_nascimento);
    IF (vidade = 16) AND ( extract (MONTH FROM SYSDATE) > extract (MONTH
FROM :NEW.data_nascimento)) THEN
       vidade:=vidade-1;
    END IF;
 --END IF;
  if (vidade < 16) THEN
      RAISE_APPLICATION_ERROR(-20301,'Autor menor do que 16 anos');
  END IF;
END;

INSERT INTO TB_AUTOR VALUES(sq_autor.nextval,'Fernando','M','23.10.1990');
```

- Os produtos são somados e não podem ultrapassar 600 reais.

```
CREATE OR REPLACE TRIGGER TR_LIMITE_PEDIDO
BEFORE INSERT OR UPDATE
ON TB_ITENS_PEDIDO
FOR EACH ROW

DECLARE vvalor_pedido number;
BEGIN
  SELECT
    SUM(preco)
  INTO
    vvalor_pedido
  FROM
    tb_itens_pedido
  WHERE
     id_pedido=:New.id_pedido;
vvalor_pedido:=vvalor_pedido+:New.preco;
IF (vvalor_pedido > 600 ) THEN
      RAISE_APPLICATION_ERROR(-20301,'Valor limite do pedido excedido!!!');
  END IF;
```

```
END;
INSERT INTO TB_ITENS_PEDIDO VALUES(sq_itens_pedido.nextval,1,1,10,350);
```

- É dado baixa no estoque ao acrescentar um novo livro ao pedido

```
CREATE OR REPLACE TRIGGER TB_BAIXA_ESTOQUE
AFTER INSERT OR UPDATE
ON TB_ITENS_PEDIDO
FOR EACH ROW

BEGIN
UPDATE
  TB_LIVRO
SET
  QTDE_ESTOQUE=QTDE_ESTOQUE-:NEW.quantidade
  WHERE
   id_livro=:NEW.id_livro;
END;

INSERT INTO TB_ITENS_PEDIDO VALUES(sq_itens_pedido.nextval,1,1,10,50);
```

- Não permite um reajuste de mais de 50%

```
CREATE OR REPLACE TRIGGER TB_LIMITE_REAJUSTE
BEFORE UPDATE
ON TB_LIVRO
FOR EACH ROW

BEGIN
IF (:NEW.preco >= :OLD.preco * 1.5) THEN
   RAISE_APPLICATION_ERROR (-20334,'Reajuste não permitido!!');
END IF;
END;

UPDATE TB_LIVRO SET PRECO=320 WHERE id_livro=1;
```

- Registrando na tabela TB_LOG a eliminação do registro da editora que deseja, informando o usuário e data do sistema

```
CREATE OR REPLACE TRIGGER TB_LOG_EDITORA
AFTER DELETE
ON TB_EDITORA
FOR EACH ROW

DECLARE voperacao varchar2(100);
BEGIN
```

```
  voperacao:='DELECAO DA EDITORA :  '||:OLD.descricao;
  INSERT INTO TB_LOG VALUES (sq_log.nextval, user,sysdate,voperacao);
END;


INSERT INTO TB_EDITORA VALUES (sq_EDITORA.nextval, 'MOREIRA','RUA
DA ESPERA');


DELETE FROM TB_EDITORA WHERE UPPER(descricao)='MOREIRA'
```

Aula 12

- Criando um pacote [PACKAGE]

```
CREATE OR REPLACE PACKAGE TREINAMENTO AS
FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
RETURN  number;
END;

CREATE OR REPLACE PACKAGE BODY TREINAMENTO AS
FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
RETURN  number
IS
  BEGIN
RETURN  pbase * paltura;
  END;
END;

set serveroutput on
DECLARE
varea number;
BEGIN
 varea:=  TREINAMENTO.FC_CALCULA_AREA(5,4);
 DBMS_OUTPUT.PUT_LINE(' A área da figura é   : ' || varea);
END;
```

- Criando um pacote [PACKAGE] com sobrecarga de função.

```
CREATE OR REPLACE PACKAGE TREINAMENTO AS
cpi constant number:=3.1416;
 FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
 RETURN  number;
 FUNCTION FC_CALCULA_AREA(praio number)
 RETURN  number;
END;

CREATE OR REPLACE PACKAGE BODY TREINAMENTO AS
FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
RETURN  number
IS
  BEGIN
RETURN  pbase * paltura;
  END;
```

```
FUNCTION FC_CALCULA_AREA(praio number)
RETURN  number
IS
 BEGIN
  RETURN  cpi * praio**2;
 END;
END;

set serveroutput on
DECLARE
varea number;
BEGIN
 varea:=  TREINAMENTO.FC_CALCULA_AREA(3);
 DBMS_OUTPUT.PUT_LINE(' A área da figura é  passando o raio  : ' || varea);
 varea:=  TREINAMENTO.FC_CALCULA_AREA(5,4);
 DBMS_OUTPUT.PUT_LINE(' A área da figura é : ' || varea);
END;
```

- Criando um pacote [PACKAGE] com encapsulamento de função.

```
CREATE OR REPLACE PACKAGE TREINAMENTO AS
cpi constant number:=3.1416;
 FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
 RETURN  number;
 FUNCTION FC_CALCULA_AREA(praio number)
 RETURN  number;
 FUNCTION FC_CALCULA_AREA_FIGURA(pmedida1 number, pmedida2 number)
 RETURN number;
END;

CREATE OR REPLACE PACKAGE BODY TREINAMENTO AS
FUNCTION FC_CALCULA_AREA(pbase number, paltura number)
RETURN  number
IS
  BEGIN
RETURN  pbase * paltura;
  END;

FUNCTION FC_CALCULA_AREA(praio number)
RETURN  number
IS
 BEGIN
  RETURN  cpi * praio**2;
 END;

FUNCTION FC_CALCULA_AREA_FIGURA(pmedida1 number, pmedida2 number)
RETURN  number
IS
```

```
 BEGIN
   IF (pmedida2 is not null)  THEN
     RETURN FC_CALCULA_AREA(pmedida1, pmedida2);
   ELSE
     RETURN FC_CALCULA_AREA(pmedida1);
   END IF;
 END;
END;

set serveroutput on

DECLARE
varea number;
BEGIN
 varea:=  TREINAMENTO.FC_CALCULA_AREA_FIGURA(3,null);
 DBMS_OUTPUT.PUT_LINE(' A área da figura é  : ' || varea);
END;
```