**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) not null,

constraint cod\_sucursal\_fk2 foreign key (cod\_sucursal) References sucursales (cod\_sucursal),

monto\_ahorrado number not null,

constraint pk\_Suc\_Ahorro primary key (Cod\_tipoahorro, cod\_sucursal));

**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 not null );

Texto

Descripción generada automáticamente

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

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;

/

Texto

Descripción generada automáticamente

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

SET SERVEROUTPUT ON

Begin

crear\_Tipoahorros(‘ca1’,’ahorro de navidad’, 0.06);

crear\_Tipoahorros(‘ca2’,’ahorro corriente’, 0.04);

crear\_Tipoahorros(‘ca3’,’ahorro personal’, 0.08);

End;

/

Texto

Descripción generada automáticamente

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

Select \* from tipo\_ahorros;

Texto

Descripción generada automáticamente

**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) not null,

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 not null,

fecha\_apertura date not null,

tasa\_interesahorro number not null,

letra\_mensualahorro number not null,

saldo\_ahorro number not null,

saldo\_interesahorro number not null,

fecha\_depositoahorro date not null,

fecha\_retiroahorro date not null,

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)

);

Texto

Descripción generada automáticamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

Create table tipo\_transaccionbanc (

cod\_tipotransac varchar2(5) primary key not null,

tipo\_transaccion varchar2(20)

);

Texto

Descripción generada automáticamente

----- >

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;

/

Texto

Descripción generada automáticamente

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

SET SERVEROUTPUT ON

Begin

crear\_tipotransaccion(‘tt1’, ‘Abono’);

crear\_tipotransaccion(‘tt2’, ‘Retiro’);

End;

/

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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 not null,

id\_usuario varchar2(5),

constraint ide\_usuario2\_fk foreign key (id\_usuario) references usuarios (id\_usuario),

fecha\_transaccion date not null,

primary key (Cod\_tipoahorro, id\_cliente, id\_usuario));

Texto

Descripción generada automáticamente

**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 not null,

fecha\_insercion date not null,

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));

Texto

Descripción generada automáticamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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;

Texto

Descripción generada automáticamente

----

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, user, 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;

/

Texto

Descripción generada automáticamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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;

Texto

Descripción generada automáticamente

-----

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 := 'ESTA TRANSACCION YA EXISTE’;

WHEN OTHERS THEN

p\_error := 'NO SE CREO EL REGISTRON';

COMMIT;

END crear\_transadeporeti;

/

Texto

Descripción generada automáticamente

**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;

/

Texto

Descripción generada automáticamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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;

/

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

**Triggers. Para las actualizaciones en la tabla de sucursales**

Create table tmp\_sucu\_ahorro (

id\_temporal number not null,

cod\_sucursal number not null,

tipodeahorro number not null,

montoahorrado number not null,

usuario varchar2(20) not null,

fecha date not null,

constraint pk\_temporalsucu primary key (id\_temporal));

Texto

Descripción generada automáticamente

CREATE SEQUENCE temporal\_sucahorro

START WITH 1

INCREMENT BY 1;

/

Texto

Descripción generada automáticamente

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;

/

Texto

Descripción generada automáticamente

**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;

/

Texto

Descripción generada automáticamente

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;

/

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente