

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

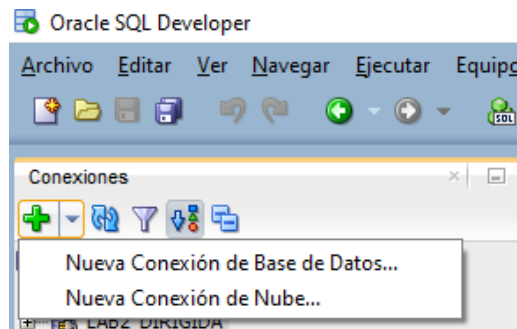
Bases de Datos 4ta. Práctica Dirigida

(Semestre Académico 2023-0)

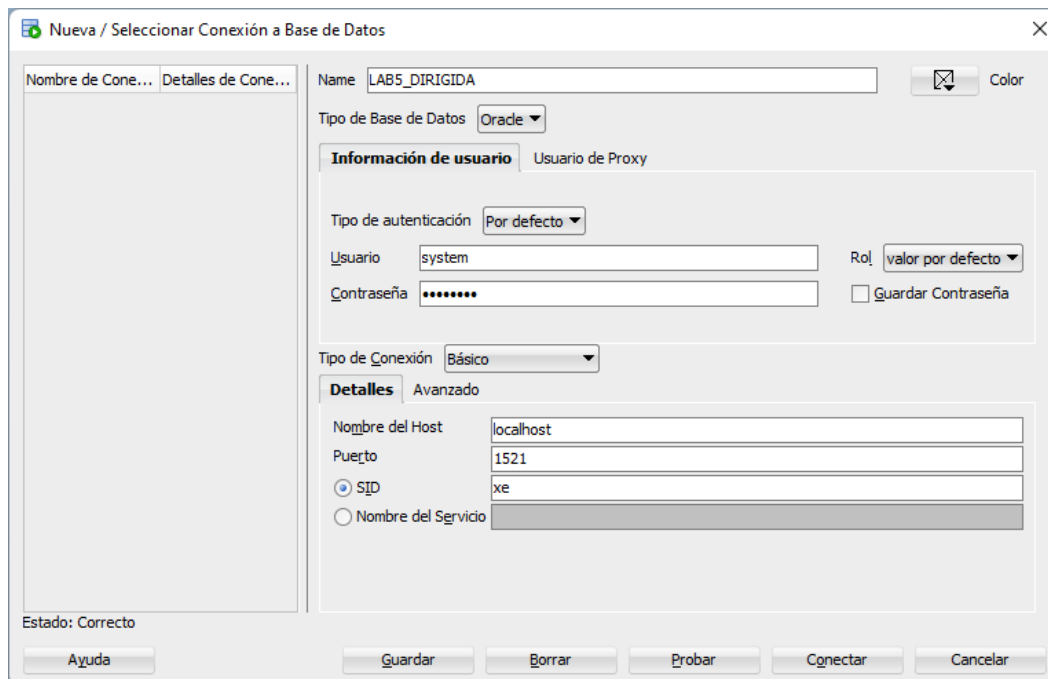
GUIA - Cursores, Triggers

Conectarse a una base de datos

Primero, ejecute **Oracle SQL Developer**, cierre la pestaña Página de bienvenida, y en el panel de **Conexiones**, haga clic en el ícono **+** para crear una nueva conexión.



Se abrirá la siguiente ventana:



Nueva / Seleccionar Conexión a Base de Datos

Nombre de Cone... Detalles de Cone...

Name: LAB5_DIRIGIDA

Tipo de Base de Datos: Oracle

Información de usuario

Tipo de autenticación: Por defecto

Usuario: system Rol: valor por defecto

Contraseña: ☐ Guardar Contraseña

Tipo de Conexión: Básico

Detalles Avanzado

Nombre del Host: localhost

Puerto: 1521

☒ SID: xe

☐ Nombre del Servicio

Estado: Correcto

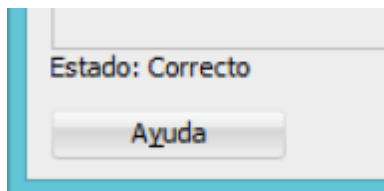
Ayuda Guardar Borrar Probar Conectar Cancelar

Ingresa los siguientes datos para la conexión:

- Nombre de conexión (Name): **LAB4_DIRIGIDA**
- Usuario: **system**
- Contraseña: Debe escribir la contraseña que ingresó al instalar.

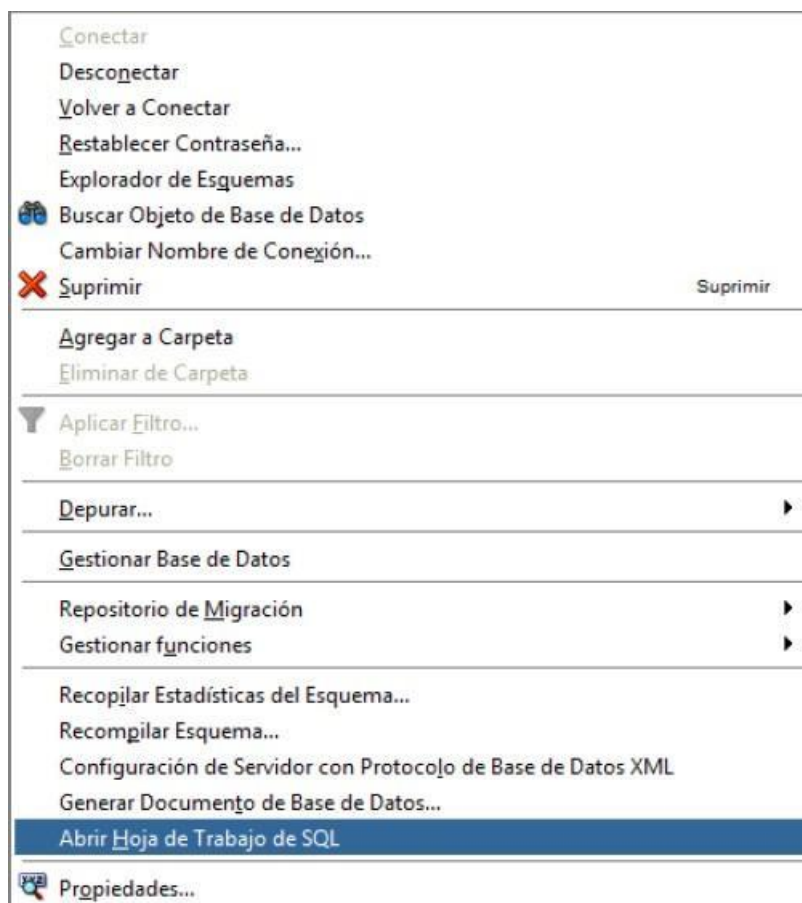
Puede dejar el resto de los parámetros tal y como aparecen por defecto.

Haga clic en el botón **Probar**. Si todo está bien configurado, aparecerá el mensaje “Estado: Correcto” en la parte inferior izquierda.

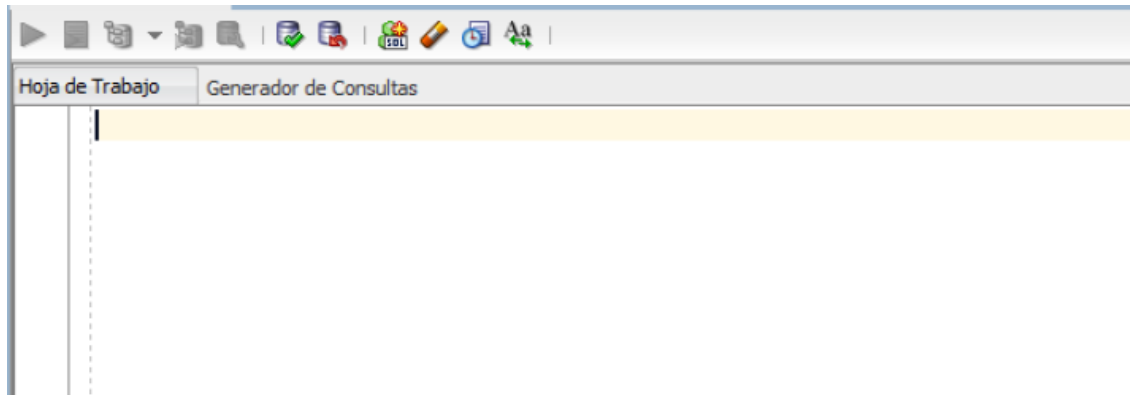


Luego, haga clic en el botón **Guardar**, y luego clic en el botón **Conectar**. El nombre de la conexión creada (LAB5_DIRIGIDA) aparecerá en el panel de Conexiones.

Haga clic derecho sobre el nombre de la conexión **LAB5_DIRIGIDA** y aparecerá el siguiente menú contextual:



Hacemos clic en **Abrir Hoja de Trabajo de SQL**. Nos saldrá una pantalla similar a la siguiente:



Esta es la hoja de trabajo, donde se podrán escribir y ejecutar los scripts SQL.

Para iniciar, ejecute los scripts que se adjuntan en la parte dirigida en el siguiente orden:

- INF246_2023-0_LAB4_DIR_Drops.sql
- INF246_2023-0_LAB4_DIR_DDL.sql
- INF246_2023-0_LAB4_DIR_DML.sql

Ver el diagrama relacional en el **anexo 1**.

Ejercicio 1

Elabore un **subprograma** SP_RECALCULAR_SALDO_CUENTA que en base a sus movimientos registrados en la tabla SB_DEPOSITO_RETIRO calcule y actualice el saldo de todas las cuentas. Para ello realizaremos lo siguiente:

- Recorreremos cada cuenta de la tabla SB_CUENTA usando un cursor.
- Por cada cuenta, de la tabla SB_DEPOSITO_RETIRO obtendremos el total depositado (cuando TIPO='D') y el total retirado (cuando TIPO='R').
- El saldo lo calcularemos de la diferencia entre el total depositado menos el total retirado.
- Solo actualizaremos el saldo de cuenta en caso el valor calculado sea diferente al valor actual indicado en el saldo de la cuenta, es decir, si ambos valores son iguales no ejecutaremos ningún UPDATE.
- Consideraremos que todos los movimientos se encuentran en la misma moneda de la cuenta.

```

CREATE OR REPLACE PROCEDURE SP_RECALCULAR_SALDO_CUENTA
AS
    nTotalDepositado NUMBER;
    nTotalRetirado NUMBER;
    nSaldo NUMBER;

    CURSOR CUR_CUENTAS IS
        SELECT ID_CUENTA, SALDO, MONEDA
        FROM SB_CUENTA;
BEGIN
    FOR RCUENTA IN CUR_CUENTAS
    LOOP
        nTotalDepositado:=0;
        nTotalRetirado:=0;
        SELECT NVL(SUM(MONTO),0) INTO nTotalDepositado
        FROM SB_DEPOSITO_RETIRO R
        WHERE R.ID_CUENTA = RCUENTA.ID_CUENTA AND TIPO='D';

        SELECT NVL(SUM(MONTO),0) INTO nTotalRetirado
        FROM SB_DEPOSITO_RETIRO R
        WHERE R.ID_CUENTA = RCUENTA.ID_CUENTA AND TIPO='R';

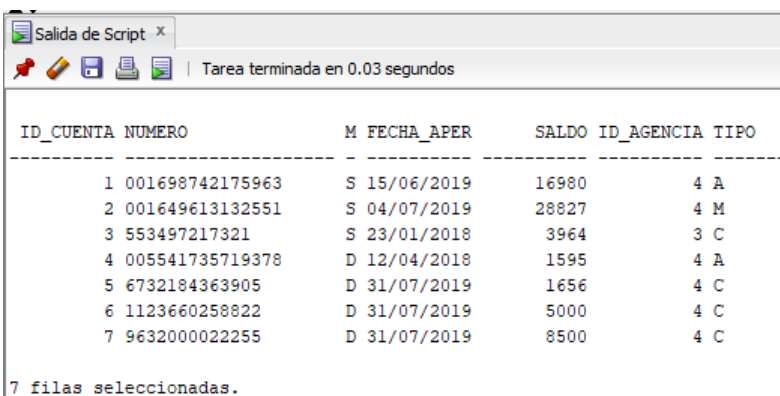
        nSaldo := nTotalDepositado - nTotalRetirado;

        dbms_output.put_line('TD:' || nTotalDepositado || ' TR:' || nTotalRetirado
        || ' SR:' || nSaldo || ' SC:' || RCUENTA.SALDO);
        IF (RCUENTA.SALDO != nSaldo) THEN
            UPDATE SB_CUENTA SET SALDO = nSaldo
            WHERE ID_CUENTA = RCUENTA.ID_CUENTA;
        END IF;

    END LOOP;
END;

```

```
SELECT * FROM SB_CUENTA
```



ID_CUENTA	NUMERO	M	FECHA_APER	SALDO	ID_AGENCIA	TIPO
1	001698742175963	S	15/06/2019	16980	4	A
2	001649613132551	S	04/07/2019	28827	4	M
3	553497217321	S	23/01/2018	3964	3	C
4	005541735719378	D	12/04/2018	1595	4	A
5	6732184363905	D	31/07/2019	1656	4	C
6	1123660258822	D	31/07/2019	5000	4	C
7	9632000022255	D	31/07/2019	8500	4	C

7 filas seleccionadas.

```
SET SERVEROUTPUT ON
EXEC SP_RECALCULAR_SALDO_CUENTA();
```

Tarea terminada en 0.053 segundos

```
TD:19800 TR:5220 SR:14580 SC:16980
TD:40000 TR:11473 SR:28527 SC:28827
TD:0 TR:0 SR:0 SC:3964
TD:0 TR:0 SR:0 SC:1595
TD:872 TR:216 SR:656 SC:1656
TD:0 TR:0 SR:0 SC:5000
TD:0 TR:0 SR:0 SC:8500

Procedimiento PL/SQL terminado correctamente.
```

```
SELECT * FROM SB_CUENTA
```

Tarea terminada en 0.027 segundos

ID_CUENTA	NUMERO	M	FECHA_APER	SALDO	ID_AGENCIA	TIPO
1	001698742175963	S	15/06/2019	14580	4	A
2	001649613132551	S	04/07/2019	28527	4	M
3	553497217321	S	23/01/2018	0	3	C
4	005541735719378	D	12/04/2018	0	4	A
5	6732184363905	D	31/07/2019	656	4	C
6	1123660258822	D	31/07/2019	0	4	C
7	9632000022255	D	31/07/2019	0	4	C

7 filas seleccionadas. |

Ejercicio 2

Elaborar un subprograma SP_GENERAR_CALENDARIO_PAGOS que genere el calendario de pagos de todos los préstamos, es decir, se deberá generar cada una de las cuotas mensuales correspondientes. Para ello realizaremos lo siguiente:

- Cada vez que se ejecute el subprograma, borraremos toda la data de la tabla SB_CUOTA.
- Recorreremos cada préstamo de la tabla SB_PRESTAMO usando un cursor.
- Consideraremos que la fecha de vencimiento para cada cuota es el último día del mes, comenzando con el mes en que se solicitó el préstamo.
- El monto que consideraremos en cada cuota se encuentra en la table SB_PRESTAMO.

```
CREATE OR REPLACE PROCEDURE SP_GENERAR_CALENDARIO_PAGOS
AS
    nContador NUMBER;
    nCuota NUMBER;
    nContadorCuota NUMBER;
    nVencimiento DATE;
    nMontoCuota NUMBER;
```

```

CURSOR CUR_PRESTAMO IS
    SELECT ID_PRESTAMO, MONTO, MONEDA, FECHA, PLAZO, INTERES, CUOTA
    FROM SB_PRESTAMO;
BEGIN

    DELETE FROM SB_CUOTA;
    nContador:=0;

    FOR RPRESTAMO IN CUR_PRESTAMO LOOP
        nCuota:=1;
        nContadorCuota:=0;
        nVencimiento:=LAST_DAY(RPRESTAMO.FECHA);

        WHILE nContadorCuota < RPRESTAMO.PLAZO LOOP
            INSERT INTO SB_CUOTA(ID_CUOTA, ID_PRESTAMO, NUM_CUOTA,
FECHA_VENCIMIENTO, MONTO_CUOTA, ESTADO)
                VALUES(nContador+1, RPRESTAMO.ID_PRESTAMO, nCuota, nVencimiento,
RPRESTAMO.CUOTA, 0);

            nContador:=nContador+1;
            nContadorCuota:=nContadorCuota+1;
            nCuota:=nCuota+1;
            nVencimiento:=ADD_MONTHS(nVencimiento,1);
        END LOOP;


        dbms_output.put_line('Préstamo ID:' || RPRESTAMO.ID_PRESTAMO || ' Cuotas
generadas:' || nContadorCuota);
    END LOOP;
END;

```

```

SET serveroutput ON
EXEC SP_GENERAR_CALENDARIO_PAGOS();

```

 | Tarea terminada en 0.04 segundos

```

Préstamo ID:1 Cuotas generadas:20
Préstamo ID:2 Cuotas generadas:25
Préstamo ID:3 Cuotas generadas:60

```

Procedimiento PL/SQL terminado correctamente.

Ejercicio 3

Elaborar un trigger que se ejecute cuando se inserte un registro en la tabla SB_DEPOSITO_RETIRO y actualice el saldo de la tabla SB_CUENTA.

```

CREATE OR REPLACE TRIGGER TR_ACTUALIZAR_SALDO_CUENTA
AFTER INSERT ON SB_DEPOSITO_RETIRO
FOR EACH ROW

```

```

DECLARE
    saldoAnterior NUMBER;
    saldoNuevo NUMBER;
BEGIN

    SELECT SALDO INTO saldoAnterior
    FROM SB_CUENTA
    WHERE ID_CUENTA = :NEW.ID_CUENTA;

    dbms_output.put_line('Saldo Anterior: ' || saldoAnterior);

    /*CUANDO ES UN DEPÓSITO*/
    IF :NEW.TIPO = 'D' THEN
        Update SB_CUENTA
        SET SALDO=SALDO+:NEW.MONTO
        WHERE ID_CUENTA = :NEW.ID_CUENTA;
        saldoNuevo := saldoAnterior+:NEW.MONTO;
    END IF;
    /*CUANDO ES UN RETIRO*/
    IF :NEW.TIPO = 'R' THEN
        Update SB_CUENTA
        SET SALDO=SALDO-:NEW.MONTO
        WHERE ID_CUENTA = :NEW.ID_CUENTA;
        saldoNuevo := saldoAnterior-:NEW.MONTO;
    END IF;


    dbms_output.put_line('Saldo Nuevo: ' || saldoNuevo);
END;

```

```

SET SERVEROUTPUT ON
insert into sb_deposito_retiro
(id_dep_ret, tipo, id_cuenta, monto, moneda, fecha_hora, id_agencia)
values
(31, 'D', 1, 500, 'S', to_date('30/06/2022 13:00', 'dd/mm/yyyy HH24:MI'), 1);

```

 Tarea terminada en 0.028 segundos

```

Saldo Anterior: 14580
Saldo Nuevo: 15080

```

```






1 fila insertadas.

```

```

SET SERVEROUTPUT ON
insert into sb_deposito_retiro
(id_dep_ret, tipo, id_cuenta, monto, moneda, fecha_hora, id_agencia)
values
(32, 'R', 1, 800, 'S', to_date('30/06/2022 13:30', 'dd/mm/yyyy HH24:MI'), 1);

```

     | Tarea terminada en 0.028 segundos

Saldo Anterior: 15080

Saldo Nuevo: 14280

1 fila insertadas.

Ejercicio 4

Actualizar el trigger anterior para que ahora solo se ejecute cuando se actualice la columna MONTO de la tabla SB_DEPOSITO_RETIRO. Este trigger debe actualizar el saldo de la tabla SB_CUENTA.

```
CREATE OR REPLACE TRIGGER TR_ACTUALIZAR_SALDO_CUENTA
AFTER UPDATE OF MONTO ON SB_DEPOSITO_RETIRO
FOR EACH ROW
DECLARE
    saldoAnterior NUMBER;
    saldoNuevo NUMBER;
BEGIN


    SELECT SALDO INTO saldoAnterior
    FROM SB_CUENTA
    WHERE ID_CUENTA = :NEW.ID_CUENTA;

    dbms_output.put_line('Saldo Anterior: ' || saldoAnterior);

    /*CUANDO ES UN DEPÓSITO*/
    IF :NEW.TIPO = 'D' THEN
        Update SB_CUENTA
        SET SALDO=SALDO+(:NEW.MONTO - :OLD.MONTO)
        WHERE ID_CUENTA = :NEW.ID_CUENTA;
        saldoNuevo := saldoAnterior+(:NEW.MONTO - :OLD.MONTO);
    END IF;
    /*CUANDO ES UN RETIRO*/
    IF :NEW.TIPO = 'R' THEN
        Update SB_CUENTA
        SET SALDO=SALDO-(:NEW.MONTO - :OLD.MONTO)
        WHERE ID_CUENTA = :NEW.ID_CUENTA;
        saldoNuevo := saldoAnterior-(:NEW.MONTO - :OLD.MONTO);
    END IF;






    dbms_output.put_line('Saldo Nuevo: ' || saldoNuevo);
END;
```

```
SET SERVEROUTPUT ON
UPDATE SB_DEPOSITO_RETIRO
SET TIPO = TIPO
WHERE ID_DEP_RET = 1
```

 | Tarea terminada en 0.026 segundos

1 fila actualizadas.

```
SET SERVEROUTPUT ON
UPDATE SB_DEPOSITO_RETIRO
  SET MONTO = MONTO*1.5
WHERE ID_DEP_RET = 1
```

     | Tarea terminada en 0.027 segundos

Saldo Anterior: 28527

Saldo Nuevo: 36027

1 fila actualizadas.

Ejercicio 5


Actualizar el trigger anterior para que ahora solo se ejecute cuando se elimine un registro de la tabla SB_DEPOSITO_RETIRO. Este trigger debe actualizar el saldo de la tabla SB_CUENTA.

```
CREATE OR REPLACE TRIGGER TR_ACTUALIZAR_SALDO_CUENTA
AFTER DELETE ON SB_DEPOSITO_RETIRO
FOR EACH ROW
DECLARE
    saldoAnterior NUMBER;
    saldoNuevo NUMBER;
BEGIN
    SELECT SALDO INTO saldoAnterior
    FROM SB_CUENTA
    WHERE ID_CUENTA = :OLD.ID_CUENTA;
    dbms_output.put_line('Saldo Anterior: ' || saldoAnterior);

    /*CUANDO ES UN DEPÓSITO*/
    IF :OLD.TIPO = 'D' THEN
        Update SB_CUENTA
        SET SALDO=SALDO - :OLD.MONTO
        WHERE ID_CUENTA = :OLD.ID_CUENTA;
        saldoNuevo := saldoAnterior - :OLD.MONTO;
    END IF;
    /*CUANDO ES UN RETIRO*/
    IF :OLD.TIPO = 'R' THEN
        Update SB_CUENTA
        SET SALDO=SALDO + :OLD.MONTO
        WHERE ID_CUENTA = :OLD.ID_CUENTA;
        saldoNuevo := saldoAnterior + :OLD.MONTO;
    END IF;

    dbms_output.put_line('Saldo Nuevo: ' || saldoNuevo);
END;
```

```
SET SERVEROUTPUT ON
DELETE FROM SB_DEPOSITO_RETIRO
WHERE ID_DEP_RET = 23
```

 Tarea terminada en 0.023 segundos

```
Saldo Anterior: 656
Saldo Nuevo: 706
```

```
1 fila eliminado
```

Ejercicio 6

Elaborar un trigger que no permita registrar un retiro si es que la cuenta no tiene saldo suficiente.

```
CREATE OR REPLACE TRIGGER TR_VALIDAR_SALDO_CUENTA
AFTER INSERT ON SB_DEPOSITO_RETIRO
FOR EACH ROW
DECLARE
    saldoAnterior NUMBER;
    saldoNuevo NUMBER;
BEGIN
    SELECT SALDO INTO saldoAnterior FROM SB_CUENTA
    WHERE ID_CUENTA = :NEW.ID_CUENTA;

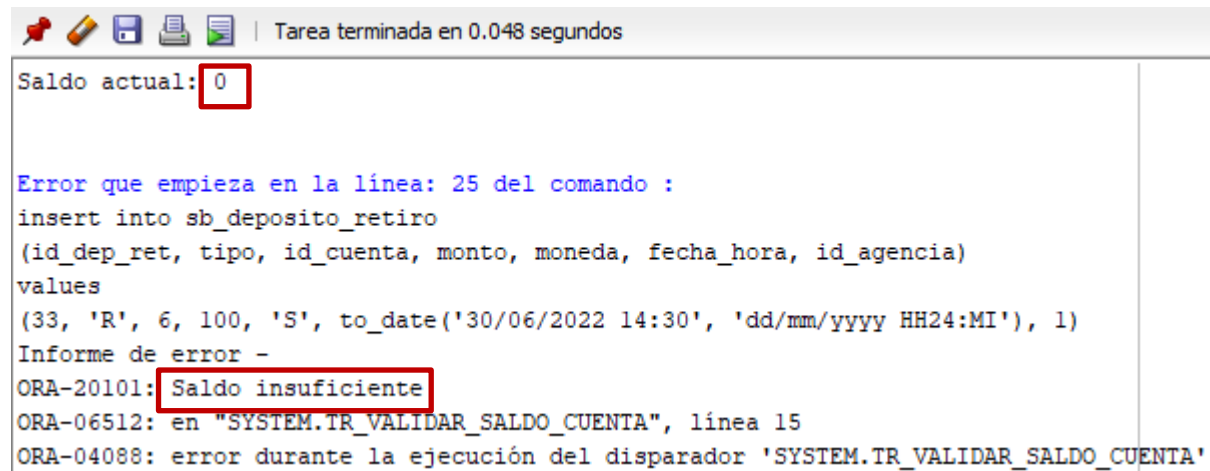
    dbms_output.put_line('Saldo actual: ' || saldoAnterior);

    /*CUANDO ES UN RETIRO*/
    IF :NEW.TIPO = 'R' THEN

        IF saldoAnterior < :NEW.MONTO THEN
            raise_application_error(-20101, 'Saldo insuficiente');
        END IF;

    END IF;
END;
```

```
SET SERVEROUTPUT ON
insert into sb_deposito_retiro
(id_dep_ret, tipo, id_cuenta, monto, moneda, fecha_hora, id_agencia)
values
(33, 'R', 6, 100, 'S', to_date('30/06/2022 14:30', 'dd/mm/yyyy HH24:MI'), 1);
```



Tarea terminada en 0.048 segundos

Saldo actual: 0

Error que empieza en la línea: 25 del comando :

```
insert into sb_deposito_retiro
(id_dep_ret, tipo, id_cuenta, monto, moneda, fecha_hora, id_agencia)
values
(33, 'R', 6, 100, 'S', to_date('30/06/2022 14:30', 'dd/mm/yyyy HH24:MI'), 1)
```

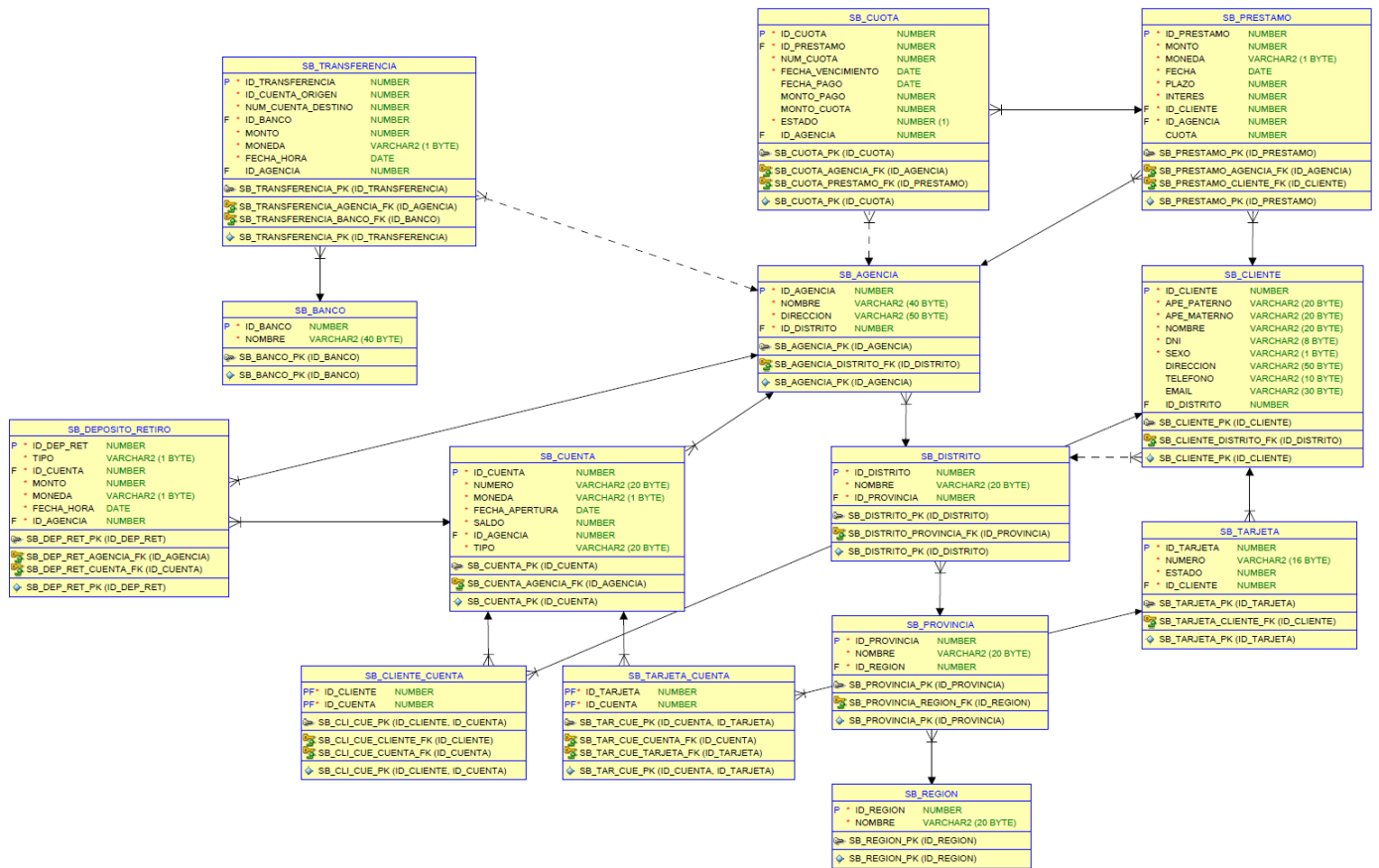
Informe de error -

ORA-20101: Saldo insuficiente

ORA-06512: en "SYSTEM.TR_VALIDAR_SALDO_CUENTA", línea 15

ORA-04088: error durante la ejecución del disparador 'SYSTEM.TR_VALIDAR_SALDO_CUENTA'

Anexo 1



Tabla(s)	Columna	Valores permitidos
SB_CLIENTE	SEXO	M = masculino F = femenino
SB_CUENTA SB_TRANSFERENCIA SB_DEPOSITO_RETIRO SB_PRESTAMO	MONEDA	S = soles D = dólares E = euros
SB_TARJETA	ESTADO	0 = inactiva 1 = activa
SB_DEPOSITO_RETIRO	TIPO	D = Depósito R = Retiro
SB_CUOTA	ESTADO	0 = No pagada 1 = Pagada a tiempo 2 = Pagada con mora

17 de febrero del 2022

Elaborado por DB