Documentação programação avançada

de banco de dados com PL/SQL

Prof. Luis Alexandre

Nome: Raziel Miranda Rodrigues

ÍNDICE

1 – TABLES E SEQUENCES

2 – FUNÇÕES

3 – CHAMADA FUNÇÕES E FOR

5 – TRIGGER

TABLES E SEQUENCES:

CREATE TABLE PERSONAGEM(

ID NUMBER PRIMARY KEY NOT NULL,

NOME VARCHAR2(120) NOT NULL,

SALARIO NUMBER(10,2) NOT NULL

);

INSERT INTO PERSONAGEM(ID,NOME,SALARIO)

VALUES (SEQ\_UPD\_PERSONAGEM.NEXTVAL,'RAZIEL MIRANDA',8600.76);

CREATE TABLE LOG\_PERSONAGEM(

ID\_LOG NUMBER PRIMARY KEY,

ID NUMBER NOT NULL,

SALARIO\_VELHO NUMBER(10,2) NOT NULL,

SALARIO\_NOVO NUMBER(10,2) NOT NULL,

USUARIO VARCHAR2(30) NOT NULL,

DATAHORA DATE NOT NULL,

CONSTRAINT FK\_PER\_LOG FOREIGN KEY (ID) REFERENCES PERSONAGEM(ID)

);

COMMIT;

CREATE SEQUENCE SEQ\_UPD\_PERSONAGEM

START WITH 4

INCREMENT BY 1

NOCACHE

NOCYCLE;

CREATE TABLE LOG\_ERRO\_EXECUCAO(

ID\_LOG NUMBER PRIMARY KEY,

NOME\_CODIGO VARCHAR2(30) NOT NULL,

PARAMETROS VARCHAR(4000) NULL,

NUMERO\_ERRO VARCHAR2(30) NOT NULL,

DESCRICAO\_ERRO VARCHAR2(4000) NOT NULL,

NUMERO\_LINHA NUMBER NOT NULL,

USUARIO VARCHAR2(30),

DATA DATE NOT NULL,

STATUS CHAR(1) NOT NULL,

DESCRICAO\_SOLUCAO VARCHAR2(4000),

CONSTRAINT CK\_STATUS\_LOGPROGEXE CHECK (STATUS IN('E', 'R'))

);

CREATE SEQUENCE SEQ\_LOG\_PROGEXEC

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

FUNÇÕES

CREATE OR REPLACE FUNCTION FUNC\_AUMENTOSALPERSONAGEM(P\_ID IN NUMBER, P\_SAL IN NUMBER)

RETURN NUMBER

IS

VQTDE NUMBER(1);

RET NUMBER(3) := 0;

NOV\_SAL PERSONAGE.SALARIO%TYPE;

BEGIN

IF P\_SAL > 0 AND P\_SAL < 99999999.99 THEN

SELECT COUNT(\*) INTO VQTDE FROM PERSONAGEM WHERE ID = P\_ID;

IF VQTDE = 1 THEN

SELECT SALARIO INTO NOV\_SAL FROM PERSONAGEM WHERE ID = P\_ID;

UPDATE PERSONAGEM SET SALARIO = P\_SAL WHERE ID = P\_ID;

INSERT INTO LOG\_PERSONAGEM(ID\_LOG,ID,SALARIO\_VELHO,SALARIO\_NOVO,USUARIO,DATAHORAL)

VALUES(SEQ\_UPD\_PERSONAGEM.NEXTVAL,P\_ID,NOV\_SAL,P\_SAL,USER,SYSDATE);

ELSE

RET := -998;

END IF;

ELSE

RET := -999;

END IF;

COMMIT;

RETURN RET;

END;

CREATE OR REPLACE FUNCTION FUNC\_DIVISAO (PDIVISOR IN NUMBER, PDIVIDENDO IN NUMBER)

RETURN NUMBER

IS

VRETORNO NUMBER;

VPARAMETROS VARCHAR2(4000);

VNUMERO\_CODIGO NUMBER(6);

VNOME\_CODIGO VARCHAR2(4000);

VNOME\_PROC VARCHAR(30) := $$PLSQL\_UNIT;

VNUMLINHA NUMBER;

BEGIN

VRETORNO := PDIVISOR / PDIVIDENDO;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

VPARAMETROS := 'PDIVISOR: '||PDIVISOR||', PDIVIDENDO: '||PDIVIDENDO;

VNUMERO\_CODIGO := SQLCODE;

VNOME\_CODIGO := SQLERRM;

VNUMLINHA := $$PLSQL\_LINE;

INSERT INTO LOG\_ERRO\_EXECUCAO(ID\_LOG, NOME\_CODIGO, PARAMETROS, NUMERO\_ERRO, DESCRICAO\_ERRO, NUMERO\_LINHA, USUARIO, DATA, STATUS)

VALUES(SEQ\_LOG\_PROGEXEC.NEXTVAL, VNOME\_PROC,VPARAMETRO, VNUMERO\_CODIGO, VNOME\_CODIGO, VNUMLINHA, USER, SYSDATE, 'E');

COMMIT;

RETURN SQLCODE;

END;

CREATE OR REPLACE FUNCTION FUNC\_RET\_NAME\_DEPT (P\_ID IN NUMBER)

RETURN VARCHAR2

IS

RETORNO DEPARTMENTS.DEPARTMENT\_NAME%TYPE;

QTDE NUMBER(1);

BEGIN

IF P\_ID > 0 AND P\_ID <= 999999 THEN

SELECT COUNT(\*) INTO QTDE FROM EMPLOYEES WHERE EMPLOYEE\_ID = P\_ID;

IF QTDE = 1 THEN

SELECT DEPARTMENT\_NAME INTO RETORNO

FROM EMPLOYEES E

LEFT JOIN DEPARTMENTS D

ON E.DEPARTMENT\_ID = D.DEPARTMENT\_ID

WHERE EMPLOYEE\_ID = P\_ID;

ELSE

RETORNO := 'NÃO HÁ FUNCIONARIO';

END IF;

ELSE

RETORNO := 'ID FORA DA FAIXA';

END IF;

RETURN RETORNO;

END;

CREATE OR REPLACE FUNCTION FUNC\_NOM\_JOB (P\_ID IN NUMBER)

RETURN VARCHAR2

IS

RETORNO JOBS.JOB\_TITLE%TYPE;

QTDE NUMBER(1);

BEGIN

IF P\_ID > 0 AND P\_ID <= 999999 THEN

IF QTDE = 1 THEN

SELECT JOB\_TITLE

FROM JOBS J

INNER JOIN EMPLOYEES E

ON J.job\_id = E.job\_id

WHERE EMPLOYEE\_ID = P\_ID;

ELSE

RETORNO := 'NÃO HÁ FUNCIONARIO NESSE ID';

END IF

ELSE

RETORNO := 'ID FORA DA FAIXA';

END IF

RETURN RETORNO;

END;

CHAMANDO FUNÇÕES E FOR:

-- CHAMANDO FUNÇÃO NOME DO DEPARTAMENTO, PARA AS OUTRAS É O MESMO ESQUEMA

DECLARE

NAME\_DEPT VARCHAR(4000);

BEGIN

NAME\_DEPT:=FUNC\_RET\_NAME\_DEPT(10000000);

DBMS\_OUTPUT.PUT\_LINE('DEPARTMENT NAME: ' || NAME\_DEPT);

END;

-- SELECT PARA USAR NO FOR

BEGIN

FOR RS IN (SELECT EMPLOYEE\_ID AS CODIGO, FIRST\_NAME, SALARY

FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 90 ORDER BY FIRST\_NAME DESC)

LOOP

DBMS\_OUTPUT.PUT\_LINE('NOME: ' || RS.FIRST\_NAME);

DBMS\_OUTPUT.PUT\_LINE('ID: ' || RS.CODIGO);

END LOOP;

END;

TRIGGER

create or replace trigger trg\_pers\_upsal

before update

of salario

on personagem

for each row

begin

if updating then

insert into log\_personagem(id\_log,salario\_velho, salario\_novo, usuario,operacao ,datahora)

values(:old.id,:old.salario,:new.salario,user,'update',sysdate);

end if;

if inserting then

insert into log\_personagem(id\_log,salario\_velho, salario\_novo, usuario, operacao,datahora)

values(:new.id,:old.salario,:new.salario,user,'insert',sysdate);

end if;

if deleting then

insert into log\_personagem(id\_log,salario\_velho, salario\_novo, usuario, operacao,datahora)

values(:old.id,:old.salario,:new.salario,user,'delete',sysdate);

end if;

end;