Ícone

Descrição gerada automaticamente

Faculdade de Informática e Administração Paulista

LISTA DE EXERCÍCIOS

PL/sql

Procedural Language/Structured Query Language

professor mEstre alexandre barcelos

**Índice**

**Exercícios**

01-Introdução ao PL/SQL 4

02-Declarando Variáveis PL/SQL 5

03-Instruções DML no PL/SQL 7

04-Estruturas de Controle 9

05-Registros PL/SQL 10

06-Criando Cursores Explícitos 11

07-Tratando Exceções 13

08-Procedures 14

09-Functions 15

10-Packages 16

11-Triggers 17

**Índice**

**Soluções**

01-Introdução ao PL/SQL 19

02-Declarando Variáveis PL/SQL 20

03-Instruções DML no PL/SQL 21

04-Estruturas de Controle 22

05-Registros PL/SQL 23

06-Criando Cursores Explícitos 24

07-Tratando Exceções 26

08-Procedures 27

09-Functions 28

10-Packages 29

11-Triggers 30

**Lista de Exercícios**

Esses exercícios estão disponíveis no final de cada apostila.

**Cap 01-Introdução ao PL/SQL**

1. Qual dos seguintes blocos PL/SQL é executado corretamente?

a. BEGIN  
END;

b. DECLARE  
amount INTEGER(10);  
END;

c. DECLARE  
BEGIN  
END;

d. DECLARE  
amount INTEGER(10);  
BEGIN  
DBMS\_OUTPUT.PUT\_LINE(amount);  
END;

1. Crie e execute um bloco anônimo simples que exiba "Hello World". Execute e salve este script como lab\_01\_02\_soln.sql.

**Cap 02-Declarando Variáveis PL/SQL**

1. Identifique se os nomes dos identificadores são válidos ou inválidos:
2. today
3. last\_name
4. today’s\_date
5. Number\_of\_days\_in\_February\_this\_year
6. Isleap$year
7. #number
8. NUMBER#
9. number1to7
10. Identifique se a declaração de variável e a inicialização é válida ou inválida:
11. printer\_name constant VARCHAR2(10);
12. deliver\_to VARCHAR2(10):=Johnson;
13. by\_when DATE:= SYSDATE+1;
14. Examine o seguinte bloco anônimo e escolha a resposta correta apropriada.

SET SERVEROUTPUT ON

DECLARE

fname VARCHAR2(20);

lname VARCHAR2(15) DEFAULT 'fernandez';

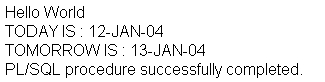
BEGIN

DBMS\_OUTPUT.PUT\_LINE( FNAME ||' ' ||lname);

END;

/

1. O bloco será executado com sucesso mas não será impresso nada.
2. O bloco retornará um erro porque a variável fname é declarada sem ser iniciada.
3. O bloco será executado com sucesso e irá imprimir 'fernandez'.
4. O bloco retornará um erro porque você não pode usar a palavra-chave DEFAULT para inicializar uma variável do tipo VARCHAR2.
5. O bloco retornará um erro porque a variável FNAME não foi declarada.
6. Crie um bloco anônimo. Carregue o script da aula 01 lab\_01\_02\_soln.sql, que você criou na pergunta 2 do exercício 1.
7. Adicione a seção DECLARE nesse bloco PL/SQL. Na seção DECLARE declare as seguintes variáveis:
   1. Variável today do tipo DATE. Inicializar today com SYSDATE
   2. Variável tomorrow do tipo today. Use o atributo% TYPE para declarar essa variável.
8. Na seção executável inicialize a variável tomorrow com uma expressão que calcula a data de tomorrow (adicione um ao valor de today). Exiba o valor de today e tomorrow após de imprimir 'Hello World‘
9. Execute e salve este script como lab\_02\_04\_soln.sql. Um exemplo da saída é mostrada a seguir.



1. Edite o script lab\_02\_04\_soln.sql.
2. Crie as bind variables basic\_percent e pf\_percent do tipo NUMBER.
3. Na seção executável do bloco PL/SQL atribua os valores 45 e 12 a basic\_percent e pf\_percent, respectivamente.
4. Encerre o bloco PL/SQL com "/" e exiba o valor das bind variables utilizando o comando PRINT.
5. Execute e salve seu arquivo de script como lab\_02\_05\_soln.sql.

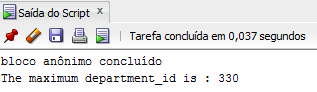
Na seção executável do bloco PL/SQL atribua os valores 45 e 12 a basic\_percent e pf\_percent, respectivamente.

Encerre o bloco PL/SQL com "/" e exibir o valor das variáveis de ligação usando o comando PRINT.

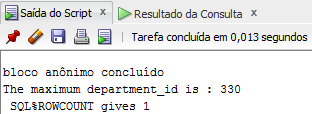
Execute e salve seu arquivo de script como lab\_02\_05\_soln.sql.

**Cap 03-Instruções DML no PL/SQL**

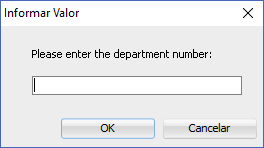
1. Crie um bloco PL/SQL que seleciona o número máximo de departamento na tabela DEPARTMENTS e o armazena em uma variável. Imprima os resultados na tela. Salve o bloco PL/SQL em um arquivo nomeado lab\_04\_01\_soln.sql.



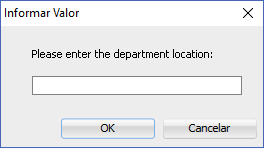
1. Modifique o bloco PL/SQL que você criou no exercício 1 (lab\_04\_01\_soln.sql) para inserir um novo departamento na tabela DEPT. Salve o bloco PL/SQL em um arquivo nomeado (lab\_04\_02\_soln.sql).
2. Em vez de imprimir o número do departamento recuperado do exercício 1, adicione 10 a ele (Se o número retornado foi 330 então adicione 10) e use-o como o número do departamento do novo departamento. O novo departamento deve ser definido como: EDUCATION
3. Use uma variável de substituição do tipo BIND VARIABLE para definir o número do departamento novo departamento.
4. Deixe um valor nulo nas colunas como MANAGER\_ID e como o LOCATION\_ID do novo departamento.
5. Execute o bloco PL/SQL.
6. Exiba as seguintes mensagens como a seguir demonstrado. (Utilize o atributo de cursor SQL%ROWCOUNT para exibir a quantidade de linhas manipuladas.)



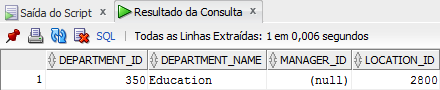
1. Crie um bloco PL/SQL que atualize a localização (location\_id) para um departamento existente (department\_id) (Tabela: DEPARTMENTS). Salve o bloco PL/SQL em um arquivo denominado lab\_04\_03\_soln.sql*.*
2. Use uma variável de substituição (VARIABLE ou ACCEPT) para informar o número de departamento.



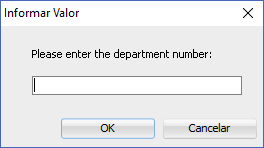
1. Use uma variável de substituição (VARIABLE ou ACCEPT) para informar a localização de departamento. (Consulte a tabela LOCATIONS para ter um número de localização válido)



1. Teste o bloco PL/SQL.
2. Exiba o nome e o número do departamento, além da localização do departamento atualizado.



1. Crie um bloco PL/SQL que delete o departamento criado no exercício 2. Salve o bloco PL/SQL em um arquivo denominado lab\_04\_04\_soln.sql*.*
2. Use uma variável de substituição (VARIABLE ou ACCEPT) para informar para o número do departamento.



1. Imprima o número de linhas afetadas na tela. (Para isso utilize uma BIND VARIABLE e exiba essa mensagem após o teste do bloco. Não use dbms\_output.put\_line)
2. Teste o bloco PL/SQL.
3. O que acontece se você informar um número de departamento inexistente?
4. Confirme que o departamento foi deletado.

**Cap 04-Estruturas de Controle**

1. Execute o comando a seguir para criar a tabela messages.

CREATE TABLE MESSAGES

(TEXT VARCHAR(50));

a. Crie um bloco PL/SQL que insira na tabela MESSAGES os números de 1 até 10, excluindo os números 6 e 8.

Após a execução do bloco faça SELECT na tabela MESSAGES para verificar se os números foram inseridos corretamente.

1. Antes de iniciar execute o script a seguir:

Utilize o script a seguir para criar a tabela EMP que será uma cópia da tabela EMPLOYEES

drop table emp;

create table emp

as select \* from employees;

alter table emp

add stars varchar(50);

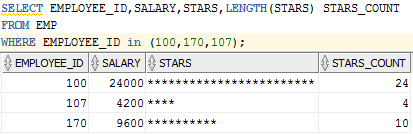
Crie um bloco PL/SQL que premie um funcionário, anexando um asterisco à coluna STARS da tabela EMP para cada US$1000 do salário do funcionário.**(2,5 pontos)**

* Se o funcionário recebe um salário (salary) de US$24000, a *string* de asteriscos deve conter 24 (vinte e quatro) asteriscos.
* Se o funcionário recebe um salário (salary) de US$9600, a *string* de asteriscos deve conter 10 (dez) asteriscos.
* Se o funcionário recebe um salário (salary) de US$4200, a *string* de asteriscos deve conter 4 (quatro) asteriscos.

Atualize a coluna STARS da tabela de funcionários (EMP) com a string de asteriscos, de acordo com a regra anterior.

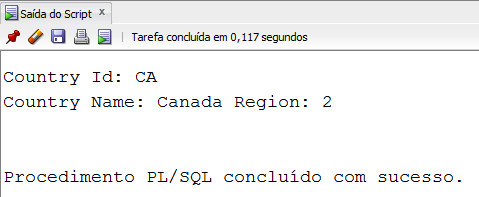
Faça uma atualização por vez, então considere que o código do funcionário deve ser informado.

Exemplos de seleção de funcionários:



**Cap 05-Registros PL/SQL**

1. Escreva um bloco PL/SQL para imprimir informações sobre um determinado país. Declare um registro PL/SQL baseado na estrutura da tabela de countries.
2. Use o comando DEFINE para definir uma variável c **countryid**. Atribua CA a countryid. Passe o valor para o bloco PL/SQL através de uma variável de substituição.
3. Na seção declare, use o atributo%ROWTYPE e declare a variável country\_record de tipo countries.
4. Na seção executável, obter todas as informações da tabela **countries** usando countryid. Mostra as informações selecionadas sobre o país.

****

**Cap 06-Criando Cursores Explícitos**

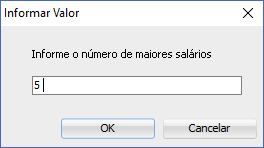
Execute o script a seguir para criar uma nova tabela de armazenamento de funcionários e salários.

**CREATE TABLE top\_dogs**

**(name VARCHAR2(25),**

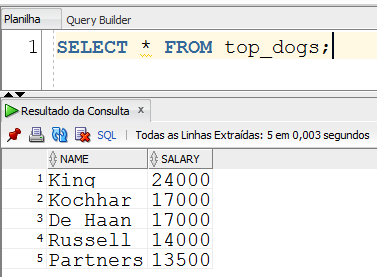
**salary NUMBER(11,2));**

1. Crie um bloco PL/SQL que determine os funcionários com os maiores salários.
2. Aceite um número *n* como entrada de usuário com um parâmetro de substituição.

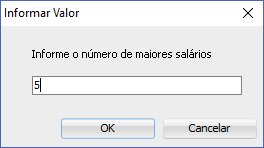


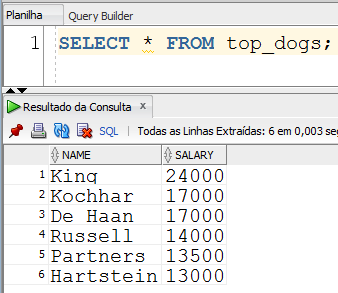
1. Em um loop, obtenha os sobrenomes (last\_name) e salários (salary) dos n funcionários com os maiores salários na tabela EMPLOYEES.
2. Armazene os nomes e os salários na tabela TOP\_DOGS.
3. Suponha que nenhum dos funcionários tenha salário igual ao do outro.
4. Teste vários casos especiais, como *n* = 0 ou em que *n* seja maior que o número de funcionários na tabela EMPLOYEES.

Esvazie a tabela TOP\_DOGS depois de cada teste (TRUNCATE TABLE TOP\_DOGS;).

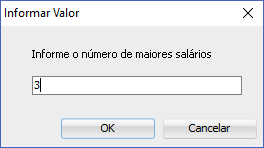


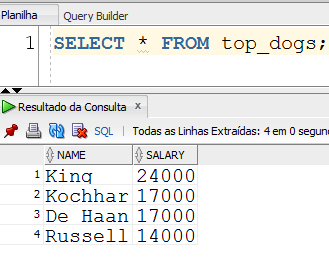
1. Considere o caso em que vários funcionários recebem o mesmo salário. Se uma pessoa estiver listada, todas as pessoas com o mesmo salário também deverão estar listadas.
2. Por exemplo, se o usuário informar um valor 5 para *n*, King, Kochhar, De Haan, Russell, Partners e Hartstein deverão ser exibidos.





1. Se o usuário informar um valor 3, King, Kochhar, De Haan e Russell deverão ser exibidos.





1. Delete todas as linhas da tabela TOP\_DOGS e teste o exercício.

###### Cap 07-Tratando Exceções

###### Crie um bloco PL/SQL para selecionar o nome do funcionário com um determinado salário.

CREATE TABLE messages

(results VARCHAR(200));

1. Se o salário informado retornar mais de uma linha, trate a exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "More than one employee with a salary of <*salário*>".
2. Se o salário informado não retornar qualquer linha, trate a exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "No employee with a salary of <*salário>*".
3. Se o salário informado retornar apenas uma linha, insira, na tabela MESSAGES, o nome do funcionário e o valor do salário.
4. Trate qualquer outra exceção com um handler de exceção apropriado e insira, na tabela MESSAGES, a mensagem "Some other error occurred".
5. Teste o bloco para vários casos.

**Cap 08-Procedures**

1. Crie uma procedure chamada CAD\_EMPLOYEES que realiza o cadastramento de um FUNCIONARIO. Para isso utilize a tabela EMPLOYEES. Crie as exceções necessárias (pelo menos duas) para o funcionamento correto desse procedimento.

Utilize a estrutura a seguir para a tabela de empregados:

Nome Nulo? Tipo

-------------- -------- ------------

EMPLOYEE\_ID NOT NULL NUMBER(6)

FIRST\_NAME VARCHAR2(20)

LAST\_NAME NOT NULL VARCHAR2(25)

EMAIL NOT NULL VARCHAR2(25)

PHONE\_NUMBER VARCHAR2(20)

HIRE\_DATE NOT NULL DATE

JOB\_ID NOT NULL VARCHAR2(10)

SALARY NUMBER(8,2)

COMMISSION\_PCT NUMBER(2,2)

MANAGER\_ID NUMBER(6)

DEPARTMENT\_ID NUMBER(4)

1. Crie uma procedure chamada CAD\_DEPARTMENTS Que realiza o cadastramento de um DEPARTAMENTO. Para isso utilize a tabela DEPARTMENTS. Crie as exceções necessárias (pelo menos duas) para o funcionamento correto desse procedimento.

Utilize a estrutura a seguir para a tabela de departamentos:

Nome Nulo? Tipo

--------------- -------- ------------

DEPARTMENT\_ID NOT NULL NUMBER(4)

DEPARTMENT\_NAME NOT NULL VARCHAR2(30)

MANAGER\_ID NUMBER(6)

LOCATION\_ID NUMBER(4)

**Cap 09-Functions**

1. Crie uma função chamada GET\_SALARY\_TAX para retornar o imposto descontado do salário mensal de um empregado. Utilize a tabela EMPLOYEES.
2. A função deve aceitar como parâmetro o código do funcionário .
3. A função deve retornar o valor do salário taxado

Use a fórmula a seguir para calcular a taxa anual:

(salary \* 0.08)

1. Crie uma função chamada GET\_ANNUAL\_COMP para retornar o salário anual calculado a partir do salário mensal de um empregado. Utilize a tabela EMPLOYEES.
2. A função deve aceitar como parâmetros de entrada o código do funcionário.
3. A função deve retornar o valor do salário taxado

OBSERVAÇÃO: O valor da comissão pode ser nulo pois existem funcionários que não são comissionados. A função deve retornar um salário anual não nulo.

Use a fórmula a seguir para calcular o salário anual:

(salary\*12) + (commission\_pct\*salary\*12)

**Cap 10-Packages**

1. Crie um pacote chamado PKG\_EXERCICIO que contenha a procedure CAD\_DEPARTMENTS e a procedure GET\_ANNUAL\_COMP. Teste a execução da procedure e da function que foram empacotadas

**Cap 11-Triggers**

Construa um TRIGGER que capture todas as alterações de linha (DML – INSERT, UPDATE e DELETE) na tabela chamada EMPLOYEES\_TRG (Cópia da Tabela EMPLOYEES) para conformidade com auditoria. Os registros de auditoria devem ser feitos na tabela AUDIT\_EMPLOYEES.

Criação da tabela de auditoria:

create table AUDIT\_EMPLOYEES

(AUD\_WHO VARCHAR2(20),

AUD\_WHEN DATE,

AUD\_OPERATION VARCHAR2(1),

AUD\_MODULE VARCHAR2(30),

EMPLOYEE\_ID NUMBER(6),

FIRST\_NAME VARCHAR2(20),

LAST\_NAME VARCHAR2(25) NOT NULL,

EMAIL VARCHAR2(25),

PHONE\_NUMBER VARCHAR2(20) NOT NULL,

HIRE\_DATE DATE NOT NULL,

JOB\_ID VARCHAR2(10) NOT NULL,

SALARY NUMBER(8,2),

COMMISSION\_PCT NUMBER(2,2),

MANAGER\_ID NUMBER(6),

DEPARTMENT\_ID NUMBER(4),

VACATION\_BALANCE NUMBER(6,2)

);

Criação da Cópia da tabela EMPLOYEES

CREATE TABLE employees\_trg

AS SELECT \*

FROM employees;

Utilize as seguintes instruções para testar o TRIGGER:

DELETE FROM employees\_trg

WHERE employee\_id = 200;

COMMIT;

INSERT INTO employees\_trg VALUES (200,'Jennifer','Whalen','JWHALEN','515.123.4444',TO\_DATE('17-SET-1987','dd-MON-yyyy'),'AD\_ASST',4400,NULL,101,10);

COMMIT;

SELECT \*

FROM audit\_employees;

UPDATE employees\_trg

SET phone\_number = '515.123.333'

WHERE employee\_id = 200;

COMMIT;

**Lista de Respostas**

**Cap 01-Introdução ao PL/SQL**

Alternativa D

SET SERVEROUTPUT ON

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

/

**Cap 02-Declarando Variáveis PL/SQL**

1. válida
2. válida
3. inválida
4. inválida
5. válida
6. inválida
7. válida
8. válida
9. inválida
10. inválida
11. válida

Alternativa c

SET SERVEROUTPUT ON

DECLARE

today DATE:=SYSDATE;

tomorrow today%TYPE;

BEGIN

tomorrow:=today +1;

DBMS\_OUTPUT.PUT\_LINE(' Hello World ');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS : '|| today);

DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS : ' || tomorrow);

END;

VARIABLE basic\_percent NUMBER

VARIABLE pf\_percent NUMBER

SET SERVEROUTPUT ON

DECLARE

today DATE:=SYSDATE;

tomorrow today%TYPE;

BEGIN

:basic\_percent:=45;

:pf\_percent:=12;

tomorrow:=today +1;

DBMS\_OUTPUT.PUT\_LINE(' Hello World ');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS : '|| today);

DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS : ' || tomorrow);

END;

/

PRINT basic\_percent

PRINT pf\_percent

**Cap 03-Instruções DML no PL/SQL**

SET SERVEROUTPUT ON

DECLARE

max\_deptno NUMBER;

BEGIN

SELECT MAX(department\_id) INTO max\_deptno FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is : ' || max\_deptno);

END;

VARIABLE dept\_id NUMBER

SET SERVEROUTPUT ON

DECLARE

dept\_name departments.department\_name%TYPE:= 'Education';

max\_deptno NUMBER;

BEGIN

SELECT MAX(department\_id) INTO max\_deptno FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is : ' ||

max\_deptno);

:dept\_id := 10 + max\_deptno;

INSERT INTO departments (department\_id, department\_name, location\_id)

VALUES (:dept\_id,dept\_name, NULL);

DBMS\_OUTPUT.PUT\_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);

END;

/

SELECT \* FROM departments WHERE department\_id=:dept\_id;

ACCEPT p\_department\_id PROMPT 'Please enter the department number'

ACCEPT p\_location\_id PROMPT 'Please entre the department location'

DECLARE

v\_department\_id departments.department\_id%TYPE:=&p\_department\_id;

v\_location\_id departments.location\_id%TYPE:=&p\_location\_id;

BEGIN

UPDATE departments

SET location\_id=v\_location\_id

WHERE department\_id=v\_department\_id;

END;

/

ACCEPT p\_department\_id PROMPT 'Please enter the department number'

DECLARE

v\_department\_id departments.department\_id%TYPE:=&p\_department\_id;

BEGIN

delete from departments

WHERE department\_id=v\_department\_id;

END;

/

**Cap 04-Estruturas de Controle**

BEGIN

FOR i in 1..10 LOOP

IF i = 6 or i = 8 THEN

null;

ELSE

INSERT INTO messages(results)

VALUES (i);

END IF;

END LOOP;

COMMIT;

END;

/

SELECT \* FROM messages;

ACCEPT p\_empno PROMPT 'Informe o número do funcionário: '

DECLARE

v\_empno emp.employee\_id%TYPE := TO\_NUMBER(&p\_empno);

v\_asterisk emp.stars%TYPE := NULL;

v\_sal emp.salary%TYPE;

BEGIN

SELECT NVL(ROUND(salary/1000), 0)

INTO v\_sal

FROM emp

WHERE employee\_id = v\_empno;

FOR i IN 1..v\_sal LOOP

v\_asterisk := v\_asterisk ||'\*';

END LOOP;

UPDATE emp

SET stars = v\_asterisk

WHERE employee\_id = v\_empno;

COMMIT;

END;

/

**Cap 05-Registros PL/SQL**

SET SERVEROUTPUT ON

SET VERIFY OFF

DEFINE countryid = CA

DECLARE

country\_record countries%ROWTYPE;

BEGIN

SELECT \*

INTO country\_record

FROM countries

WHERE country\_id = UPPER('&countryid');

DBMS\_OUTPUT.PUT\_LINE ('Country Id: ' || country\_record.country\_id ||

' Country Name: ' || country\_record.country\_name

|| ' Region: ' || country\_record.region\_id);

END;

**Cap 06-Criando Cursores Explícitos**

DROP TABLE top\_dogs;

CREATE TABLE hr.top\_dogs

(name VARCHAR2(25),

salary NUMBER(11,2));

ACCEPT p\_num PROMPT 'Informe o número de maiores salários '

DECLARE

v\_num NUMBER(3) := &p\_num;

v\_ename employees.last\_name%TYPE;

v\_sal employees.salary%TYPE;

CURSOR emp\_cursor IS

SELECT last\_name, salary

FROM employees

ORDER BY 2 DESC;

BEGIN

OPEN emp\_cursor;

FETCH emp\_cursor INTO v\_ename, v\_sal;

WHILE emp\_cursor%ROWCOUNT <= v\_num AND emp\_cursor%FOUND LOOP

INSERT INTO top\_dogs (name, salary)

VALUES (v\_ename, v\_sal);

FETCH emp\_cursor INTO v\_ename, v\_sal;

END LOOP;

CLOSE emp\_cursor;

COMMIT;

END;

/

SELECT \* FROM top\_dogs;

truncate table top\_dogs;

ACCEPT p\_num PROMPT 'Informe o número de maiores salários '

DECLARE

v\_num NUMBER(3) := &p\_num;

v\_ename employees.last\_name%TYPE;

v\_current\_sal employees.salary%TYPE;

v\_last\_sal v\_current\_sal%TYPE;

CURSOR emp\_cursor IS

SELECT last\_name, salary

FROM employees

ORDER BY 2 DESC;

BEGIN

OPEN emp\_cursor;

FETCH emp\_cursor INTO v\_ename, v\_current\_sal;

WHILE emp\_cursor%ROWCOUNT <= v\_num AND emp\_cursor%FOUND LOOP

INSERT INTO top\_dogs (name, salary)

VALUES (v\_ename, v\_current\_sal);

v\_last\_sal := v\_current\_sal;

FETCH emp\_cursor INTO v\_ename, v\_current\_sal;

IF v\_last\_sal = v\_current\_sal THEN

v\_num := v\_num + 1;

END IF;

END LOOP;

CLOSE emp\_cursor;

COMMIT;

END;

/

SELECT \* FROM top\_dogs;

**Cap 07-Tratando Exceções**

DELETE FROM MESSAGES;

SET VERIFY OFF

DEFINE sal = 6000

DECLARE

ename employees.last\_name%TYPE;

emp\_sal employees.salary%TYPE := &sal;

BEGIN

SELECT last\_name

INTO ename

FROM employees

WHERE salary = emp\_sal;

INSERT INTO messages (results)

VALUES (ename || ' - ' || emp\_sal);

EXCEPTION

WHEN no\_data\_found THEN

INSERT INTO messages (results)

VALUES ('No employee with a salary of '|| TO\_CHAR(emp\_sal));

WHEN too\_many\_rows THEN

INSERT INTO messages (results)

VALUES ('More than one employee with a salary of '||

TO\_CHAR(emp\_sal));

WHEN others THEN

INSERT INTO messages (results)

VALUES ('Some other error occurred.');

END;

/

SELECT \* FROM messages;

**Cap 08-Procedures**

create or replace procedure CAD\_EMPLOYEES

(P\_EMPLOYEE\_ID NUMBER,

P\_FIRST\_NAME VARCHAR2,

P\_LAST\_NAME VARCHAR2,

P\_EMAIL VARCHAR2,

P\_PHONE\_NUMBER VARCHAR2,

P\_HIRE\_DATE DATE,

P\_JOB\_ID VARCHAR2,

P\_SALARY NUMBER,

P\_COMMISSION\_PCT NUMBER,

P\_MANAGER\_ID NUMBER,

P\_DEPARTMENT\_ID NUMBER

)

is

BEGIN

INSERT INTO EMPLOYEES

VALUES (P\_EMPLOYEE\_ID,P\_FIRST\_NAME,P\_LAST\_NAME,P\_EMAIL,P\_PHONE\_NUMBER,P\_HIRE\_DATE,P\_JOB\_ID,

P\_SALARY, P\_COMMISSION\_PCT,P\_MANAGER\_ID,P\_DEPARTMENT\_ID);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Empregado já cadastrado');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ocorreu um erro'|| SQLERRM);

END ;

/

create or replace procedure CAD\_DEPARTMENTS

(P\_DEPARTMENT\_ID NUMBER ,

P\_DEPARTMENT\_NAME VARCHAR2 ,

P\_MANAGER\_ID NUMBER,

P\_LOCATION\_ID NUMBER)

is

BEGIN

INSERT INTO DEPARTMENTS

VALUES (P\_DEPARTMENT\_ID,P\_DEPARTMENT\_NAME,P\_MANAGER\_ID,P\_LOCATION\_ID);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Departamento já cadastrado');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ocorreu um erro'|| SQLERRM);

END ;

**Cap 09-Functions**

create or replace function get\_salary\_tax (

p\_ei employees.employee\_id%type)

return number

is

v\_sal employees.salary%type;

begin

select salary \* 0.8

into v\_sal

from employees

where employee\_id = p\_ei;

return v\_sal;

end get\_salary\_tax;

/

select employee\_id,last\_name,salary,get\_salary\_tax(employee\_id)

from employees;

create or replace function GET\_ANNUAL\_COMP

(p\_ei employees.employee\_id%type)

return number

is

v\_saltax employees.salary%type;

begin

select (salary\*12) + (nvl(commission\_pct,0)\*salary\*12)

into v\_saltax

from employees

where employee\_id=p\_ei;

return v\_saltax;

end GET\_ANNUAL\_COMP;

/

select employee\_id,last\_name,salary,GET\_ANNUAL\_COMP(employee\_id) ann\_sal

from employees

where department\_id=90;

**Cap 10-Packages**

create or replace package PKG\_EXERCICIO is

procedure CAD\_DEPARTMENTS

(P\_DEPARTMENT\_ID NUMBER ,

P\_DEPARTMENT\_NAME VARCHAR2 ,

P\_MANAGER\_ID NUMBER,

P\_LOCATION\_ID NUMBER);

FUNCTION get\_annual\_comp(

sal IN employees.salary%TYPE,

comm IN employees.commission\_pct%TYPE)

RETURN NUMBER;

end;

/

create or replace package body PKG\_EXERCICIO is

procedure CAD\_DEPARTMENTS

(P\_DEPARTMENT\_ID NUMBER ,

P\_DEPARTMENT\_NAME VARCHAR2 ,

P\_MANAGER\_ID NUMBER,

P\_LOCATION\_ID NUMBER)

is

BEGIN

INSERT INTO DEPARTMENTS

VALUES (P\_DEPARTMENT\_ID,P\_DEPARTMENT\_NAME,P\_MANAGER\_ID,P\_LOCATION\_ID);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Departamento já cadastrado');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Ocorreu um erro'|| SQLERRM);

END ;

FUNCTION hr.get\_annual\_comp(

sal IN employees.salary%TYPE,

comm IN employees.commission\_pct%TYPE)

RETURN NUMBER IS

BEGIN

RETURN (NVL(sal,0) \* 12 + (NVL(comm,0) \* nvl(sal,0) \* 12));

END get\_annual\_comp;

end;

/

**Cap 11-Triggers**

create trigger TRG\_AUDIT\_EMPLOYEES;

after insert or update or delete on EMPLOYEES

for each row

declare

l\_operation varchar2(1) :=

case when updating then 'U'

when deleting then 'D'

else 'I' end;

begin

if updating or inserting then

insert into AUDIT\_EMPLOYEES

(aud\_who

,aud\_when

,aud\_operation

,aud\_module

,employee\_id

,first\_name

,last\_name

,email

,phone\_number

,hire\_date

,job\_id

,salary

,commission\_pct

,manager\_id

,department\_id)

values

(user

,sysdate

,l\_operation

,sys\_context('USERENV','MODULE')

,:new.employee\_id

,:new.first\_name

,:new.last\_name

,:new.email

,:new.phone\_number

,:new.hire\_date

,:new.job\_id

,:new.salary

,:new.commission\_pct

,:new.manager\_id

,:new.department\_id);

else

insert into AUDIT\_EMPLOYEES

(aud\_who

,aud\_when

,aud\_operation

,aud\_module

,employee\_id

,first\_name

,last\_name

,email

,phone\_number

,hire\_date

,job\_id

,salary

,commission\_pct

,manager\_id

,department\_id)

values

(user

,sysdate

,l\_operation

,sys\_context('USERENV','MODULE')

,:old.employee\_id

,:old.first\_name

,:old.last\_name

,:old.email

,:old.phone\_number

,:old.hire\_date

,:old.job\_id

,:old.salary

,:old.commission\_pct

,:old.manager\_id

,:old.department\_id);

end if;

end;

/