UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

LICENCIATURA EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN

SISTEMAS DE BASES DE DATOS II

LABORATORIO No. 7

FUNDAMENTOS DEL LENGUAJE-CURSORES-PROCEDIMIENTOS Y FUNCIONES y TRIGGERS ‘APLICAR PROGRAMACION ALMACENADA DE BASE DE DATOSPL/SQL ORACLE’.

Integrantes:

ANDREINA GÓMEZ 8-939-1682

STEFANIE AROSEMENA 8-885-1747

EMANOL GONZALEZ 3-745-1637

JOSÉ QUINTERO 8-952-698

DOCENTE: ING. HENRY J. LEZCANO P.

GRUPO: 1IF121

FECHA: 16 NOVIEMBRE.

AÑO 2020.

1. Agregar una tabla al modelo físico que almacenes las sucursales de la empresa la financiera con las restricciones correspondientes:

Create Table Suc_Ahorro(

Cod_tipoahorro varchar2(5) not null,

constraint cod_tipoahorro2_fk foreign key (Cod_tipoahorro) references tipo_ahorros(Cod_tipoahorro),

cod_sucursal varchar2(2),

constraint cod_sucursal_fk2 foreign key (cod_sucursal) References sucursales (cod_sucursal),

monto_ahorrado number,

constraint pk_Suc_Ahorro primary key (Cod_tipoahorro, cod_sucursal));

```
SQL> Create Table Suc_Ahorro(
  2     Cod_tipoahorro varchar2(5) not null,
  3     constraint cod_tipoahorro2_fk foreign key (Cod_tipoahorro) r
eferences tipo_ahorros(Cod_tipoahorro),
  4     cod_sucursal varchar2(2),
  5     constraint cod_sucursal_fk2 foreign key (cod_sucursal) Refer
ences sucursales (cod_sucursal),
  6     monto_ahorrado number,
  7     constraint pk_Suc_Ahorro primary key (Cod_tipoahorro, cod_su
cursal));

Table created.
```

2. Agregar la tabla tipos de ahorros donde los tipos cuyos atributos son tipo de ahorro, descripción, y tasa de interés.

Create table tipo_ahorros(

Cod_tipoahorro varchar2(5) primary key not null,

Descrip_ahorro varchar2(20),

Interes_ahorro number);

```
SQL> Create table tipo_ahorros(
  2     Cod_tipoahorro varchar2(5) primary key not null,
  3     Descrip_ahorro varchar2(20),
  4     Interes_ahorro number);

Table created.
```

Procedimiento almacenado para la carga o inserción de la tabla paramétrica

```
CREATE OR REPLACE PROCEDURE crear_Tipoahorros (

p_cod_tipoahorro tipo_ahorros.Cod_tipoahorro%TYPE,

p_descrip_ahorro tipo_ahorros.Descrip_ahorro%TYPE,

p_interes_ahorro tipo_ahorros.Interes_ahorro%TYPE

)

AS

BEGIN

Insert into tipo_ahorros values(

p_cod_tipoahorro, p_descrip_ahorro, p_interes_ahorro);

EXCEPTION WHEN DUP_VAL_ON_INDEX THEN

        DBMS_OUTPUT.PUT_LINE(‘ NO PUEDE DUPLICAR DATOS…’);

COMMIT;
```

END crear_Tipoahorros;

/

```
SQL> CREATE OR REPLACE PROCEDURE crear_Tipoahorros (
  2  p_cod_tipoahorro tipo_ahorros.Cod_tipoahorro%TYPE,
  3  p_descrip_ahorro tipo_ahorros.Descrip_ahorro%TYPE,
  4  p_interes_ahorro tipo_ahorros.Interes_ahorro%TYPE
  5  )
  6  AS
  7  BEGIN
  8  Insert into tipo_ahorros values(
  9  p_cod_tipoahorro, p_descrip_ahorro, p_interes_ahorro);
 10  EXCEPTION WHEN DUP_VAL_ON_INDEX THEN
 11  DBMS_OUTPUT.PUT_LINE(‘ NO PUEDE DUPLICAR DATOS…’);
 12  COMMIT;
 13  END crear_Tipoahorros;
 14  /

Procedure created.
```

SET SERVEROUTPUT ON

Begin

crear_Tipoahorros(‘ca1’,’ahorro de navidad’, 0.01);

crear_Tipoahorros(‘ca2’,’ahorro corriente’, 0.09);

crear_Tipoahorros(‘ca3’,’ahorro personal’, 0.12);

End;

/

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL>  Begin
  2      crear_Tipoahorros('ca1','ahorro de navidad', 0.01);
  3      crear_Tipoahorros('ca2','ahorro corriente', 0.09);
  4      crear_Tipoahorros('ca3','ahorro personal', 0.12);
  5      End;
  6      /

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.
```

Select * from tipo_ahorros;

```
SQL> select * from  tipo_ahorros;

COD_T DESCRIP_AHORRO       INTERES_AHORRO
----- -------------------- --------------
ca1   ahorro de navidad               .01
ca2   ahorro corriente                .09
ca3   ahorro personal                 .12
```

3. Agregar la tabla de ahorros al modelo de base de datos cuyos atributos serán:

Create table Ahorros (

cod_sucursal varchar2(2),

constraint cod_sucursal2_fk foreign key(cod_sucursal) references sucursales(cod_sucursal),

id_cliente varchar2(10),

constraint ide_cliente_ahorro_fk foreign key(id_cliente) references Cliente(id_cliente),

cod_tipoahorro varchar2(5),

```
    constraint cod_tipoahorro_fk foreign key(cod_tipoahorro) references
tipo_ahorros (cod_tipoahorro),
    no_cuenta_ahorro number,
    fecha_apertura date,
    tasa_interesahorro number,
    letra_mensualahorro number,
    saldo_ahorro number,
    saldo_interesahorro number,
    fecha_depositoahorro date,
    fecha_retiroahorro date,
    id_usuario varchar2(5),
    constraint ide_usuario_fk foreign key (id_usuario) references
usuarios (id_usuario),
    fecha_modificacion_ahorro date not null,
    primary key (no_cuenta_ahorro, id_cliente, cod_tipoahorro));
```

```
SQL> Create table Ahorros (
  2     cod_sucursal varchar2(2),
  3     constraint cod_sucursal2_fk foreign key(cod_sucursal) refere
nces sucursales(cod_sucursal),
  4     id_cliente varchar2(10),
  5     constraint ide_cliente_ahorro_fk foreign key(id_cliente) ref
erences Cliente(id_cliente),
  6      cod_tipoahorro varchar2(5),
  7     constraint cod_tipoahorro_fk foreign key(cod_tipoahorro) ref
erences tipo_ahorros (cod_tipoahorro),
  8     no_cuenta_ahorro number,
  9     fecha_apertura date,
 10     tasa_interesahorro number,
 11     letra_mensualahorro number,
 12     saldo_ahorro number,
 13     saldo_interesahorro number,
 14     fecha_depositoahorro date,
 15     fecha_retiroahorro date,
 16     id_usuario varchar2(5),
 17     constraint ide_usuario_fk foreign key (id_usuario) reference
s usuarios (id_usuario),
 18     fecha_modificacion_ahorro date not null,
 19     primary key (no_cuenta_ahorro, id_cliente, cod_tipoahorro));


Table created.
```

****

****

****

Create table tipo_transaccionbanc (

cod_tipotransac varchar2(5) primary key not null,

tipo_transaccion varchar2(20)

);

```
SQL> Create table tipo_transaccionbanc (
  2  cod_tipotransac varchar2(5) primary key not null,
  3  tipo_transaccion varchar2(20)
  4  );

Table created.
```

----- >

```
CREATE OR REPLACE PROCEDURE crear_tipotransaccion (
  p_cod_tipotransa tipo_transaccionbanc.cod_tipotransac%TYPE,
  p_tipo_transaccion tipo_transaccionbanc.tipo_transaccion%TYPE)
  AS
  BEGIN
  Insert into tipo_transaccionbanc values(
  p_cod_tipotransa, p_tipo_transaccion);
  EXCEPTION WHEN DUP_VAL_ON_INDEX THEN
  DBMS_OUTPUT.PUT_LINE(‘ NO PUEDE DUPLICAR …’);
  COMMIT;
  END crear_tipotransaccion;
  /
```

```
SQL> CREATE OR REPLACE PROCEDURE crear_tipotransaccion (
  2     p_cod_tipotransa tipo_transaccionbanc.cod_tipotransac%TYPE,
  3     p_tipo_transaccion tipo_transaccionbanc.tipo_transaccion%TYP
E)
  4     AS
  5     BEGIN
  6     Insert into tipo_transaccionbanc values(
  7     p_cod_tipotransa, p_tipo_transaccion);
  8     EXCEPTION WHEN DUP_VAL_ON_INDEX THEN
  9     DBMS_OUTPUT.PUT_LINE(‘ NO PUEDE DUPLICAR …’);
 10     COMMIT;
 11     END crear_tipotransaccion;
 12     /

Procedure created.
```

SET SERVEROUTPUT ON

Begin

crear_tipotransaccion(‘tt1’, ‘Abono’);

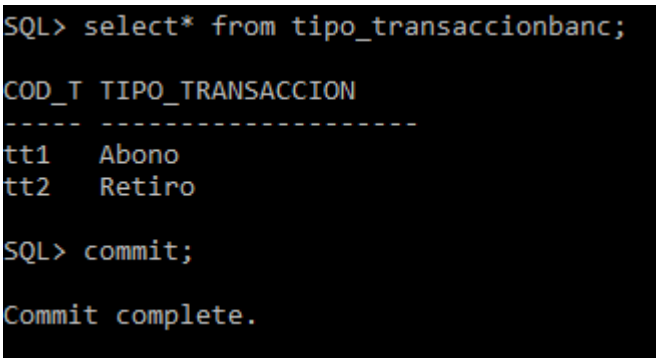
crear_tipotransaccion(‘tt2’, ‘Retiro’);

End;

/

```
SQL> SET SERVEROUTPUT ON
SQL> Begin
  2  crear_tipotransaccion(‘tt1’, ‘Abono’);
  3  crear_tipotransaccion(‘tt2’, ‘Retiro’);
  4  End;
  5  /

PL/SQL procedure successfully completed.
```



****

4. Agregar una tabla de auditoria que llevara la trazabilidad transaccional de cuentas de ahorros.

Create table auditoria_cahorros (

Id_transaccion number,

tabla_audit varchar2(25),

id_cliente varchar2(10),

constraint id_clienteaudit_fk foreign key(id_cliente) references Cliente(id_cliente),

Cod_tipoahorro varchar2(5),

constraint cod_tipoahorroaudit_fk foreign key(Cod_tipoahorro) references tipo_ahorros(Cod_tipoahorro),

cod_tipotransac varchar2(5),

constraint cod_tipotransac_fk foreign key(cod_tipotransac) references tipo_transaccionbanc(cod_tipotransac),

saldo_ahorro number,

saldo_anterior number,

Saldo_total number,

id_usuario varchar2(5),

constraint ide_usuario2_fk foreign key (id_usuario) references usuarios (id_usuario),

fecha_transaccion date,

primary key (Cod_tipoahorro, id_cliente, id_usuario));

```
SQL> Create table auditoria_cahorros (
  2     Id_transaccion number,
  3     tabla_audit varchar2(25),
  4     id_cliente varchar2(10),
  5     constraint id_clienteaudit_fk foreign key(id_cliente) refere
nces Cliente(id_cliente),
  6     Cod_tipoahorro varchar2(5),
  7     constraint cod_tipoahorroaudit_fk foreign key(Cod_tipoahorro
) references tipo_ahorros(Cod_tipoahorro),
  8     cod_tipotransac varchar2(5),
  9     constraint cod_tipotransac_fk foreign key(cod_tipotransac) r
eferences tipo_transaccionbanc(cod_tipotransac),
 10     saldo_ahorro number,
 11     saldo_anterior number,
 12     Saldo_total number,
 13    id_usuario varchar2(5),
 14     constraint ide_usuario2_fk foreign key (id_usuario) referenc
es usuarios (id_usuario),
 15     fecha_transaccion date,
 16     primary key (Cod_tipoahorro, id_cliente, id_usuario));

Table created.
```

5. Agregar una tabla transaccional para recibir los depósitos y retiros de los clientes a sus cuentas de ahorros.

Create table transadeporeti (

cod_sucursal varchar2(2),

constraint cod_sucursal3_fk foreign key(cod_sucursal) references sucursales(cod_sucursal),

```
   id_transacciondeporeti number,
   id_cliente varchar2(10),
   constraint id_clientetrans_fk foreign key(id_cliente) references
Cliente(id_cliente),
   Cod_tipoahorro varchar2(5),
   constraint cod_tipoahorrotransa_fk foreign key(Cod_tipoahorro)
references tipo_ahorros(Cod_tipoahorro),
   cod_tipotransac varchar2(5),
   constraint cod_tipotransac2_fk foreign key(cod_tipotransac)
references tipo_transaccionbanc(cod_tipotransac),
   monto_transac number,
   fecha_transaccion date,
   fecha_insercion date,
   id_usuario varchar2(5),
   constraint ide_usuario3_fk foreign key (id_usuario) references
usuarios (id_usuario),
  primary key(cod_sucursal, id_cliente,cod_tipoahorro,
fecha_transaccion, id_usuario));
```

```
SQL>  Create table transadeporeti (
  2     cod_sucursal varchar2(2),
  3     constraint cod_sucursal3_fk foreign key(cod_sucursal) refere
nces sucursales(cod_sucursal),
  4     id_transacciondeporeti number,
  5     id_cliente varchar2(10),
  6     constraint id_clientetrans_fk foreign key(id_cliente) refere
nces Cliente(id_cliente),
  7     Cod_tipoahorro varchar2(5),
  8     constraint cod_tipoahorrotransa_fk foreign key(Cod_tipoahorr
o) references tipo_ahorros(Cod_tipoahorro),
  9     cod_tipotransac varchar2(5),
 10     constraint cod_tipotransac2_fk foreign key(cod_tipotransac)
references tipo_transaccionbanc(cod_tipotransac),
 11     monto_transac number,
 12     fecha_transaccion date,
 13     fecha_insercion date,
 14     id_usuario varchar2(5),
 15     constraint ide_usuario3_fk foreign key (id_usuario) referenc
es usuarios (id_usuario),
 16    primary key(cod_sucursal, id_cliente,cod_tipoahorro, fecha_tr
ansaccion, id_usuario));

Table created.
```

Procedimiento almacenado para la apertura o inserción de los ahorros aprobados con toda la información correspondiente.

CREATE SEQUENCE No_Cuenta

START WITH 1

INCREMENT BY 1;

```
SQL> CREATE SEQUENCE No_Cuenta
  2  START WITH 1
  3  INCREMENT BY 1;

Sequence created.
```

CREATE OR REPLACE PROCEDURE Crear_Ahorrosaprob(

p_codsucursal Ahorros.cod_sucursal%TYPE,

```
   p_idcliente Ahorros.id_cliente%TYPE,
   p_tipoahorro Ahorros.cod_tipoahorro%TYPE,
   p_interesa Ahorros.tasa_interesahorro%TYPE,
   p_letramensual Ahorros.letra_mensualahorro%TYPE,
   p_saldoahorro Ahorros.saldo_ahorro%TYPE,
   p_saldointeres Ahorros.saldo_interesahorro%TYPE,
   p_usuario Ahorros.id_usuario%TYPE,
   p_error out varchar2 ) AS
   BEGIN
   INSERT INTO Ahorros (
   no_cuenta_ahorro, cod_sucursal, id_cliente, cod_tipoahorro,
fecha_apertura, tasa_interesahorro, letra_mensualahorro,
saldo_ahorro, saldo_interesahorro, id_usuario,
fecha_modificacion_ahorro)
  VALUES ( No_Cuenta.NEXTVAL, p_codsucursal, p_idcliente,
p_tipoahorro, SYSDATE,p_interesa, p_letramensual, p_saldoahorro,
p_saldointeres,p_usuario, sysdate);
  UPDATE Suc_Ahorro
  SET monto_ahorrado = monto_ahorrado + p_saldoahorro
   Where Cod_tipoahorro = p_tipoahorro AND cod_sucursal =
p_codsucursal;
   IF SQL%ROWCOUNT=0 THEN
```

```
   INSERT INTO
   Suc_Ahorro (cod_tipoahorro, cod_sucursal, monto_ahorrado)
   VALUES ( p_tipoahorro, p_codsucursal, p_saldoahorro);
   END IF;
   p_error := 'Registro Creado Satisfactoriamente';
   EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
   p_error := 'Cuenta de Ahorro existente';
   WHEN OTHERS THEN
   p_error := 'No se creo el registro';
   COMMIT;
   END Crear_Ahorrosaprob;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE Crear_Ahorrosaprob(
  2     p_codsucursal Ahorros.cod_sucursal%TYPE,
  3     p_idcliente Ahorros.id_cliente%TYPE,
  4     p_tipoahorro Ahorros.cod_tipoahorro%TYPE,
  5     p_interesa Ahorros.tasa_interesahorro%TYPE,
  6     p_letramensual Ahorros.letra_mensualahorro%TYPE,
  7     p_saldoahorro Ahorros.saldo_ahorro%TYPE,
  8     p_saldointeres Ahorros.saldo_interesahorro%TYPE,
  9     p_usuario Ahorros.id_usuario%TYPE,
 10     p_error out varchar2 ) AS
 11     BEGIN
 12     INSERT INTO Ahorros (
 13     no_cuenta_ahorro, cod_sucursal, id_cliente, cod_tipoahorro,
fecha_apertura, tasa_interesahorro, letra_mensualahorro, saldo_ahorr
o, saldo_interesahorro, id_usuario, fecha_modificacion_ahorro)
 14    VALUES ( No_Cuenta.NEXTVAL, p_codsucursal, p_idcliente, p_tip
oahorro, SYSDATE,p_interesa, p_letramensual, p_saldoahorro, p_saldoi
nteres,p_usuario, sysdate);
 15    UPDATE Suc_Ahorro
 16    SET monto_ahorrado = monto_ahorrado + p_saldoahorro
 17     Where Cod_tipoahorro = p_tipoahorro AND cod_sucursal = p_cod
sucursal;
 18     IF SQL%ROWCOUNT=0 THEN
 19     INSERT INTO
 20     Suc_Ahorro (cod_tipoahorro, cod_sucursal, monto_ahorrado)
 21     VALUES ( p_tipoahorro, p_codsucursal, p_saldoahorro);
 22     END IF;
 23     p_error := 'Registro Creado Satisfactoriamente';
 24     EXCEPTION
 25     WHEN DUP_VAL_ON_INDEX THEN
 26     p_error := 'Cuenta de Ahorro existente';
 27     WHEN OTHERS THEN
 28     p_error := 'No se creo el registro';
 29     COMMIT;
 30     END Crear_Ahorrosaprob;
 31     /

Procedure created.
```

SET SERVEROUTPUT ON

Declare

p_error varchar2(55);

Begin

Crear_Ahorrosaprob('s1','cl1','ca1',0.01,40,null,null,'u1',p_error);

Crear_Ahorrosaprob('s2','cl2','ca2',0.09,20,null,null,'u2',p_error);

Crear_Ahorrosaprob('s3','cl3','ca3',0.12,10,null,null,'u2',p_error);

    dbms_output.put_line(p_error);

END;

/

```
SQL> SET SERVEROUTPUT ON
SQL> Declare
  2  p_error varchar2(55);
  3  Begin
  4  Crear_Ahorrosaprob('s1','cl1','ca1',0.01,40,null,null,'u1',p_er
ror);
  5  Crear_Ahorrosaprob('s2','cl2','ca2',0.09,20,null,null,'u2',p_er
ror);
  6  Crear_Ahorrosaprob('s3','cl3','ca3',0.12,10,null,null,'u2',p_er
ror);
  7
  8      dbms_output.put_line(p_error);
  9  END;
 10  /
Registro Creado Satisfactoriamente

PL/SQL procedure successfully completed.
```

Select* from Ahorros;

```
SQL> select* from Ahorros;

CO ID_CLIENTE COD_T NO_CUENTA_AHORRO FECHA_AP TASA_INTERESAHORRO
-- ---------- ----- ---------------- -------- ------------------
LETRA_MENSUALAHORRO SALDO_AHORRO SALDO_INTERESAHORRO FECHA_DE FECHA_
RE ID_US
------------------- ------------ ------------------- -------- ------
-- -----
FECHA_MO
--------
s1 cl1        ca1                  1 18/11/20                .01
                 40
   u1
18/11/20

s2 cl2        ca2                  2 18/11/20                .09
                 20
   u2
18/11/20

CO ID_CLIENTE COD_T NO_CUENTA_AHORRO FECHA_AP TASA_INTERESAHORRO
-- ---------- ----- ---------------- -------- ------------------
LETRA_MENSUALAHORRO SALDO_AHORRO SALDO_INTERESAHORRO FECHA_DE FECHA_
RE ID_US
------------------- ------------ ------------------- -------- ------
-- -----
FECHA_MO
--------

s3 cl3        ca3                  3 18/11/20                .12
                 10
   u2
18/11/20
```

Procedimiento almacenado para la carga o inserción de los depósitos o retiros recibidos de los clientes para sus cuentas de ahorros que se almacena en la tabla transadeporeti. Por lo menos uno para cada tipo de ahorro.

CREATE SEQUENCE id_transac

START WITH 1

INCREMENT BY 1;

```
SQL> CREATE SEQUENCE id_transac
  2  START WITH 1
  3  INCREMENT BY 1;

Sequence created.
```

```
CREATE OR REPLACE PROCEDURE crear_transadeporeti (
   p_cod_sucursal transadeporeti.cod_sucursal%TYPE,
   p_cod_tipotransac transadeporeti.cod_tipotransac%TYPE,
   p_id_cliente transadeporeti.id_cliente%TYPE,
   p_Cod_tipoahorro transadeporeti.Cod_tipoahorro%TYPE,
   p_montotransa transadeporeti.monto_transac%TYPE,
   p_id_usuario transadeporeti.id_usuario%TYPE,
   p_error out varchar2
   ) AS
  BEGIN
   Insert into transadeporeti (
   cod_sucursal, id_transacciondeporeti, id_cliente, Cod_tipoahorro,
cod_tipotransac, monto_transac, fecha_transaccion, fecha_insercion, id_usuario)
   Values (
   p_cod_sucursal, id_transac.NEXTVAL, p_id_cliente, p_Cod_tipoahorro,
p_cod_tipotransac, p_montotransa, sysdate, sysdate, p_id_usuario
);
   EXCEPTION WHEN DUP_VAL_ON_INDEX THEN
   p_error := 'TRANSACCION EXISTENTE';
```

```
  WHEN OTHERS THEN

  p_error := 'ERROR EN LA TRANSACCION';

  COMMIT;

  END crear_transadeporeti;

  /
```

```
SQL> CREATE OR REPLACE PROCEDURE crear_transadeporeti (
  2     p_cod_sucursal transadeporeti.cod_sucursal%TYPE,
  3     p_cod_tipotransac transadeporeti.cod_tipotransac%TYPE,
  4     p_id_cliente transadeporeti.id_cliente%TYPE,
  5     p_Cod_tipoahorro transadeporeti.Cod_tipoahorro%TYPE,
  6     p_montotransa transadeporeti.monto_transac%TYPE,
  7     p_id_usuario transadeporeti.id_usuario%TYPE,
  8     p_error out varchar2
  9     ) AS
 10    BEGIN
 11     Insert into transadeporeti (
 12     cod_sucursal, id_transacciondeporeti, id_cliente, Cod_tipoah
orro, cod_tipotransac, monto_transac, fecha_transaccion, fecha_inser
cion, id_usuario)
 13     Values (
 14     p_cod_sucursal, id_transac.NEXTVAL, p_id_cliente, p_Cod_tipo
ahorro, p_cod_tipotransac, p_montotransa, sysdate, sysdate, p_id_usu
ario
 15  );
 16     EXCEPTION WHEN DUP_VAL_ON_INDEX THEN
 17     p_error := 'TRANSACCION EXISTENTE';
 18     WHEN OTHERS THEN
 19     p_error := 'ERROR EN LA TRANSACCION';
 20     COMMIT;
 21     END crear_transadeporeti;
 22     /

Procedure created.
```

```
DECLARE

        p_error varchar2(35);

BEGIN

crear_transadeporeti('s1','cl1','ca1','tt1','40','u1',p_error);

   dbms_output.put_line(p_error);
```

END;

/

```
SQL> DECLARE
  2  p_error varchar2(35);
  3  BEGIN
  4  crear_transadeporeti('s1','cl1','ca1','tt1','40','u1',p_error);

  5      dbms_output.put_line(p_error);
  6  END;
  7  /
ERROR EN LA TRANSACCION

PL/SQL procedure successfully completed.
```

Procedimiento almacenado que actualice los depósitos o retiros de las cuentas de ahorro correspondientes. Deberá implementar un cursor que busque los depósitos/retiros insertados en la tabla uno a uno y los vaya actualizando en la tabla de ahorros de cada cliente (proceso en lote o proceso en línea usted decide).

De la siguiente forma:

- **Si el tipo de ahorro es navidad o escolar por cada deposito realizado debe calcular el interés que corresponde Monto_transaccion * tasade interes% que calculo que lo debe realizar una función diseñada previamente. El procedimiento debe actualizar el saldo de ahorro y el saldo interés de la cuenta de ahorro de los clientes**
- **Si el tipo de ahorro es corriente simplemente se realizar la aplicación del depósito o retiro a la cuenta de ahorro del cliente correspondiente. De las únicas cuentas que se puede realizar retiros es de la cuenta de ahorro corriente por lo tanto el procedimiento debe controlar esta situación.**

****

```
create or replace function intereses_cdeahorros (
p_monto_transa transadeporeti.monto_transac%TYPE,
p_interesahorro Ahorros.tasa_interesahorro%TYPE)
RETURN NUMBER AS
BEGIN
    return p_monto_transa*p_interesahorro;
END intereses_cdeahorros;
/
```

```
SQL> create or replace function intereses_cdeahorros (
  2  p_monto_transa transadeporeti.monto_transac%TYPE,
  3  p_interesahorro Ahorros.tasa_interesahorro%TYPE)
  4  RETURN NUMBER AS
  5  BEGIN
  6      return p_monto_transa*p_interesahorro;
  7  END intereses_cdeahorros;
  8  /

Function created.
```

****

```
CREATE OR REPLACE PROCEDURE actualizar_deporeti
(p_error out varchar2) AS
CURSOR actualizardeporeti IS
SELECT cod_sucursal, id_cliente, cod_tipotransac, Cod_tipoahorro,
monto_transac
From transadeporeti;
```

```
p_codsucursal transadeporeti.cod_sucursal%TYPE;
p_tipotransac transadeporeti.cod_tipotransac%TYPE;
p_idcliente transadeporeti.id_cliente%TYPE;
p_monto transadeporeti.monto_transac%TYPE;
p_tipoahorro tipo_ahorros.Cod_tipoahorro%TYPE;
p_interes tipo_ahorros.interes_ahorro%TYPE;
p_saldointeres Ahorros.saldo_interesahorro%TYPE;
p_saldoahorro Ahorros.saldo_ahorro%TYPE;
p_nocuenta Ahorros.no_cuenta_ahorro%TYPE;
BEGIN
OPEN actualizardeporeti;
LOOP
    FETCH actualizardeporeti INTO p_codsucursal, p_idcliente, p_tipotransac, p_tipoahorro, p_monto;
    Select Interes_ahorro INTO p_interes
    From tipo_ahorros WHERE (Cod_tipoahorro=p_tipoahorro);
    SELECT saldo_ahorro INTO p_saldoahorro
    FROM Ahorros
    WHERE id_cliente = p_idcliente AND cod_sucursal = p_codsucursal AND Cod_tipoahorro = p_tipoahorro;
```

```
    IF ((p_tipoahorro = 1 OR p_tipoahorro = 3) AND p_tipotransac
=1) THEN
    p_interes := intereses_cdeahorros (p_monto, p_interes);
    UPDATE Ahorros
    SET
    saldo_ahorro =saldo_ahorro + p_monto + p_interes,
    saldo_interesahorro = saldo_interesahorro+p_interes,
    fecha_modificacion_ahorro=sysdate,
    fecha_depositoahorro=sysdate
    WHERE p_idcliente=id_cliente AND
p_nocuenta=no_cuenta_ahorro;
    UPDATE Suc_Ahorro
    SET monto_ahorrado = monto_ahorrado + p_monto
    Where Cod_tipoahorro= p_tipoahorro AND
cod_sucursal=p_codsucursal;

   ELSIF ((p_tipoahorro = 1 OR p_tipoahorro = 3) AND
p_tipotransac=2) THEN
   DBMS_OUTPUT.PUT_LINE('RETIRO NO PERMITIDO');
    ELSIF (p_tipoahorro = 2 AND p_tipotransac=1) THEN
    UPDATE Ahorros
```

```
    SET
    saldo_ahorro = saldo_ahorro + p_monto,
    fecha_modificacion_ahorro=sysdate,
    fecha_depositoahorro=sysdate
    WHERE p_idcliente=id_cliente AND
p_nocuenta=no_cuenta_ahorro;
    UPDATE Suc_Ahorro
    SET monto_ahorrado = monto_ahorrado + p_monto
    Where Cod_tipoahorro=p_tipoahorro AND
cod_sucursal=p_codsucursal;
    ELSIF (p_tipoahorro = 2 AND p_tipotransac=2) THEN
            IF(p_monto <= p_saldoahorro) THEN
            UPDATE Ahorros
            SET
            saldo_ahorro = saldo_ahorro - p_monto,
            fecha_modificacion_ahorro=sysdate,
            fecha_retiroahorro = sysdate
            WHERE p_idcliente=id_cliente AND
p_nocuenta=no_cuenta_ahorro;
            UPDATE Suc_Ahorro
            SET monto_ahorrado = monto_ahorrado - p_monto
```

```
        Where Cod_tipoahorro=p_tipoahorro AND
cod_sucursal=p_codsucursal;
DBMS_OUTPUT.PUT_LINE ('RETIRO SATISFACTORIO');
        ELSE
        DBMS_OUTPUT.PUT_LINE ('NO CUENTA CON FONDOS
SUFICIENTES');
        END IF;
    END IF;
EXIT WHEN actualizardeporeti%NOTFOUND;
END LOOP;
CLOSE actualizardeporeti;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    p_error := 'DATOS NO ENCONTRADOS';
    WHEN OTHERS THEN
    p_error := 'CUENTA DE AHORROS NO ACTUALIZADA';
COMMIT;
END actualizar_deporeti;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE actualizar_deporeti
  2  (p_error out varchar2) AS
  3  CURSOR actualizardeporeti IS
  4  SELECT cod_sucursal, id_cliente, cod_tipotransac, Cod_tipoahorro, monto_tran
sac
  5  From transadeporeti;
  6  p_codsucursal transadeporeti.cod_sucursal%TYPE;
  7  p_tipotransac transadeporeti.cod_tipotransac%TYPE;
  8  p_idcliente transadeporeti.id_cliente%TYPE;
  9  p_monto transadeporeti.monto_transac%TYPE;
 10  p_tipoahorro tipo_ahorros.Cod_tipoahorro%TYPE;
 11  p_interes tipo_ahorros.interes_ahorro%TYPE;
 12  p_saldointeres Ahorros.saldo_interesahorro%TYPE;
 13  p_saldoahorro Ahorros.saldo_ahorro%TYPE;
 14  p_nocuenta Ahorros.no_cuenta_ahorro%TYPE;
 15  BEGIN
 16  OPEN actualizardeporeti;
 17  LOOP
 18  FETCH actualizardeporeti INTO p_codsucursal, p_idcliente, p_tipotransac, p_t
ipoahorro, p_monto;
 19  Select Interes_ahorro INTO p_interes
 20  From tipo_ahorros WHERE (Cod_tipoahorro=p_tipoahorro);
 21  SELECT saldo_ahorro INTO p_saldoahorro
 22     FROM Ahorros
 23  WHERE id_cliente = p_idcliente AND cod_sucursal = p_codsucursal AND Cod_tipo
ahorro = p_tipoahorro;
 24  IF ((p_tipoahorro = 1 OR p_tipoahorro = 3) AND p_tipotransac =1) THEN
 25  p_interes := intereses_cdeahorros (p_monto, p_interes);
 26  UPDATE Ahorros
 27  SET
 28  saldo_ahorro =saldo_ahorro + p_monto + p_interes,
 29  saldo_interesahorro = saldo_interesahorro+p_interes,
 30  fecha_modificacion_ahorro=sysdate,
 31  fecha_depositoahorro=sysdate
 32  WHERE p_idcliente=id_cliente AND p_nocuenta=no_cuenta_ahorro;
 33  UPDATE Suc_Ahorro
 34  SET monto_ahorrado = monto_ahorrado + p_monto
 35  Where Cod_tipoahorro= p_tipoahorro AND cod_sucursal=p_codsucursal;
```

```
36
37      ELSIF ((p_tipoahorro = 1 OR p_tipoahorro = 3) AND p_tipotransac=2) THEN

38      DBMS_OUTPUT.PUT_LINE('RETIRO NO PERMITIDO');
39  ELSIF (p_tipoahorro = 2 AND p_tipotransac=1) THEN
40  UPDATE Ahorros
41  SET
42  saldo_ahorro = saldo_ahorro + p_monto,
43  fecha_modificacion_ahorro=sysdate,
44  fecha_depositoahorro=sysdate
45  WHERE p_idcliente=id_cliente AND p_nocuenta=no_cuenta_ahorro;
46  UPDATE Suc_Ahorro
47  SET monto_ahorrado = monto_ahorrado + p_monto
48  Where Cod_tipoahorro=p_tipoahorro AND cod_sucursal=p_codsucursal;
49  ELSIF (p_tipoahorro = 2 AND p_tipotransac=2) THEN
50  IF(p_monto <= p_saldoahorro) THEN
51  UPDATE Ahorros
52  SET
53  saldo_ahorro = saldo_ahorro - p_monto,
54  fecha_modificacion_ahorro=sysdate,
55  fecha_retiroahorro = sysdate
56  WHERE p_idcliente=id_cliente AND p_nocuenta=no_cuenta_ahorro;
57  UPDATE Suc_Ahorro
58  SET monto_ahorrado = monto_ahorrado - p_monto
59  Where Cod_tipoahorro=p_tipoahorro AND cod_sucursal=p_codsucursal;
60  DBMS_OUTPUT.PUT_LINE ('RETIRO SATISFACTORIO');
61  ELSE
62  DBMS_OUTPUT.PUT_LINE ('NO CUENTA CON FONDOS SUFICIENTES');
63  END IF;
64  END IF;
65  EXIT WHEN actualizardeporeti%NOTFOUND;
66  END LOOP;
67  CLOSE actualizardeporeti;
68  EXCEPTION
69  WHEN NO_DATA_FOUND THEN
70  p_error := 'DATOS NO ENCONTRADOS';
71  WHEN OTHERS THEN
72  p_error := 'CUENTA DE AHORROS NO ACTUALIZADA';
73  COMMIT;
```

```
74  END actualizar_deporeti;
75  /

Procedure created.
```

Triggers. Para las actualizaciones en la tabla de sucursales

Create table tmp_sucu_ahorro (

id_temporal number not null,

cod_sucursal number,

tipodeahorro number,

montoahorrado number,

usuario varchar2(20),

fecha date,

constraint pk_temporalsucu primary key (id_temporal));

```
SQL> Create table tmp_sucu_ahorro (
  2     id_temporal number not null,
  3     cod_sucursal number,
  4     tipodeahorro number,
  5     montoahorrado number,
  6     usuario varchar2(20),
  7     fecha date,
  8     constraint pk_temporalsucu primary key (id_temporal));

Table created.
```

CREATE SEQUENCE temporal_sucahorro

START WITH 1

INCREMENT BY 1;

/

```
SQL> CREATE SEQUENCE temporal_sucahorro
  2  START WITH 1
  3  INCREMENT BY 1;

Sequence created.
```

Create or replace trigger sucursal_cuentadeahorro

BEFORE UPDATE ON Suc_Ahorro

FOR EACH ROW

BEGIN

INSERT INTO tmp_sucu_ahorro

values(temporal_sucahorro.nextval, cod_sucursal, tipodeahorro,montoahorrado,usuario,sysdate);

END sucursal_cuentadeahorro;

/

```
SQL> Create or replace trigger sucursal_cuentadeahorro
  2  BEFORE UPDATE ON Suc_Ahorro
  3  FOR EACH ROW
  4  BEGIN
  5  INSERT INTO tmp_sucu_ahorro
  6       values(temporal_sucahorro.nextval, cod_sucursal, tipodeahorro,montoahorr
ado,usuario,sysdate);
  7  END tmp_sucu_ahorro;
  8  /

Warning: Trigger created with compilation errors.
```

Procedimiento. Llegamos a fin de mes y hay pagar los intereses a la cuenta de ahorro corriente. Deberá diseñar un cursor que consulte todos los ahorros corrientes de forma controlada y calcule los intereses que habrá que pagar mensualmente a estos ahorros (saldo de ahorros por la tasade interes%) este cálculo lo realizar una función. El procedimiento deberá actualizar el saldo del ahorro y saldo de interés de las cuentas.' El interés es sumado al saldo de ahorro'.

CREATE OR REPLACE FUNCTION calcula_interescc(

p_saldoahorro Ahorros.saldo_ahorro%TYPE,

p_interes Ahorros.tasa_interesahorro%type)

RETURN NUMBER AS

BEGIN

RETURN (p_saldoahorro*p_interes)/100;

END calcula_interescc;

/

```
SQL> CREATE OR REPLACE FUNCTION calcula_interescc(
  2  p_saldoahorro Ahorros.saldo_ahorro%TYPE,
  3  p_interes Ahorros.tasa_interesahorro%type)
  4  RETURN NUMBER AS
  5  BEGIN
  6  RETURN (p_saldoahorro*p_interes)/100;
  7  END calcula_interescc;
  8  /

Function created.
```

CREATE OR REPLACE PROCEDURE pagodeinteres

(p_error out varchar2)

AS

CURSOR consultor_cc IS

SELECT id_cliente, no_cuenta_ahorro, saldo_ahorro, Cod_tipoahorro, cod_sucursal

FROM Ahorros WHERE cod_tipoahorro=2;

p_codsucursal Ahorros.cod_sucursal%TYPE;

p_idcliente Ahorros.id_cliente%TYPE;

p_tipoahorro Ahorros.cod_tipoahorro%TYPE;

```
p_nocuenta Ahorros.no_cuenta_ahorro%TYPE;
p_saldoahorro Ahorros.saldo_ahorro%TYPE;
p_interes Ahorros.tasa_interesahorro%TYPE;
p_saldointeres Ahorros.saldo_interesahorro%TYPE;
BEGIN
OPEN consultor_cc;
LOOP
    FETCH consultor_cc INTO
p_idcliente, p_nocuenta, p_saldoahorro, p_tipoahorro,
p_codsucursal;
    Select tasa_interesahorro INTO p_interes
    From Ahorros WHERE (cod_tipoahorro = p_tipoahorro);
    p_saldointeres:= calcula_interescc(p_saldoahorro,p_interes);
    UPDATE Ahorros
    SET
        saldo_ahorro=saldo_ahorro+ p_saldointeres,
        saldo_interesahorro=p_saldointeres,
        fecha_modificacion_ahorro=sysdate
   WHERE  id_cliente=p_idcliente AND no_cuenta_ahorro =
p_nocuenta;
    UPDATE Suc_Ahorro
```

```
    SET
        monto_ahorrado = monto_ahorrado+ p_saldointeres
    Where cod_sucursal=p_codsucursal AND Cod_tipoahorro=
p_tipoahorro;
EXIT WHEN consultor_cc%NOTFOUND;
END LOOP;
CLOSE consultor_cc;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    p_error := 'DATOS NO ENCONTRADOS';
    WHEN OTHERS THEN
    p_error := 'CUENTA DE AHORROS NO ACTUALIZADA';
COMMIT;
END pagodeinteres;
/
```

```
SQL> CREATE OR REPLACE PROCEDURE pagodeinteres
  2  (p_error out varchar2)
  3  AS
  4  CURSOR consultor_cc IS
  5  SELECT id_cliente, no_cuenta_ahorro, saldo_ahorro, Cod_tipoahorro, cod_sucur
sal
  6  FROM Ahorros WHERE cod_tipoahorro=2;
  7  p_codsucursal  Ahorros.cod_sucursal%TYPE;
  8  p_idcliente Ahorros.id_cliente%TYPE;
  9  p_tipoahorro Ahorros.cod_tipoahorro%TYPE;
 10  p_nocuenta Ahorros.no_cuenta_ahorro%TYPE;
 11  p_saldoahorro Ahorros.saldo_ahorro%TYPE;
 12  p_interes Ahorros.tasa_interesahorro%TYPE;
 13  p_saldointeres Ahorros.saldo_interesahorro%TYPE;
 14  BEGIN
 15  OPEN consultor_cc;
 16  LOOP
 17      FETCH consultor_cc INTO
 18  p_idcliente, p_nocuenta, p_saldoahorro, p_tipoahorro, p_codsucursal;
 19      Select tasa_interesahorro INTO p_interes
 20      From Ahorros WHERE (cod_tipoahorro = p_tipoahorro);
 21      p_saldointeres:= calcula_interescc(p_saldoahorro,p_interes);
 22      UPDATE Ahorros
 23      SET
 24          saldo_ahorro=saldo_ahorro+ p_saldointeres,
 25          saldo_interesahorro=p_saldointeres,
 26          fecha_modificacion_ahorro=sysdate
 27     WHERE  id_cliente=p_idcliente AND no_cuenta_ahorro = p_nocuenta;
 28      UPDATE Suc_Ahorro
 29      SET
 30          monto_ahorrado = monto_ahorrado+ p_saldointeres
 31      Where cod_sucursal=p_codsucursal AND Cod_tipoahorro= p_tipoahorro;
 32  EXIT WHEN consultor_cc%NOTFOUND;
 33  END LOOP;
 34  CLOSE consultor_cc;
 35  EXCEPTION
```

```
 36  WHEN NO_DATA_FOUND THEN
 37  p_error := 'DATOS NO ENCONTRADOS';
 38  WHEN OTHERS THEN
 39  p_error := 'CUENTA DE AHORROS NO ACTUALIZADA';
 40  COMMIT;
 41  END pagodeinteres;
 42  /

Procedure created.
```