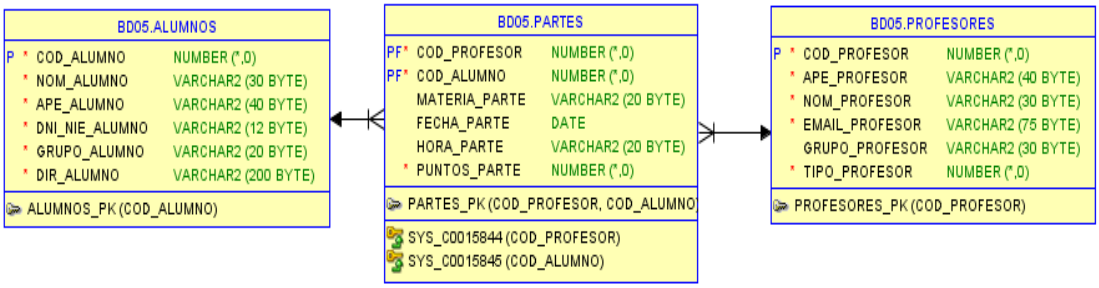
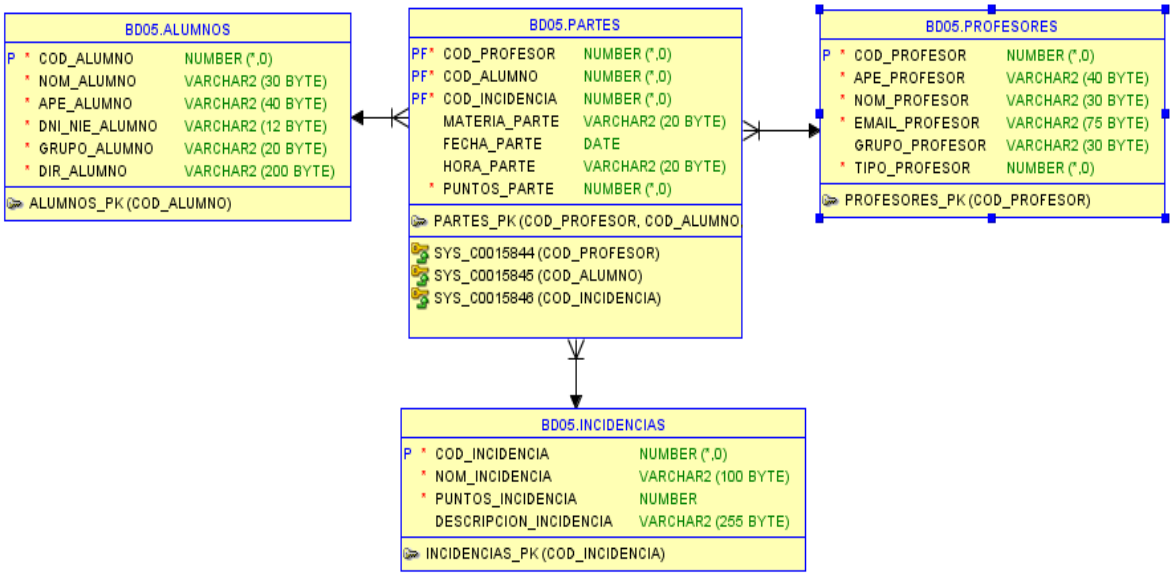
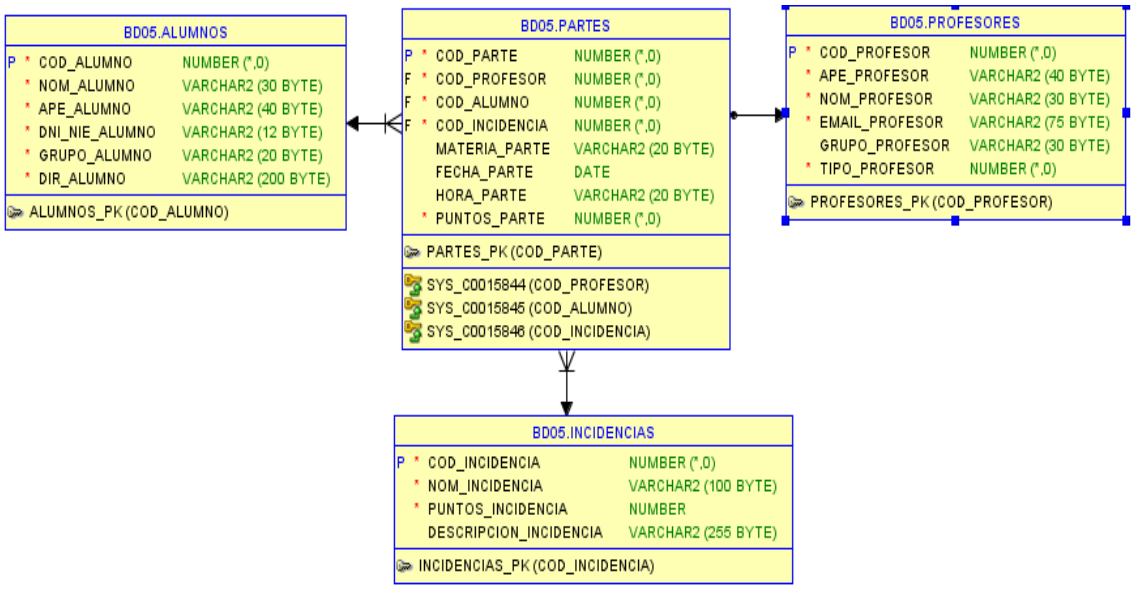
1. Crear todas tablas necesarias para reflejar la siguiente relación. *Las restricciones de claves principal y clave ajena las realizaremos a nivel de tabla.*



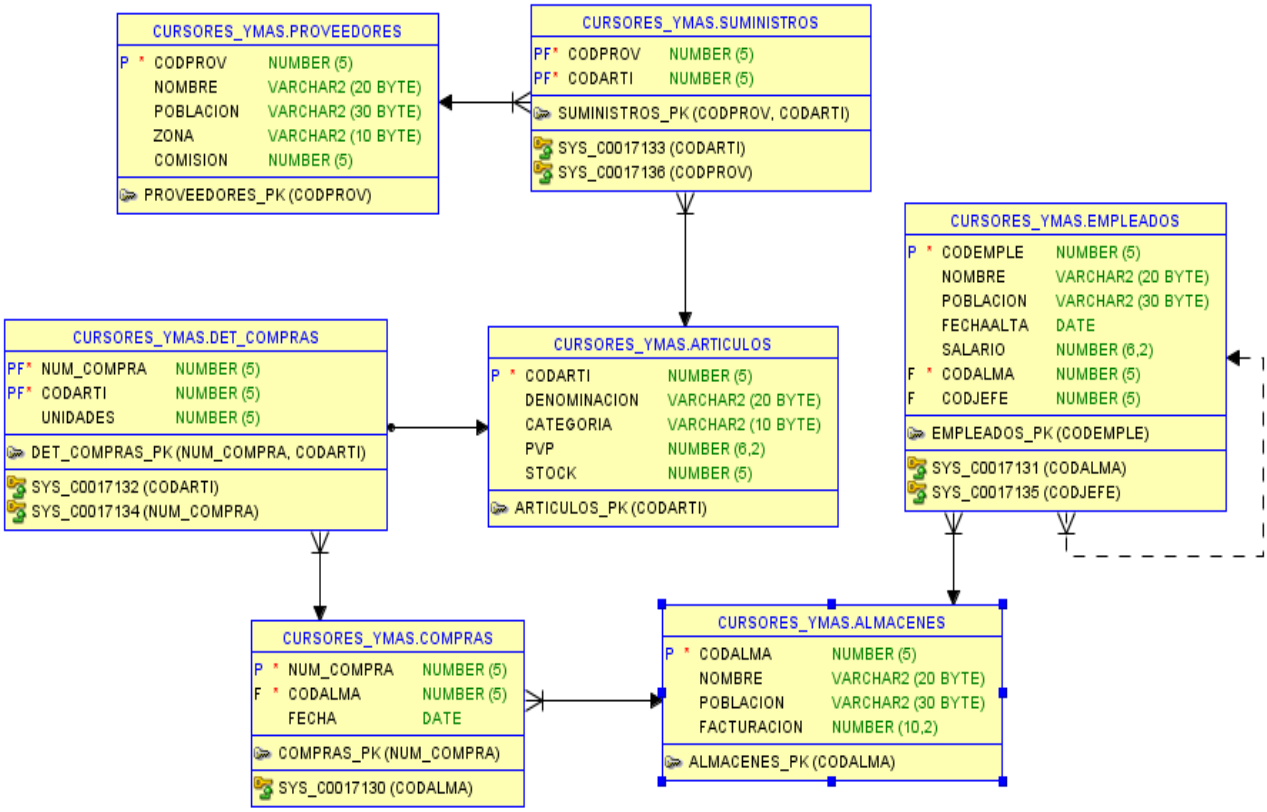
2. Uy!!!! Realmente la relación anterior era de la siguiente forma. Realiza **todas** las modificaciones necesarias, para reflejar los cambios realizados en el modelo. *Las restricciones de claves principal y clave ajena las realizaremos a nivel de tabla.*



3. Uuuuuyyyyy!!! Ahora tenemos que añadir un nuevo campo a la tabla de relación, que sea clave, es decir, la clave cambiará y pasará a estar formada sólo por el nuevo campo. Los campos que teníamos anteriormente en la tabla, seguirán existiendo en la tabla, pero tendrás que realizar los cambios necesarios como consecuencia de la inserción del nuevo campo clave. *Las restricciones de claves principal y clave ajena las realizaremos a nivel de tabla.*



UTILIZA LA BASES DE DATOS BDALMACEN



1. Crear una vista que nos permite obtener todos los datos del artículo(s) que más unidades se hayan comprado.

```
SELECT * FROM ARTICULOS
WHERE CODARTI IN (SELECT CODARTI FROM DET_COMPRAS
                  GROUP BY CODARTI
                  HAVING SUM(UNIDADES) = (SELECT MAX(SUM(UNIDADES))
                                          FROM DET_COMPRAS
                                          GROUP BY CODARTI));
```

2. Crear una vista con los datos de los empleados que trabajen en los almacenes del artículo del que se hayan realizado más compras. (NO DEL ARTÍCULO DEL QUE SE HAYAN COMPRADO MÁS UNIDADES).

```
SELECT DISTINCT (E.CODEMPLE), E.NOMBRE, E.POBLACION, E.FECHAALTA, E.SALARIO
FROM EMPLEADOS E JOIN ALMACENES A ON(E.CODALMA = A.CODALMA)
                 JOIN COMPRAS C ON (A.CODALMA = C.CODALMA)
                 JOIN DET_COMPRAS D ON (C.NUM_COMPRA = D.NUM_COMPRA)
WHERE CODARTI IN (SELECT CODARTI FROM DET_COMPRAS
                  GROUP BY CODARTI
                  HAVING COUNT(CODARTI) = (SELECT MAX(COUNT(CODARTI))
                                           FROM DET_COMPRAS
                                           GROUP BY CODARTI))
ORDER BY E.CODEMPLE;
```

3. Crear un bloque de código que nos permita controlar si existe el artículo que introduzcamos por teclado. Si el artículo existe, mostraremos todos los dados de sus proveedores por pantalla.

```
DECLARE
 COD_ARTICULO NUMBER(5):=&CODIGO_ARTICULO;
 CONT NUMBER(1);
 CURSOR C1 (ARTICULO NUMBER)IS SELECT *
        FROM PROVEEDORES P  JOIN SUMINISTROS S ON(P.CODPROV = S.CODPROV)
        WHERE S.CODARTI = ARTICULO;
 REG C1%ROWTYPE;
```

```
BEGIN
  SELECT COUNT(*) INTO CONT FROM ARTICULOS WHERE CODARTI=COD_ARTICULO;
  IF(CONT=0) THEN
    DBMS_OUTPUT.PUT_LINE('ERROR!!! EL ARTÍCULO INTRODUCIDO NO EXISTE');
  ELSE
    OPEN C1(COD_ARTICULO);
    FETCH C1 INTO REG;
     DBMS_OUTPUT.PUT_LINE(RPAD('NOMBRE',20) || RPAD('POBLACION',25) || RPAD('ZONA',10)
|| RPAD('COMISION',10));
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');
    WHILE C1%FOUND LOOP
              DBMS_OUTPUT.PUT_LINE(RPAD(REG.NOMBRE,20) || RPAD(REG.POBLACION,25) ||
RPAD(REG.ZONA,10) || RPAD(REG.COMISION,5));
      FETCH C1 INTO REG;
    END LOOP;
  END IF;
END;
```

4. Realiza el ejercicio anterior utilizando una solución diferente para comprobar si existe algún proveedor del artículo introducido.

```
DECLARE
 COD_ARTICULO NUMBER(5):=&CODIGO_ARTICULO;
 CONT NUMBER(5);
 CURSOR C1 (ARTICULO NUMBER)IS SELECT *
        FROM PROVEEDORES P  JOIN SUMINISTROS S ON(P.CODPROV = S.CODPROV)
        WHERE S.CODARTI = ARTICULO;
 REG C1%ROWTYPE;
BEGIN
  SELECT COUNT(CODPROV) INTO CONT FROM SUMINISTROS WHERE CODARTI=COD_ARTICULO;
  OPEN C1(COD_ARTICULO);
  FETCH C1 INTO REG;
  IF (C1%FOUND) THEN
   DBMS_OUTPUT.PUT_LINE(RPAD('NOMBRE',20) || RPAD('POBLACION',25) || RPAD('ZONA',10) ||
RPAD('COMISION',10));
  DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');
  ELSE
     DBMS_OUTPUT.PUT_LINE('ERROR!!! NO EXISTE NINGÚN PROVEEDOR PARA EL ARTÍCULO!!!!');
  END IF;
  WHILE C1%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(REG.NOMBRE,20)  ||  RPAD(REG.POBLACION,25)  ||
RPAD(REG.ZONA,10) || RPAD(REG.COMISION,5));
    FETCH C1 INTO REG;
  END LOOP;
  CLOSE C1;
END;
```

5. Crear un procedimiento al que le pasaremos desde un bloque de código como parámetro un número de compra (que introducirá el usuario) para que nos muestre por pantalla la denominación de cada uno de los artículos de la compra, su categoría, las unidades vendidas en una compra y el precio del artículo.

   Tendremos que comprobar que el número de la compra sea correcto, en caso de no ser correcto, no se realizará la llamada al procedimiento y mostraremos un mensaje de error.

   La salida se realizará el procedimiento será la siguiente forma:

```
NUMERO_COMPRA: xxxxxxxx                                    FECHA_COMPRA: xxxxxx
-------------------------------------------------------------------------------
          DENOMINACION    CATEGORIA    UNIDADES   PRECIO    SUBTOTAL
          ------------    ---------    --------   ------    --------
          xxxxxxxxxxxx    xxxxxxxxx    xxxxxxxx   xxxxxx    xxxxxxxx
          xxxxxxxxxxxx    xxxxxxxxx    xxxxxxxx   xxxxxx    xxxxxxxx
-------------------------------------------------------------------------------
TOTAL_ARTÍCULOS:xxxxx        TOTAL_UNIDADES:xxxxx        TOTAL_COMPRA: xxxxx
```

```
CREATE OR REPLACE PROCEDURE P1 (COMPRA NUMBER)IS
 CONT NUMBER(1);
  CURSOR C1 (COMPRA NUMBER)IS SELECT C.NUM_COMPRA, C.FECHA, A.DENOMINACION,
A.CATEGORIA,A.PVP,D.UNIDADES
        FROM DET_COMPRAS D  JOIN ARTICULOS A ON(D.CODARTI = A.CODARTI)
                JOIN  COMPRAS C ON (C.NUM_COMPRA = D.NUM_COMPRA)
        WHERE D.NUM_COMPRA = COMPRA;
 REG C1%ROWTYPE;
 TOTAL_ARTICULOS NUMBER:=0;
 TOTAL_UNIDADES NUMBER:=0;
 TOTAL_COMPRA NUMBER:=0;
BEGIN
  OPEN C1(COMPRA);
  FETCH C1 INTO REG;
       DBMS_OUTPUT.PUT_LINE(RPAD(('NÚMERO  COMPRA:  '||  REG.NUM_COMPRA),60)  ||
RPAD(('FECHA: ' || REG.FECHA),25));
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------');
         DBMS_OUTPUT.PUT_LINE(RPAD('DENOMINACION',20)  ||  RPAD('CATEGORIA',20)  ||
RPAD('UNIDADES',15) || RPAD('PVP',10)|| RPAD('SUBTOTAL',10));
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------');
  WHILE C1%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(REG.DENOMINACION,20) || RPAD(REG.CATEGORIA,20)
    || RPAD(REG.UNIDADES,15) || RPAD(REG.PVP,10)|| RPAD((REG.PVP*REG.UNIDADES),5));

    TOTAL_ARTICULOS := TOTAL_ARTICULOS + 1;
    TOTAL_UNIDADES := TOTAL_UNIDADES + REG.UNIDADES;
    TOTAL_COMPRA := TOTAL_COMPRA + (REG.PVP * REG.UNIDADES);

     FETCH C1 INTO REG;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------------------------------');
       DBMS_OUTPUT.PUT_LINE(RPAD(('TOTAL  ARTÍCULOS:  '  ||  TOTAL_ARTICULOS),25)  ||
RPAD(('TOTAL  UNIDADES:  '  ||  TOTAL_UNIDADES),25)  ||  RPAD((' TOTAL  COMPRA:  '  ||
TOTAL_COMPRA),20));
  CLOSE C1;
END;


--PROGRAMA PRINCIPAL DESDE DONDE LLAMAMOS AL PROCEDIMIENTO. RECUERDA QUE VAN POR
--SEPARADO.


DECLARE
 COMPRA NUMBER(5):=&NUMEROCOMPRA;
 CONT NUMBER(1);
BEGIN
          SELECT  COUNT(NUM_COMPRA)  INTO  CONT  FROM  DET_COMPRAS  WHERE
NUM_COMPRA=COMPRA;
  IF(CONT=0) THEN
    DBMS_OUTPUT.PUT_LINE('ERROR!!!  NO EXISTE EL CÓDIGO DE COMPRA INDICADO');
  ELSE
  P1(COMPRA);
  END IF;
END;
```

6. Crear un procedimiento que a partir de un código de almacén nos muestre todas sus compras y para compra todos sus artículos. El código de almacén se pasará como parámetro al procedimiento. En caso de que no exista el código del almacén, mostraremos un mensaje que así lo indique y se realizará la llamada al procedimiento.

   La salida se realizará de la siguiente forma:

```
NUMERO_COMPRA: xxxxxxxx
--------------------------------------------------------------------------------
   DENOMINACION   UNIDADES   PRECIO
   ------------   --------   ------
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
--------------------------------------------------------------------------------

NUMERO_COMPRA: xxxxxxxx
--------------------------------------------------------------------------------
   DENOMINACION   UNIDADES   PRECIO
   ------------   --------   ------
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
--------------------------------------------------------------------------------

NUMERO_COMPRA: xxxxxxxx
--------------------------------------------------------------------------------
   DENOMINACION   UNIDADES   PRECIO
   ------------   --------   ------
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
   xxxxxxxxxxxx   xxxxxxxx   xxxxxx
--------------------------------------------------------------------------------
```

```
CREATE OR REPLACE PROCEDURE P2 (ALMACEN NUMBER)IS
 CONT NUMBER(1);
 CURSOR C1 (ALMACEN NUMBER) IS SELECT NUM_COMPRA
                  FROM ALMACENES A JOIN COMPRAS C ON (A.CODALMA = C.CODALMA)
                  WHERE A.CODALMA = ALMACEN;

 CURSOR C2 (COMPRA NUMBER)IS SELECT  A.DENOMINACION, D.UNIDADES,A.PVP
                FROM DET_COMPRAS D  JOIN ARTICULOS A ON(D.CODARTI = A.CODARTI)
                          JOIN  COMPRAS C ON (C.NUM_COMPRA = D.NUM_COMPRA)
                WHERE D.NUM_COMPRA = COMPRA;
 REGCOMPRAS C1%ROWTYPE;
 REGARTICULOS C2%ROWTYPE;
BEGIN
  OPEN C1(ALMACEN);
  FETCH C1 INTO REGCOMPRAS;
  WHILE C1%FOUND LOOP
     DBMS_OUTPUT.PUT_LINE(RPAD(('NÚMERO COMPRA: '|| REGCOMPRAS.NUM_COMPRA),60));
     DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------');
               DBMS_OUTPUT.PUT_LINE(RPAD('DENOMINACION',20)   ||   RPAD('UNIDADES',15)   ||
RPAD('PVP',10));
     DBMS_OUTPUT.PUT_LINE('--------------------------------------------------------------------------------');
     OPEN C2(REGCOMPRAS.NUM_COMPRA);
     FETCH C2 INTO REGARTICULOS;
     WHILE C2%FOUND LOOP
                              DBMS_OUTPUT.PUT_LINE(RPAD(REGARTICULOS.DENOMINACION,20)   ||
RPAD(REGARTICULOS.UNIDADES,15) || RPAD(REGARTICULOS.PVP,10));
        FETCH C2 INTO REGARTICULOS;
     END LOOP;
     CLOSE C2;
     FETCH C1 INTO REGCOMPRAS;
   END LOOP;
   CLOSE C1;
END;


--PROGRAMA PRINCIPAL DESDE DONDE LLAMAMOS AL PROCEDIMIENTO. RECUERDA QUE VAN POR
--SEPARADO.


DECLARE
 ALMACEN NUMBER(5):=&NUMEROALMACEN;
 CONT NUMBER(1);
BEGIN
   SELECT COUNT(CODALMA) INTO CONT FROM ALMACENES WHERE CODALMA=ALMACEN;
```

```
  IF(CONT=0) THEN
    DBMS_OUTPUT.PUT_LINE('ERROR!!!  NO EXISTE EL CÓDIGO DE ALMACEN INDICADO');
  ELSE
  P2(ALMACEN);
  END IF;
END;
```

7. Crear una función que nos devuelva el código del empleado que tenga el mayor sueldo del almacén que introduzca el usuario. La función será llamada desde un bloque donde se mostrarán los datos del empleado. A la función le paremos como parámetro el código del almacén y nos devolverá el código del empleado.

   En caso de que el almacén no exista, o se produzca otro error. Se mostrará un mensaje y la función no realizará ninguna acción.

```
CREATE OR REPLACE FUNCTION F1 (ALMACEN NUMBER)RETURN NUMBER IS
   EMPLEADO NUMBER(5);
BEGIN
   SELECT CODEMPLE INTO EMPLEADO FROM EMPLEADOS E JOIN ALMACENES A ON(E.CODALMA =
A.CODALMA)
                  WHERE SALARIO = (SELECT MAX(SALARIO) FROM EMPLEADOS WHERE  CODALMA
= ALMACEN)
                  AND E.CODALMA = ALMACEN;
  RETURN EMPLEADO;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('ERROR!!!  EL ALMACEN NO EXISTE');
       RETURN 0;
  WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE('ERROR!!!  EXISTEN DEMASIADOS VALORES');
      RETURN 0;
  WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE('ERROR!!!  LA OPERACIÓN NO SE PUEDE REALIZAR');
     RETURN 0;
END F1;


--PROGRAMA PRINCIPAL DESDE DONDE LLAMAMOS AL PROCEDIMIENTO. RECUERDA QUE VAN POR
--SEPARADO.


DECLARE
  ALMACEN NUMBER(5):= &NUMEROALMACEN;
  EMPLEADO NUMBER(5);
  DATOS_EMPLEADO EMPLEADOS%ROWTYPE;
BEGIN
   EMPLEADO := F1(ALMACEN);
   IF(EMPLEADO != 0) THEN
     SELECT * INTO DATOS_EMPLEADO FROM EMPLEADOS WHERE CODEMPLE=EMPLEADO;
          DBMS_OUTPUT.PUT_LINE(RPAD('NOMBRE',20) || RPAD('POBLACION',20) || RPAD('FECHA
ALTA',15) || RPAD('SALARIO',10));
     DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------------------');
                              DBMS_OUTPUT.PUT_LINE(RPAD(DATOS_EMPLEADO.NOMBRE,20)     ||
RPAD(DATOS_EMPLEADO.POBLACION,20)
     || RPAD(DATOS_EMPLEADO.FECHAALTA,15) || RPAD(DATOS_EMPLEADO.SALARIO,5));
   END IF;
END;
```

8. Crear una función que devolverá el código del proveedor que introduzca el usuario por teclado, esta información será pasada a un procedimiento que se encargará de

mostrar los datos del proveedor introducido o bien un mensaje de error en caso de que no exista el proveedor.

Ni la función, ni el procedimiento se almacenarán en la base de datos.

```
DECLARE
  PROVEEDOR NUMBER(5);
  DATOS_PROVEEDOR PROVEEDORES%ROWTYPE;

  -- funcion de bloque
  FUNCTION F2 RETURN NUMBER IS
    PROVEEDOR NUMBER(5):=&CODIGOPROVEEDOR;
  BEGIN
    RETURN PROVEEDOR;
  END F2;

  -- procedimiento de bloque
  PROCEDURE P3(PROVEEDOR NUMBER)IS
  BEGIN
    SELECT * INTO DATOS_PROVEEDOR FROM PROVEEDORES WHERE CODPROV=PROVEEDOR;
     DBMS_OUTPUT.PUT_LINE(RPAD('NOMBRE',20) || RPAD('POBLACION',25) || RPAD('ZONA',10)
|| RPAD('COMISION',10));
    DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------');
                          DBMS_OUTPUT.PUT_LINE(RPAD(DATOS_PROVEEDOR.NOMBRE,20)    ||
RPAD(DATOS_PROVEEDOR.POBLACION,25)
    || RPAD(DATOS_PROVEEDOR.ZONA,10) || RPAD(DATOS_PROVEEDOR.COMISION,5));

  EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('ERROR!!!  EL PROVEEDOR NO EXISTE');
  END P3;

BEGIN
  --llamadas a la función y al procedimiento
  P3(F2);
END;
```

9. Crear un trigger que nos permita controlar si existe el proveedor y el artículo cuando insertemos un registro en la tabla de suministros.

```
CREATE OR REPLACE TRIGGER VALIDA_INSERT_SUMINISTROS BEFORE INSERT ON SUMINISTROS
FOR EACH ROW
DECLARE
  PROVEEDOR NUMBER(5);
  ARTICULO NUMBER(5);
  SUMINISTRO SUMINISTROS%ROWTYPE;
BEGIN
  SELECT CODPROV INTO PROVEEDOR FROM PROVEEDORES WHERE CODPROV = :NEW.CODPROV;
  SELECT CODARTI INTO ARTICULO FROM ARTICULOS WHERE CODARTI = :NEW.CODARTI;
    SELECT * INTO SUMINISTRO FROM SUMINISTROS WHERE CODARTI = :NEW.CODARTI AND
CODPROV = :NEW.CODPROV;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('LOS DATOS NO EXISTEN');
  WHEN DUP_VAL_ON_INDEX  THEN
    DBMS_OUTPUT.PUT_LINE('LOS DATOS NO EXISTEN');
END;
```

10. Crear un trigger que almacene en la tabla de HISTORIAL_EMPLEADOS (que no existe) el usuario, la fecha y la operación que se ha realizado. El trigger se disparará cada vez que se realice una operación sobre la tabla de empleados.

```
CREATE OR REPLACE TRIGGER HISTORICO_EMPLEADOS AFTER INSERT OR UPDATE OR DELETE ON
EMPLEADOS FOR EACH ROW
BEGIN
```

```
IF INSERTING THEN
   INSERT INTO HISTORIAL_EMPLEADOS VALUES ( 'USUARIO: ' || USER || ' FECHA: ' || SYSDATE || '
HORA: '|| TO_CHAR(SYSDATE,'HH:MM')|| ' OPERACION: INSERCIÓN' );
END IF;

IF DELETING THEN
   INSERT INTO HISTORIAL_EMPLEADOS VALUES ( 'USUARIO: ' || USER || ' FECHA: ' || SYSDATE || '
HORA: '|| TO_CHAR(SYSDATE,'HH:MM')|| ' OPERACION: BORRADO');
 END IF;

IF UPDATING THEN
   INSERT INTO HISTORIAL_EMPLEADOS VALUES ( 'USUARIO: ' || USER || ' FECHA: ' || SYSDATE || '
HORA: '|| TO_CHAR(SYSDATE,'HH:MM')|| ' OPERACION: MODIFICACIÓN');
 END IF;
END;
```

11. Crear un trigger que se ejecute cuando modifiques el stock de un artículo, de tal forma que no pueda tomar valores negativos.

    Si al realizar la modificación el campo stock tomara un valor negativo, le asignaremos 10 unidades.

    Si al realizar la modificación el campo stock el valor nuevo fuera menor que el valor que tenía anteriormente el campo no se realizará la modificación y se almacenará el stock que había anteriormente.

```
CREATE OR REPLACE TRIGGER MOFIDIFICAUNIDADES  BEFORE UPDATE OF STOCK ON ARTICULOS
FOR EACH ROW
 DECLARE
 CUENTA NUMBER(4);
 BEGIN
  IF:NEW.STOCK <= 0 THEN
     :NEW.STOCK :=10;
  ELSE
    IF:NEW.STOCK < :OLD.STOCK THEN
     :NEW.STOCK := :OLD.STOCK;
     END IF;
  END IF;
 END;
```

12. Crear un trigger que se ejecute cuando se realice un cambio de los datos de un empleado de tal forma que realice las siguientes comprobaciones sobre los nuevos datos del empleado:

    a. La fecha de alta del nuevo contrato del empleado debe ser mayor o igual a la fecha de hoy. En caso contrario, se le asignará la fecha de hoy.
    b. El salario debe ser mayor que 0 y mayor que el salario anterior. Si no es así, se le asignará un salario un 10% mayor del salario que tenía anteriormente.
    c. El código de almacén debe existir en la tabla de almacenes. Si el nuevo código de almacén no se existe le asignaremos el código que tenía anteriormente.

```
CREATE OR REPLACE TRIGGER MODIFICAEMPLEADOS BEFORE UPDATE ON EMPLEADOS FOR EACH
ROW
DECLARE
  CONT NUMBER;  CONT1 NUMBER;
BEGIN
  IF(:NEW.FECHAALTA < SYSDATE) THEN
    :NEW.FECHAALTA := SYSDATE;
  END IF;

  IF(:NEW.SALARIO < :OLD.SALARIO) THEN
    :NEW.SALARIO := :OLD.SALARIO*1.10;
  END IF;
```

```
 SELECT COUNT(*) INTO CONT FROM ALMACENES WHERE CODALMA= :NEW.CODALMA;
  IF(CONT=0) THEN
     :NEW.CODALMA:= :OLD.CODALMA;
  END IF;
END;
```