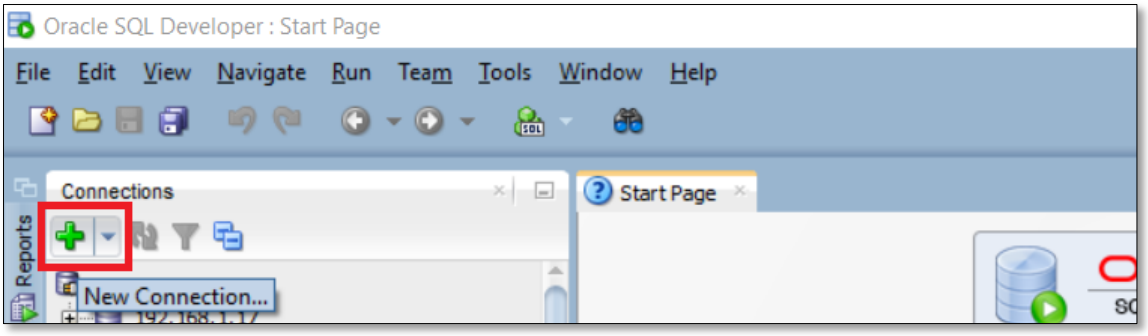
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Bases de Datos
5ta. Práctica Dirigida
(Primer Semestre 2020)

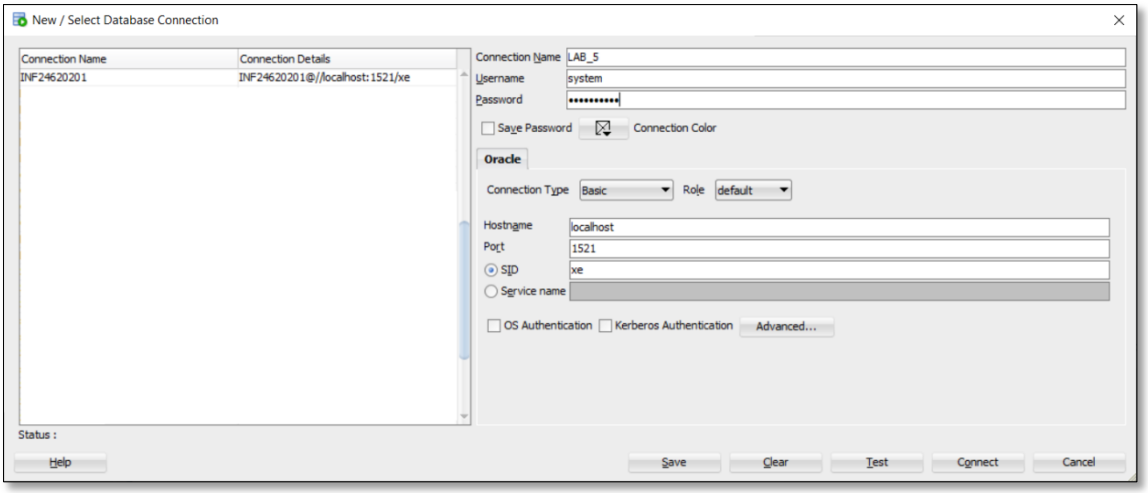
GUIA – CURSORES – TRIGGERS

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:

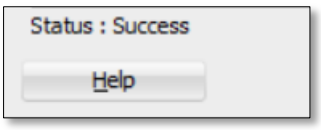


Ingrese los siguientes datos para la conexión:

- Nombre de conexión: **LAB5**
- Usuario: **system**
- Contraseña: Debe escribir la contraseña que ingresó al instalar Oracle Database Express Edition.

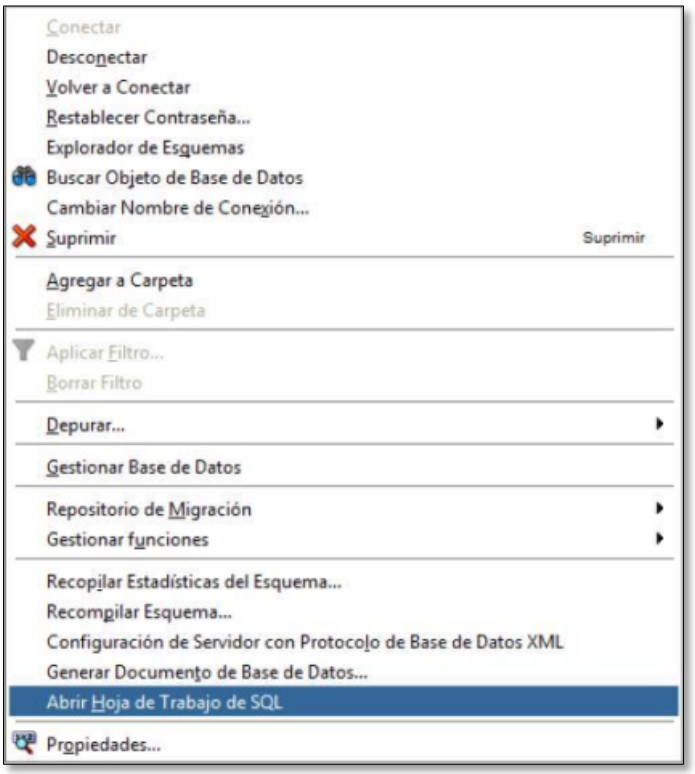
Puede dejar el resto de los parámetros tal y como aparecen por defecto.

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

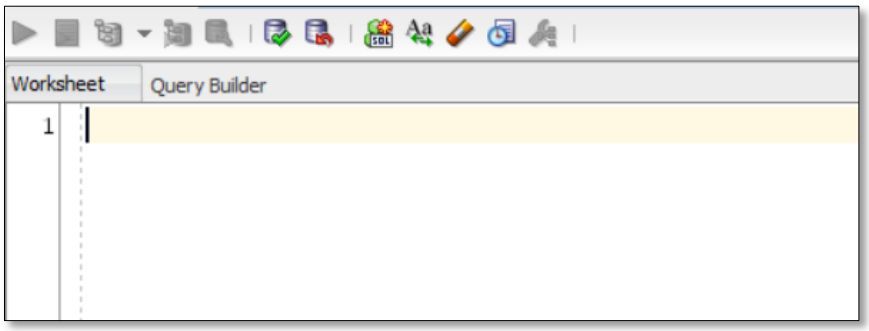


Luego, haga clic en el botón **Guardar**, y luego clic en el botón **Conectar**. El nombre de la conexión creada **(LAB_5)** aparecerá en el panel de conexiones.

Haga clic derecho sobre el nombre de la conexión **LAB_5** y aparecerá el siguiente menú contextual:



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



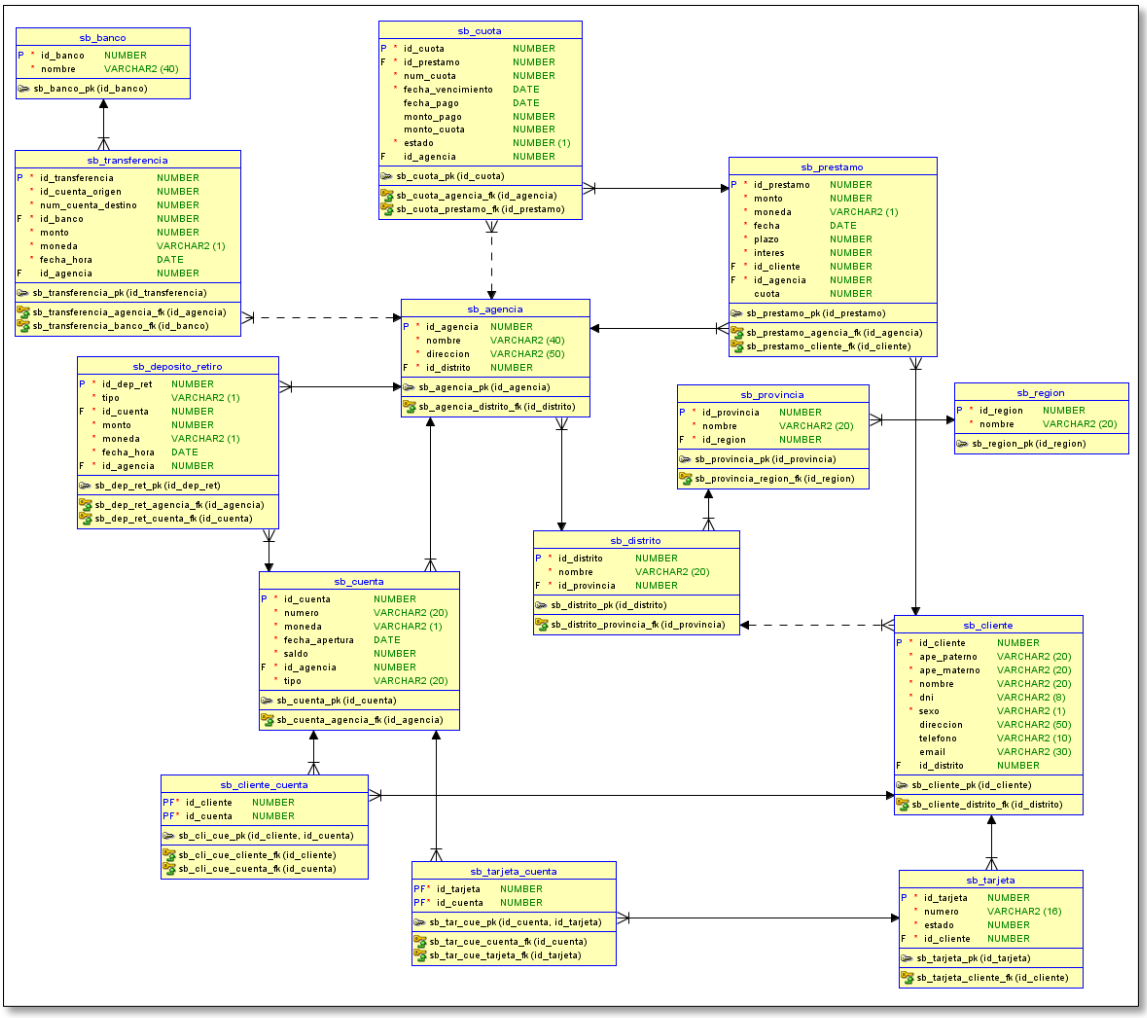
Esta es la **hoja de trabajo**, donde se podrán escribir y ejecutar los scripts SQL.

Sección Dirigida:

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

- INF246_2020-1_LAB5_01_dirigida_drops.sql
- INF246_2020-1_LAB5_02_dirigida_modelo.sql
- INF246_2020-1_LAB5_03_dirigida_inserts.sql

La estructura de tablas se muestra a continuación (también se adjunta el archivo **INF246_2020-1_LAB5_MODELO.png** del modelo):



1.- Se pide crear la siguiente tabla para llevar la trazabilidad de qué usuarios realizan inserciones sobre la tabla **SB_CLIENTE**, la tabla se llamará **SB_LOG**, ejecutar el siguiente script:

```
CREATE TABLE SB_LOG (
   ID_LOG NUMBER NOT NULL,
   NOMBRE_TABLA VARCHAR2(50) NOT NULL,
   USUARIO VARCHAR2(50) NOT NULL,
   FECHA_REGISTRO DATE NOT NULL
);

ALTER TABLE SB_LOG ADD CONSTRAINT SB_LOG_PK PRIMARY KEY ( ID_LOG );
```

La operación (trigger) debe registrar el nombre de la tabla, el usuario que realizó el registro y la fecha de operación. Para estos dos últimos datos puede usar las funciones **USER** y **SYSDATE** respectivamente.

Solución:

El trigger a crear tiene que dispararse cada vez que un usuario **inserte** datos en la tabla **SB_CLIENTE**, por lo tanto será un trigger **"after insert"**:

```
CREATE OR REPLACE TRIGGER TR_LOG_CLIENTE
AFTER INSERT ON SB_CLIENTE
FOR EACH ROW
DECLARE
  V_ID_LOG NUMBER;
BEGIN
  SELECT NVL(MAX(ID_LOG),0) INTO V_ID_LOG FROM SB_LOG;

  INSERT INTO SB_LOG (ID_LOG, NOMBRE_TABLA, USUARIO, FECHA_REGISTRO)
  VALUES (V_ID_LOG + 1, 'SB_CLIENTE', USER, SYSDATE);
END;
```

Realizamos la prueba del trigger de la siguiente manera:

- Insertamos un cliente en **SB_CLIENTE**.
- Comprobamos el registro insertado con un query en **SB_CLIENTE**.
- Comprobamos el funcionamiento del trigger con un query en **SB_LOG**.

Verificamos el último id para nuestro insert, en este ejemplo es 5:

```
SELECT * FROM SB_CLIENTE;
```

ID_CL...	APE_PATERNO	APE_MATERNO	NOMBRE	DNI	SEXO	DIRECCION
1	TORRES	SALAZAR	ROMINA ISABEL	40894715	F	Av. Paseo d
2	BENAVIDES	QUISPE	RAUL CARLOS	00478547	M	Calle Las P
3	CRUZ	RAMOS	MARIA DEL CARMEN	45330178	F	Av. Brasil
4	NESTOR	UBILLUZ	FELIPE	05930128	M	Calle B 112
5	NUEVO	PEREZ	LUIS	02365202	M	Calle B 112

Insertamos el registro para disparar el trigger:

```
INSERT INTO SB_CLIENTE (ID_CLIENTE, APE_PATERNO, APE_MATERNO, NOMBRE, DNI, SEXO,
DIRECCION, TELEFONO, EMAIL, ID_DISTRITO)
VALUES (6, 'THEO', 'MONZEN', 'OTTO', '41414141', 'M', 'AV UNION 123', '2528222',
'otto.theo@pucp.edu.pe', 1);
```

Verificamos que se insertó en la tabla SB_CLIENTE:

```
SELECT * FROM SB_CLIENTE;
```

ID_CLIENTE	APE_PATERNO	APE_MATERNO	NOMBRE	DNI	SEXO	DIRECCION
1	TORRES	SALAZAR	ROMINA ISABEL	40894715	F	Av. Paseo de la Rep
2	BENAVIDES	QUISPE	RAUL CARLOS	00478547	M	Calle Las Palmeras
3	CRUZ	RAMOS	MARIA DEL CARMEN	45330178	F	Av. Brasil 1678
4	NESTOR	UBILLUZ	FELIPE	05930128	M	Calle B 112
5	NUEVO	PEREZ	LUIS	02365202	M	Calle B 112
6	THEO	MONZEN	OTTO	41414141	M	AV UNION 123

Revisamos la tabla donde el trigger insertó el resultado de la acción:

```
SELECT * FROM SB_LOG;
```

ID_LOG	NOMBRE_TABLA	USUARIO	FECHA_REGISTRO
1	SB_CLIENTE	INF24620201	06-JUL-20

Y efectivamente se realizó la inserción. Con esto podemos llevar la trazabilidad de los usuarios que hicieron modificaciones en cualquier tabla del sistema, en este ejemplo de SB_CLIENTE.

2.- La tabla **SB_LOG** creada en el punto anterior puede ser utilizada para crear triggers sobre cualquiera de las tablas maestras del sistema, como SB_PRESTAMO, SB_TARJETA, entre otros, basta crear el trigger respectivo y cambiar el campo nombre_tabla por la tabla deseada. Sin embargo se desea tener un **control mayor** sobre la tabla **SB_CLIENTE** para poder tener la historia de los cambios realizados a los datos de los clientes (inserción, modificación y eliminación), solo se deben considerar los campos **id_distrito, dirección, teléfono** y **email** (los demás se consideran que no varían).

Se pide crear una tabla **SB_LOG_CLIENTE** que llevará el registro de los datos de los clientes nuevos, actualizados y eliminados en el sistema. Ejecutar el siguiente script:

```
CREATE TABLE SB_LOG_CLIENTE (
   ID_LOG_CLIENTE NUMBER NOT NULL,
   ID_CLIENTE NUMBER NOT NULL,
   DIREC_ANT VARCHAR2(50) NULL,
   TELEF_ANT VARCHAR2(10) NULL,
   EMAIL_ANT VARCHAR2(30) NULL,
   ID_DIS_ANT NUMBER NULL,
   DIREC_NUE VARCHAR2(50) NULL,
   TELEF_NUE VARCHAR2(10) NULL,
   EMAIL_NUE VARCHAR2(30) NULL,
   ID_DIS_NUE NUMBER NULL,
   USUARIO VARCHAR2(50) NOT NULL,
   FECHA_REGISTRO DATE NOT NULL,
   ACCION CHAR(1) NOT NULL
);

ALTER TABLE SB_LOG_CLIENTE ADD CONSTRAINT SB_LOG_CLIENTE_PK PRIMARY KEY (
ID_LOG_CLIENTE );
```

Con esto estamos creando una tabla que llevará los registros anteriores y nuevos.

- Si es **inserción** los campos **_ANT** estarán en NULL y los campos **_NUE** con los registros nuevos.
- Si es **eliminación** los campos **_ANT** tendrán los últimos registros antes de eliminar al cliente y los campos **_NUE** estarán en NULL.
- Si es **actualización** los campos **_ANT** guardarán la data anterior y los campos **_NUE** guardarán la data nueva.
- El campo **ACCION** tendrá los valores de **'I'** para insert, **'U'** para update y **'D'** para delete.

Solución:

El trigger a crear tiene que dispararse cada vez que un usuario **inserte**, **actualice** o **elimine** datos en la tabla **SB_CLIENTE**, por lo tanto será un trigger **"after insert or update or delete"**:

```
CREATE OR REPLACE TRIGGER TR_TRAZA_CLIENTE
AFTER INSERT OR UPDATE OR DELETE
ON SB_CLIENTE
FOR EACH ROW
DECLARE
 V_ID_LOG NUMBER;
BEGIN
 SELECT NVL(MAX(ID_LOG_CLIENTE),0) INTO V_ID_LOG FROM SB_LOG_CLIENTE;

 IF INSERTING THEN
  INSERT INTO SB_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, DIREC_NUE, TELEF_NUE,
EMAIL_NUE, ID_DIS_NUE, USUARIO, FECHA_REGISTRO, ACCION)
  VALUES (V_ID_LOG, :NEW.ID_CLIENTE, :NEW.DIRECCION, :NEW.TELEFONO, :NEW.EMAIL,
:NEW.ID_DISTRITO, USER, SYSDATE, 'I');
  -- USAMOS :NEW.ID_CLIENTE PORQUE ESTAMOS CREANDO UN NUEVO REGISTRO
 END IF;

 IF UPDATING THEN
  INSERT INTO SB_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, DIREC_ANT, TELEF_ANT,
EMAIL_ANT, ID_DIS_ANT,
   DIREC_NUE, TELEF_NUE, EMAIL_NUE, ID_DIS_NUE, USUARIO, FECHA_REGISTRO,
ACCION)
  VALUES (V_ID_LOG, :NEW.ID_CLIENTE, :OLD.DIRECCION, :OLD.TELEFONO, :OLD.EMAIL,
:OLD.ID_DISTRITO,
  :NEW.DIRECCION, :NEW.TELEFONO, :NEW.EMAIL, :NEW.ID_DISTRITO, USER, SYSDATE,
'U');
  -- PODEMOS USAR :OLD.ID_CLIENTE o :NEW.ID_CLIENTE, YA EL ID_CLIENTE NO CAMBIA,
ES UN PK
 END IF;

 IF DELETING THEN
  INSERT INTO SB_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, DIREC_ANT, TELEF_ANT,
EMAIL_ANT, ID_DIS_ANT, USUARIO, FECHA_REGISTRO, ACCION)
  VALUES (V_ID_LOG, :OLD.ID_CLIENTE, :OLD.DIRECCION, :OLD.TELEFONO, :OLD.EMAIL,
:OLD.ID_DISTRITO, USER, SYSDATE, 'D');
```

```
  -- USAMOS :OLD.ID_CLIENTE PORQUE ESTAMOS BORRANDO EL REGISTRO
 END IF;
END;
```

Realizamos la prueba del trigger de la siguiente manera:

- **Insertamos** un cliente en **SB_CLIENTE**.
- Comprobamos el registro insertado con un query en **SB_CLIENTE**.
- Comprobamos el funcionamiento del trigger con un query en **SB_LOG_CLIENTE**.
- **Actualizamos** un cliente en **SB_CLIENTE**.
- Comprobamos el registro actualizado con un query en **SB_CLIENTE**.
- Comprobamos el funcionamiento del trigger con un query en **SB_LOG_CLIENTE**.
- **Eliminamos** un cliente de **SB_CLIENTE**.
- Comprobamos el registro eliminado con un query en **SB_CLIENTE**.
- Comprobamos el funcionamiento del trigger con un query en **SB_LOG_CLIENTE**.

Verificamos el último id para nuestro insert, en este ejemplo es 6:

```
SELECT * FROM SB_CLIENTE;
```

ID_CLIENTE	APE_PATERNO	APE_MATERNO	NOMBRE	DNI	SEXO	DIR
1	TORRES	SALAZAR	ROMINA ISABEL	40894715	F	Av.
2	BENAVIDES	QUISPE	RAUL CARLOS	00478547	M	Call
3	CRUZ	RAMOS	MARIA DEL CARMEN	45330178	F	Av.
4	NESTOR	UBILLUZ	FELIPE	05930128	M	Call
5	NUEVO	PEREZ	LUIS	02365202	M	Call
6	THEO	MONZEN	OTTO	41414141	M	AV U

Insertamos el registro para disparar el trigger:

```
INSERT INTO SB_CLIENTE (ID_CLIENTE, APE_PATERNO, APE_MATERNO, NOMBRE, DNI, SEXO,
DIRECCION, TELEFONO, EMAIL, ID_DISTRITO)
VALUES (7, 'ARANA', 'CORDOVA', 'DANIEL', '42424242', 'M', 'AV JOSE OLAYA 123', '2528333',
'darana@pucp.pe', 1);
```

Verificamos que se insertó en la tabla SB_CLIENTE:

```
SELECT * FROM SB_CLIENTE;
```

ID_CLIENTE	APE_PATERNO	APE_MATERNO	NOMBRE	DNI	SEXO	DIRECC
1	TORRES	SALAZAR	ROMINA ISABEL	40894715	F	Av. Pa
2	BENAVIDES	QUISPE	RAUL CARLOS	00478547	M	Calle
3	CRUZ	RAMOS	MARIA DEL CARMEN	45330178	F	Av. Br
4	NESTOR	UBILLUZ	FELIPE	05930128	M	Calle
5	NUEVO	PEREZ	LUIS	02365202	M	Calle
6	THEO	MONZEN	OTTO	41414141	M	AV UNI
7	ARANA	CORDOVA	DANIEL	42424242	M	AV JOS

Revisamos la tabla donde el trigger insertó el resultado de la acción:

```
SELECT * FROM SB_LOG_CLIENTE;
```

ID_LO...	ID_...	DIREC_ANT	TELEF_ANT	EMAIL_ANT	ID_DIS_ANT	DIREC_NUE	TELEF_NUE	EMAIL_NUE	ID_DIS_NUE	USUARIO	FECHA_REGISTRO	ACCION
1	7	(null)	(null)	(null)	(null)	AV JOSE OL...	2528333	darana@pucp.pe	1	INF24620201	06-JUL-20	I

Y efectivamente se realizó la inserción. Los campos _ANT están en NULL pero los _NEW tienen los datos del registro.

Probemos una actualización de email al mismo cliente, de darana@pucp.pe a darana@pucp.edu.pe:

UPDATE SB_CLIENTE SET EMAIL = 'darana@pucp.edu.pe' WHERE ID_CLIENTE = 7;

Comprobamos los resultados del trigger:
SELECT * FROM SB_LOG_CLIENTE;

ID_LO...	ID_...	DIREC_ANT	TELEF_ANT	EMAIL_ANT	ID_DIS_ANT	DIREC_NUE	TELEF_NUE	EMAIL_NUE	ID_DIS_NUE	USUARIO	FECHA_REGISTRO	ACCION
1	7	(null)	(null)	(null)	(null)	AV JOSE OL...	2528333	darana@pucp.pe	1	INF24620201	06-JUL-20	I
2	7	AV JOS...	2528333	darana...	1	AV JOSE OL...	2528333	darana@pucp...	1	INF24620201	06-JUL-20	U

Actualización correcta.

Ahora probemos eliminar el cliente:

DELETE FROM SB_CLIENTE WHERE ID_CLIENTE = 7;

Comprobamos los resultados del trigger:
SELECT * FROM SB_LOG_CLIENTE;

ID_LO...	ID_...	DIREC_ANT	TELEF_ANT	EMAIL_ANT	ID_DIS_ANT	DIREC_NUE	TELEF_NUE	EMAIL_NUE	ID_DIS_NUE	USUARIO	FECHA_REGISTRO	ACCION
1	7	(null)	(null)	(null)	(null)	AV JOSE OL...	2528333	darana@pucp.pe	1	INF24620201	06-JUL-20	I
2	7	AV JOS...	2528333	darana...	1	AV JOSE OL...	2528333	darana@pucp...	1	INF24620201	06-JUL-20	U
3	7	AV JOS...	2528333	darana...	1	(null)	(null)	(null)	(null)	INF24620201	06-JUL-20	D

Con esto podemos guardar la historia de los cambios que tienen los registros de la tabla SB_CLIENTE.

A manera de observación, si revisamos la tabla **SB_LOG** del **punto 1.-** vemos que también se registraron los cambios necesarios:

SELECT ID_LOG, NOMBRE_TABLA, USUARIO, TO_CHAR(FECHA_REGISTRO, 'DD/MM/YYYY HH24:MI:SS') FROM SB_LOG;

ID_LOG	NOMBRE_TABLA	USUARIO	TO_CHAR(FECHA_REGISTRO,'DI
1	SB_CLIENTE	INF24620201	06/07/2020 15:34:09
2	SB_CLIENTE	INF24620201	06/07/2020 16:30:07
3	SB_CLIENTE	INF24620201	06/07/2020 16:40:55
4	SB_CLIENTE	INF24620201	06/07/2020 16:41:00
5	SB_CLIENTE	INF24620201	06/07/2020 16:41:31
6	SB_CLIENTE	INF24620201	06/07/2020 16:41:42
7	SB_CLIENTE	INF24620201	06/07/2020 16:46:09
8	SB_CLIENTE	INF24620201	06/07/2020 16:47:36

Podemos observar todos los cambios realizados a la tabla cliente (en este caso por el usuario INF24620201) y la fecha y hora en que fueron realizados. Pero la información detallada la tenemos en **SB_LOG_CLIENTE**.

3.- Se pide crear un procedimiento almacenado que imprima un reporte con todos los préstamos registrados en el sistema y sus respectivas cuotas en un formato como el siguiente:

CÓD. PRÉSTAMO: 1 / FECHA PRÉSTAMO: 15-03-2020 / MONTO TOTAL: 5000
NRO. CUOTAS: 10 / VALOR CUOTA: 500 / MONEDA: SOLES / INTERÉS: 6%
CLIENTE: 40894715 - TORRES SALAZAR, ROMINA ISABEL
AGENCIA: Agencia Open Plaza
NRO. CUOTA: 1
FECHA VENCIMIENTO: 15-04-2020
FECHA PAGO: 15-04-2020
MONTO DE LA CUOTA: 500
MONTO PAGADO: 500
ESTADO: Pagado a tiempo

NRO. CUOTA: 2
FECHA VENCIMIENTO: 15-05-2020
FECHA PAGO: 12-05-2020
MONTO DE LA CUOTA: 500
MONTO PAGADO: 500
ESTADO: Pagado a tiempo

CÓD. PRÉSTAMO: 2 / FECHA PRÉSTAMO: 12-04-2020 / MONTO TOTAL: 50000
NRO. CUOTAS: 5 / VALOR CUOTA: 10000 / MONEDA: SOLES / INTERÉS: 9%
CLIENTE: 05930128 - NESTOR UBILLUZ, FELIPE
AGENCIA: Agencia Las Musas
NRO. CUOTA: 1
FECHA VENCIMIENTO: 12-05-2020
FECHA PAGO: 12-05-2020
MONTO DE LA CUOTA: 10000
MONTO PAGADO: 10000
ESTADO: Pagado a tiempo

NRO. CUOTA: 2
FECHA VENCIMIENTO: 12-06-2020
FECHA PAGO: 18-06-2020
MONTO DE LA CUOTA: 10000
MONTO PAGADO: 10020
ESTADO: Pagado con mora

NRO. CUOTA: 3
FECHA VENCIMIENTO: 12-07-2020
FECHA PAGO: No cancelado
MONTO DE LA CUOTA: 10000
MONTO PAGADO: 10000
ESTADO: No pagado

Consideraciones:

- La moneda S, D o E debe ser escrita en el reporte como SOLES, DÓLARES o EUROS.

- El estado de la cuota 0, 1 y 2 debe ser escrita en el reporte como No pagado, Pagado a tiempo y Pagado con mora.
- Si el monto pagado es NULL colocar 0.
- El formato de fechas debe ser DD-MM-YYYY

Solución:

Para este problema debemos recorrer todos los elementos de la tabla SB_PRESTAMO, y para cada uno recorrer todos los elementos de la tabla SB_CUOTA, esta solución se va a realizar con dos cursores uno para que liste los préstamos y otro cursor anidado que reciba como parámetro el ID del préstamo y listar las cuotas de cada préstamo.

En esta solución se mostrará como recorrer los elementos del cursor de préstamos con variables independientes, lo cual es algo complicado si se manejan muchas variables. Y para el segundo cursos de cuotas se empleará una variable del tipo de dato de la fila del cursor.

```
CREATE OR REPLACE PROCEDURE PR_REPORTE_PRESTAMO IS
 CURSOR C_PRESTAMO IS
  SELECT P.ID_PRESTAMO, P.MONTO, P.MONEDA, P.FECHA, P.PLAZO,
  P.INTERES, P.ID_CLIENTE, P.ID_AGENCIA, P.CUOTA,
  A.NOMBRE AS AGENCIA, C.DNI || ' - ' || C.APE_PATERNO || ' ' || C.APE_MATERNO || ', '
|| C.NOMBRE AS CLIENTE
  FROM SB_PRESTAMO P, SB_CLIENTE C, SB_AGENCIA A
  WHERE P.ID_CLIENTE = C.ID_CLIENTE
  AND P.ID_AGENCIA = A.ID_AGENCIA;

 CURSOR C_CUOTA(P_ID_PRESTAMO NUMBER) IS
  SELECT C.ID_CUOTA, C.ID_PRESTAMO, C.NUM_CUOTA, C.FECHA_VENCIMIENTO,
C.FECHA_PAGO,
  C.MONTO_PAGO, C.MONTO_CUOTA, C.ESTADO
  FROM SB_CUOTA C
  WHERE C.ID_PRESTAMO = P_ID_PRESTAMO
  ORDER BY C.NUM_CUOTA;

 -- PODEMOS USAR CAMPOS POR SEPARADO PARA GUARDAR LOS RESULTADOS DEL
QUERY:
 V_ID_PRESTAMO NUMBER; V_MONTO NUMBER; V_MONEDA VARCHAR2(1); V_FECHA
DATE;
 V_PLAZO NUMBER; V_INTERES NUMBER; V_ID_CLIENTE NUMBER; V_ID_AGENCIA
NUMBER;
 V_CUOTA NUMBER; V_AGENCIA VARCHAR2(40); V_CLIENTE VARCHAR2(100);
 -- O PODEMOS USAR UN REGISTRO DEL MISMO TIPO DE DATO QUE LA FILA DEL CURSOR
 -- ESTO SI TENEMOS MUCHOS CAMPOS
 REG_CUOTA C_CUOTA%ROWTYPE;
 -- VARIABLE PARA GUARDAR LA MONEDA
 V_NOMBRE_MON VARCHAR2(10);
 -- VARIABLE PARA GUARDAR EL ESTADO DE LA CUOTA
 V_ESTADO VARCHAR2(20);
BEGIN
 OPEN C_PRESTAMO;
 LOOP
```

```
   -- EN ESTE EJEMPLO HACEMOS FETCH DE CADA CAMPO POR SEPARADO
   FETCH C_PRESTAMO INTO V_ID_PRESTAMO, V_MONTO, V_MONEDA, V_FECHA,
V_PLAZO,
   V_INTERES, V_ID_CLIENTE, V_ID_AGENCIA, V_CUOTA, V_AGENCIA, V_CLIENTE;
   EXIT WHEN C_PRESTAMO%NOTFOUND;
   -- NOMBRE DE MONEDA
   IF V_MONEDA = 'S' THEN V_NOMBRE_MON := 'SOLES'; END IF;
   IF V_MONEDA = 'D' THEN V_NOMBRE_MON := 'DÓLARES'; END IF;
   IF V_MONEDA = 'E' THEN V_NOMBRE_MON := 'EUROS'; END IF;
   -- IMPRIMIMOS LA CABECERA DEL REPORTE
   DBMS_OUTPUT.PUT_LINE('CÓD. PRÉSTAMO: ' || V_ID_PRESTAMO || ' / FECHA
PRÉSTAMO: ' || TO_CHAR(V_FECHA, 'DD-MM-YYYY') || ' / MONTO TOTAL: ' || V_MONTO);
   DBMS_OUTPUT.PUT_LINE('NRO. CUOTAS: ' || V_PLAZO || ' / VALOR CUOTA: ' ||
V_CUOTA || ' / MONEDA: ' || V_NOMBRE_MON || ' / INTERÉS: ' || V_INTERES || '%');
   DBMS_OUTPUT.PUT_LINE('CLIENTE: ' || V_CLIENTE);
   DBMS_OUTPUT.PUT_LINE('AGENCIA: ' || V_AGENCIA);

   -- AHORA ITERAMOS PARA CADA PRESTAMOS CON EL ID_PRESTAMO

   OPEN C_CUOTA(V_ID_PRESTAMO);
   LOOP
    -- EN ESTE EJEMPLO HACEMOS FETCH EN EL REGISTRO REG_CUOTA
    FETCH C_CUOTA INTO REG_CUOTA;
    EXIT WHEN C_CUOTA%NOTFOUND;
    --ESTADO
    IF REG_CUOTA.ESTADO = 0 THEN V_ESTADO := 'No pagado'; END IF;
    IF REG_CUOTA.ESTADO = 1 THEN V_ESTADO := 'Pagado a tiempo'; END IF;
    IF REG_CUOTA.ESTADO = 2 THEN V_ESTADO := 'Pagado con mora'; END IF;
    DBMS_OUTPUT.PUT_LINE('    NRO. CUOTA: ' || REG_CUOTA.NUM_CUOTA);
    DBMS_OUTPUT.PUT_LINE('    FECHA VENCIMIENTO: ' ||
TO_CHAR(REG_CUOTA.FECHA_VENCIMIENTO, 'DD-MM-YYYY'));
    DBMS_OUTPUT.PUT_LINE('    FECHA PAGO: ' ||
NVL(TO_CHAR(REG_CUOTA.FECHA_PAGO, 'DD-MM-YYYY'), 'No cancelado'));
    DBMS_OUTPUT.PUT_LINE('    MONTO DE LA CUOTA: ' || REG_CUOTA.MONTO_CUOTA);
    DBMS_OUTPUT.PUT_LINE('    MONTO PAGADO: ' || NVL(REG_CUOTA.MONTO_PAGO,
0));
    DBMS_OUTPUT.PUT_LINE('    ESTADO: ' || V_ESTADO);
    DBMS_OUTPUT.PUT_LINE(''); -- LINEA EN BLANCO
   END LOOP;
   CLOSE C_CUOTA;

  END LOOP;
  CLOSE C_PRESTAMO;
END;
```

Probamos el procedure del reporte:

```
SET SERVEROUTPUT ON;
EXEC PR_REPORTE_PRESTAMO;
```

Se coloca la instrucción **SET SERVEROUTPUT ON**; antes de llamar al procedure ya que este usa la función para imprimir llamada **DBMS_OUTPUT.PUT_LINE**.

09 de julio de 2020
OTM