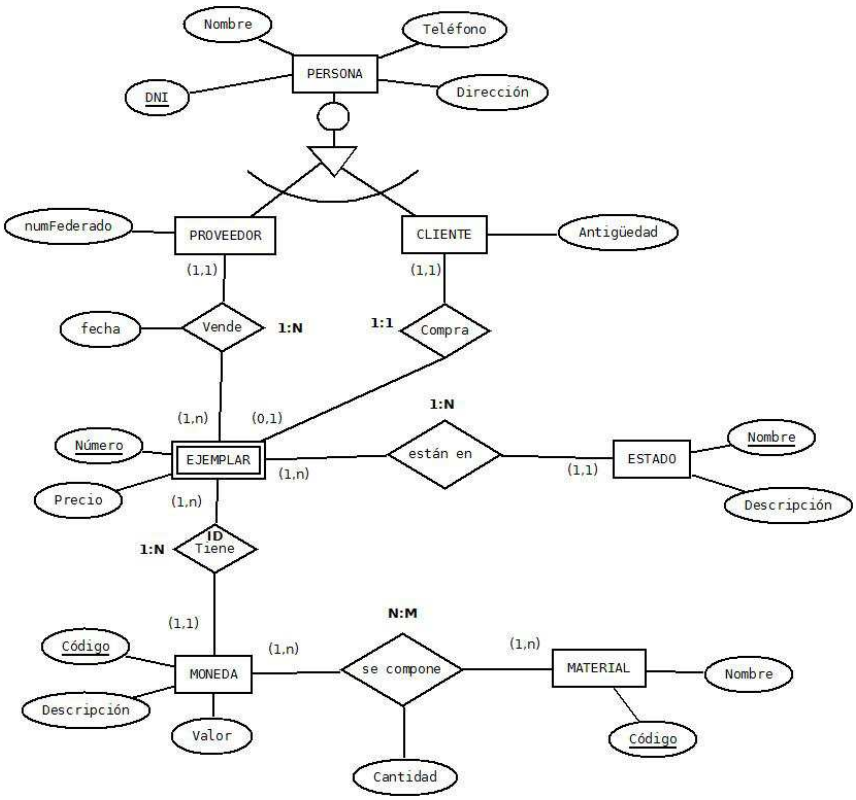
Módulo Profesional: BASES DE DATOS
I.E.S.: Aguadulce **C.F.G.S. : Desarrollo de Aplicaciones Multiplataforma (DAM) y Desarrollo de Aplicaciones Web (DAW)**
Fecha: 10 de Junio de 2015 **Hora de Comienzo: 15:30h**
Examen: FINAL JUNIO **Duración: 2 horas**
Nombre: **D.N.I.:**
Centro en el que se realiza el examen:

PRIMER PARCIAL. SOLUCIÓN PARTE PRÁCTICA (7 PUNTOS)

1. REALIZA EL PASO A TABLAS DEL SIGUIENTE MODELO ENTIDAD RELACIÓN. (2 Puntos)



SOLUCIÓN:

PERSONA (**DNI**, Nombre, Teléfono, Dirección)

CLIENTE (**DNI_c,** Antigüedad)

PROVEEDOR (**DNI_p**, numFederado)

EJEMPLAR (**Código_m, Número**, precio, *Dni_p, Dni_c, Nombre_estado*)

ESTADO (**Nombre**, Descripción)

MATERIAL (**Código**, nombre)

SE_COMPONE (***Código_moneda, Código_material***, cantidad)

MONEDA (**Código**, Descripción, valor)

2. REALIZA EL MODELO ENTIDAD – RELACIÓN DEL SIGUIENTE SUPUESTO (2 PUNTOS)

Una Escuela de Informática quiere generar un sistema para tener controlado en una base de datos todo lo referente a los Trabajos Fin de Carrera: alumnos que los realizan, profesores que los dirigen, temas de los que tratan y tribunales que los corrigen. Por tanto, interesa conocer:

• De los alumnos: su número de matrícula, DNI y nombre. Un alumno realiza, evidentemente, sólo un T.F.C que es dirigido por único profesor.

• De los T.F.C: su nombre (qué es único), tema, y fecha de comienzo. Un T.F.C. determinado, no puede ser realizado por varios alumnos.

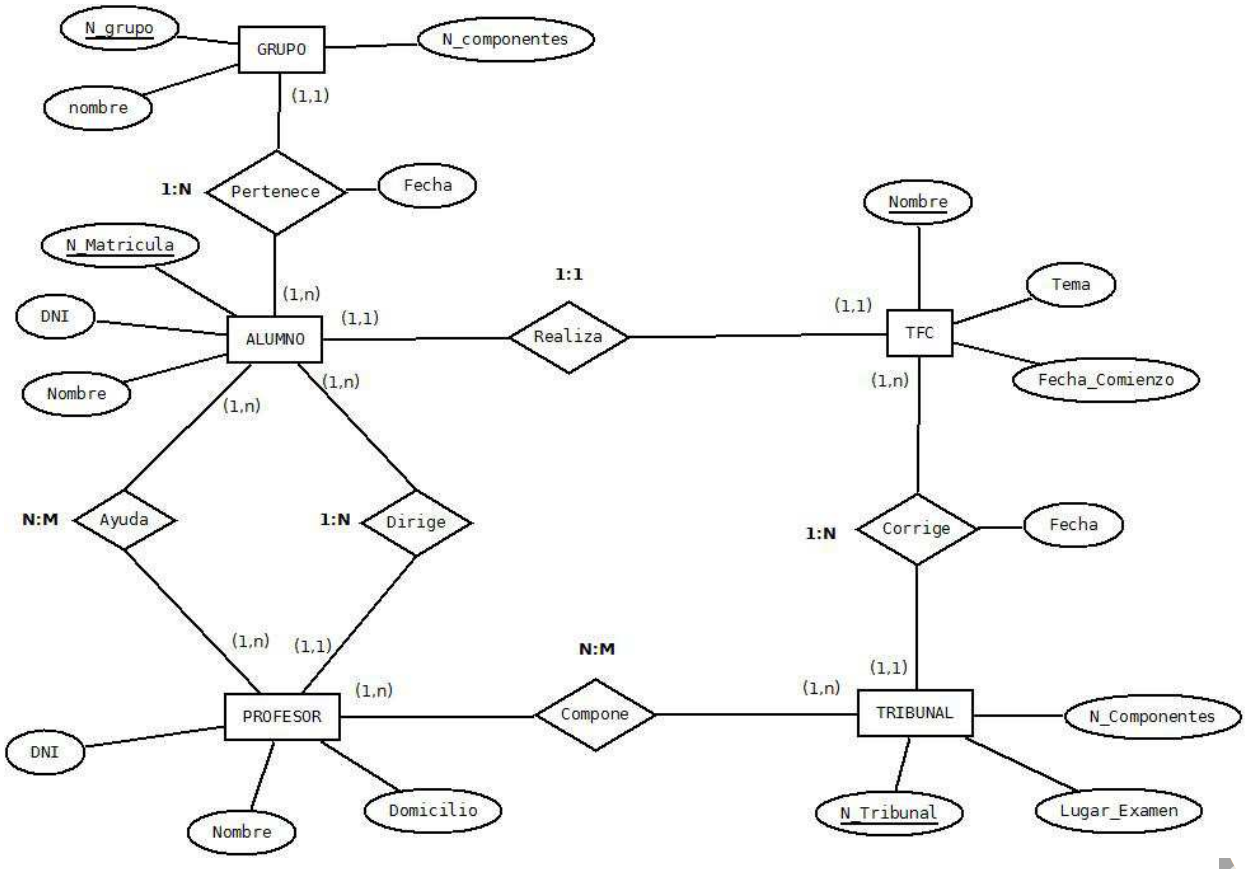
• De los profesores: su DNI, nombre y domicilio; y puesto que los T.F.C. son del área en el que trabaja, NO interesa conocer los T.F.C. que dirige sino a qué alumnos se los dirige.

• De los tribunales: número de tribunal, lugar de examen y número de componentes. Un Tribunal está formado por varios profesores y los profesores pueden formar parte de varios tribunales. Un tribunal puede corregir varios T.FC siendo de interés la fecha en que se corrige cada T.F.C. Por otra parte un T.F.C sólo puede ser corregido por un tribunal.

• Al margen de esto, un alumno puede haber pertenecido a algún grupo de investigación del que haya surgido la idea del T.F.C. Dichos grupos se identifican por un número de grupo, su nombre y por su número de componentes. Un alumno no puede pertenecer a más de un grupo y NO es de interés saber si el grupo tiene algo que ver o no con el T.F.C. del alumno; sí siendo de interés la fecha de incorporación a dicho grupo.

• Por otra parte, un profesor, al margen de dirigir el T.F.C. de algunos alumnos, puede haber colaborado con otros en la realización de dicho T.F.C. pero siendo otro profesor el que lo dirige. En este caso, sólo es interesante conocer qué profesor ha ayudado a qué alumno (a un alumno le pueden ayudar varios profesores).

SOLUCIÓN:



3.- Utilizando sentencias SQL crea las siguientes tablas en el orden apropiado, utilizando para cada campo el tipo de dato más adecuado (tipos de datos Oracle). Las tablas con sus correspondientes campos son las siguientes:(3 Puntos)

- Tabla **PAQUETE** que almacena una referencia única, el nombre del destinatario, su dirección completa, el peso del paquete en gramos, el precio, la fecha de envío, la fecha de entrega, un campo de observaciones y por último el campo DNI del cliente que envía el paquete.
- Tabla **CLIENTE** que almacena el DNI, el nombre, la dirección, el teléfono, la ciudad y el código postal de la persona que envía el paquete.

En las mismas sentencias SQL de creación de tablas debes crear las siguientes restricciones: (***NOTA***: *No crear más restricciones de las que se piden en el ejercicio*)

1. Elige la clave primaria adecuada para cada una de las tablas.
2. El campo nombre del destinatario no puede estar vacío.
3. El campo peso del paquete no puede tener valores inferiores a 0 gramos.
4. Establece la relación entre las tablas. Se debe cumplir la regla de integridad referencial.

Posteriormente realiza con SQL las siguientes operaciones:

1. Establece la restricción para que la fecha de entrega de los paquetes no pueda ser anterior a la fecha de envío.
2. Añade el campo “tipo de envío” a la tabla paquete.
3. Establece la restricción para que el tipo de envío sólo pueda tomar los valores “Normal” o “Urgente”
4. Elimina el campo observaciones de la tabla paquete.

SOLUCIÓN:

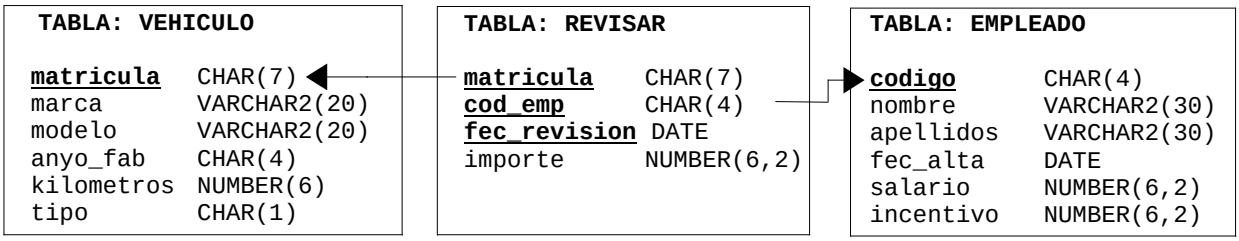
```
CREATE TABLE CLIENTE(
DNI CHAR(9),
NOMBRE VARCHAR2(50),
DIRECCION VARCHAR2(50),
TELEFONO CHAR(9),
CIUDAD VARCHAR2(20),
COD_POSTAL CHAR(5),
CONSTRAINT CLI_DNI_PK PRIMARY KEY (DNI)
);

CREATE TABLE PAQUETE (
REFERENCIA CHAR(10) PRIMARY KEY,
NOMBRE_DESTINATARIO VARCHAR2(50) NOT NULL,
DIRECCION VARCHAR2(50),
PESO NUMBER(6,1),
PRECIO NUMBER(6,2),
FECHA_ENVIO DATE,
FECHA_ENTREGA DATE,
```

```
OBSERVACIONES VARCHAR2(100),
DNI_CLI CHAR(9),
CONSTRAINT PAQ_DNI_fk FOREIGN KEY (DNI_CLI) REFERENCES CLIENTE(DNI) ON DELETE
CASCADE,
CONSTRAINT PAQ_PES_ck CHECK (PESO>0)
);
```

1. `ALTER TABLE PAQUETE ADD CONSTRAINT PAQ_FEC_CK CHECK (FECHA_ENTREGA > FE-CHA_ENVIO);`
2. `ALTER TABLE PAQUETE ADD TIPO_ENVIO CHAR(20);`
3. `ALTER TABLE PAQUETE ADD CONSTRAINT PAQ_TIP_CK CHECK (TIPO_ENVIO IN ('Normal','Urgente'));`
4. `ALTER TABLE PAQUETE DROP COLUMN OBSERVACIONES;`

SEGUNDO PARCIAL. SOLUCIÓN PARTE PRÁCTICA: (7 PUNTOS)

Un taller de vehículos ha informatizado su nuevo servicio de *"Puesta a punto"* y ofrece a sus clientes la posibilidad de revisar sus vehículos antes de pasar la ITV (Inspección Técnica de Vehículos). Para ello, disponemos entre otras tablas de la BD las siguientes:



NOTA: Las claves primarias se representan subrayadas y en negrita y las claves ajenas de la tabla REVISAR con las flechas que referencian a sus tablas correspondientes.

1.- Utilizando las tablas y campos necesarios, se pide obtener las sentencias SQL de consulta que permitan realizar los siguientes apartados: (1,5 puntos)

a) Obtener todos los datos del empleado con menos antigüedad (fecha de alta) en el taller.

```
SELECT * FROM EMPLEADO WHERE fec_alta >=(SELECT max(fec_alta) FROM EMPLEADO)
```

b) Obtener todas las matrículas, marcas y modelos de aquellos vehículos de tipo 'T' (turismo) que ha revisado el empleado de código 0033

```
SELECT V.matricula, V.marca, V.modelo
FROM VEHICULO V, REVISAR R
WHERE V.matricula = R.matricula
AND V.tipo = 'T'
AND cod_emp = '0033'
```

c) Obtener un listado con la fecha de revisión con formato "10 de junio de 2015" seguida de todos los campos de la tabla vehículo y también al nombre y apellidos del empleado que realizó la revisión. El listado debe ordenarse por la fecha de revisión desde la más reciente a la más lejana en el tiempo.

```
SELECT TO_CHAR(fec_revision,'DD') || ' de ' || TO_CHAR(fec_revision,'MONTH') || ' de ' ||
TO_CHAR(fec_revision,'YYYY'), V.*, E.nombre, E.apellidos
FROM VEHICULO V, REVISAR R, EMPLEADO E
WHERE V.matricula = R.matricula
AND E.codigo = R.cod_emp
ORDER BY R.fec_revision DESC
```

d) Obtener por cada nombre y apellidos de empleados, el total de revisiones que se han realizado durante todo el año 2014 ordenados de más a menos revisiones realizadas.

```
SELECT E.nombre, E.apellidos, COUNT(R.matricula)
FROM EMPLEADO E, REVISAR R
WHERE E.codigo = R.cod_emp
AND R.fec_revision BETWEEN '01/01/2014' AND '12/31/2014'
GROUP BY E.nombre, E.apellidos
ORDER BY 3 DESC
```

e) Obtener todos los datos de los vehículos que han gastado en el total de sus revisiones un importe superior a 300€.

```
SELECT * FROM VEHICULO
WHERE matricula IN (SELECT matricula
                    FROM REVISAR
                    GROUP BY matricula
                    HAVING SUM(importe)>300)
```

2.- Teniendo en cuenta las mismas tablas del ejercicio anterior, debes realizar las siguientes operaciones de actualización, inserción y borrado de registros: (1,5 puntos)

a) Insertar un nuevo vehículo con matrícula 1234JKL de marca Opel y modelo Corsa, año fabricación 2015 y con 1450 kilómetros. El tipo de vehículo es un turismo por lo que deberá almacenarse el valor 'T'. A continuación crea otra sentencia que almacene la revisión de éste vehículo en la fecha actual del sistema por el empleado de código 0033 sin introducir de momento el importe.

```
INSERT INTO VEHICULO VALUES ('1234JKL', 'Opel','Corsa','2015', 1450, 'T');
INSERT INTO REVISAR VALUES ('1234JKL','0033',SYSDATE,null);
```

b) Elimina todos los vehículos que no hayan vuelto a pasar una revisión en el taller desde el comienzo del 2010 hasta la fecha actual. Realiza la operación en una única sentencia.

```
DELETE FROM VEHICULO
WHERE matricula NOT IN (SELECT matricula FROM REVISAR
                        WHERE fec_revision > '01/01/2010)
```

c) Reduce el salario en un 10% y el incentivo en un 2% para aquellos empleados cuyas ganancias (salario + incentivo) superen la media de las ganancias de todos los empleados del taller.

```
UPDATE EMPLEADO SET salario = (salario - salario * 0.1), incentivo = (incentivo - incentivo * 0.02)
WHERE (salario + incentivo) > (SELECT AVG(salario + incentivo) FROM EMPLEADO)
```

d) Incrementa el número de kilómetros en 15.000 de los vehículos cuyas matrículas son 1111XYZ, 2222XYZ y 3333XYZ. Realiza la operación en una única sentencia.

```
UPDATE VEHICULO SET kilometros = kilometros + 15000
WHERE MATRICULA IN ('1111XYZ','2222XYZ','3333XYZ')
```

e) Borra el empleado o empleados que menos revisiones han realizado.

```
DELETE FROM EMPLEADO WHERE CODIGO IN (
        SELECT R.COD_EMP
        FROM REVISAR R
        GROUP BY COD_EMP
        HAVING COUNT(R.MATRICULA) = (SELECT MIN(COUNT(R.MATRICULA))
                                     FROM REVISAR R
                                     GROUP BY COD_EMP))
```

3.- Utilizando las mismas tablas, realiza con el lenguaje PL/SQL los siguientes apartados: (3 puntos)

a) Crea un procedimiento que reciba como parámetro el código de un empleado y devuelva la matrícula, marca y modelo de los vehículos que ha revisado. (1,25 puntos)

```
CREATE OR REPLACE PROCEDURE muestra_vehiculos (codigo_emple  VARCHAR2)
AS
/*Declaración de cursor */
CURSOR cvehiculos IS
SELECT v.matricula, v.marca, v.modelo FROM VEHICULO V, REVISAR  R
WHERE(v.matricula=r.matricula)  AND UPPER(codigo_emple) = UPPER(e.codigo);

/*Declaración de variables */
fvehiculo  cvehiculos%ROWTYPE;

BEGIN
/*Muestra por pantalla */
DBMS_OUTPUT.PUT_LINE ('+----------------------------------------------------------------------------------------+');
DBMS_OUTPUT.PUT_LINE ('Listado de vehículos revisados por el empleado ' || codigo_emple);
DBMS_OUTPUT.PUT_LINE ('+----------------------------------------------------------------------------------------+');
DBMS_OUTPUT.PUT_LINE('Matrícula ' || 'Marca   ' || 'Modelo   '  );

/*Abre el cursor */
OPEN cvehiculos;

/*Lee la primera fila recuperada por el cursor y la almacena en la variable fvehiculo*/
FETCH cvehiculos INTO fvehiculo;

/*Bucle de control, mientras el cursor devuelva resultados se mostrará por pantalla el valor recuperado y se leerá la
siguiente fila del cursor */
WHILE cvehiculos%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(fvehiculo.matricula || ' ' || fvehiculo.marca || ' ' || fvehiculo.modelo);
        FETCH cvehiculos INTO fvehiculo;
END LOOP;

/*Cierra el cursor */
CLOSE cvehiculos;
END muestra_vehiculos;
```

b) Crea una función que reciba como parámetros el nombre de un empleado y una fecha de revisión y devuelva el número de vehículos revisados por el empleado en esa fecha. (1 punto)

```
CREATE OR REPLACE FUNCTION Numero_vehiculos
        (nombre_emple empleado.nombre%TYPE, fecha revisar.fec_revision%TYPE)
RETURN NUMBER
IS

/*Declaración de variables */
Num_vehiculos NUMBER(4);
codigo_emp REVISAR.codigo%TYPE;

BEGIN
        SELECT codigo into codigo_emp FROM EMPLEADO WHERE nombre=nombre_emple;
        SELECT count(matricula) INTO num_vehiculos FROM Revisar
                WHERE (codigo_emp=cod_emp) AND (fec_revision=fecha)

/*Valor de retorno de la función */
RETURN num_vehiculos;

END numero_vehiculos;
```

c) Crea un disparador (Trigger) de tal forma que cada vez que actualice el salario de un empleado se almacene en una tabla HISTORIAL_SALARIOS el usuario (con el que estamos autenticado en oracle), código del empleado, nombre del empleado, fecha actual del sistema, salario anterior y salario nuevo (0,75 puntos)
Nota: La tabla HISTORIAL_SALARIOS la tendrás que crear previamente con los campos necesarios.

```
CREATE TABLE HISTORIAL_SALARIOS (
USUARIO VARCHAR2(30);
CODIGO CHAR(4);
NOMBRE VARCHAR2(30);
FECHA DATE;
SALARIO_OLD NUMBER(6,2);
SALARIO_NEW NUMBER(6,2);
);
CREATE OR REPLACE TRIGGER Historial_salarios
        BEFORE INSERT ON EMPLEADO
        FOR EACH ROW
        DECLARE7
        Usuario VARCHAR2(30);
        BEGIN
                SELECT SYS_CONTEXT('userenv', 'current_user') INTO usuario FROM dual,
                INSERT INTO historial_salarios VALUES(usuario, :old.codigo, :old.nombre, sysdate, :old.salario, :new.salario);
        END;
```

4.- Crea el tipo de objeto "EMPLEADO" con los siguientes atributos: (1 punto)

Codigo CHAR(4);
Nombre VARCHAR2(30);
Apellidos VARCHAR2(30);
Fec_alta DATE;
Salario NUMBER(8,2);
Incentivo NUMBER(8,2);

Crea también un método para el tipo de objetos "EMPLEADO" que devuelva el salario NETO del empleado. El salario bruto se calcula sumando el salario más el incentivo. El salario neto se calcula restando al salario bruto la cantidad correspondiente al aplicarle un porcentaje sobre el salario bruto. Ese porcentaje es diferente dependiendo del valor del salario bruto. Si éste es menor que 20000 € entonces el salario bruto se reduce en un 10%, sin embargo si es mayor o igual que 20000€, el salario bruto se reduce en un 15%.

```
CREATE OR REPLACE TYPE Empleado AS OBJECT (
Codigo CHAR(4),
Nombre VARCHAR2(30),
Apellidos VARCHAR2(30),
Fec_alta  DATE,
Salario NUMBER(8,2),
Incentivo NUMBER(8,2),

MEMBER FUNCTION SalarioNeto RETURN NUMBER
);

CREATE OR REPLACE TYPE BODY Empleado AS
MEMBER FUNCTION SalarioNeto RETURN NUMBER
IS
sal_bruto NUMBER;

BEGIN
sal_bruto:=(salario+incentivo);
        IF (sal_bruto<20000) THEN
                RETURN(sal_bruto-(sal_bruto)*10/100);
        ELSE
                RETURN(sal_bruto-(sal_bruto)*15/100);
        END IF;
END salarioneto;

END;
```

ANEXO: Estructuras PL/SQL que pueden ser útiles para la realización del examen práctico.

Procedimiento en PL/SQL

```
CREATE [OR REPLACE] PROCEDURE nombre_procedimiento ([parámetros])
IS
[DECLARE]
      [<variables locales>]
BEGIN
      <código del procedimiento>
[EXCEPTION]
END [nombre_procedimiento];
```

Función en PL/SQL

```
CREATE [OR REPLACE] FUNCTION nombre_función ([parámetros])
RETURN tipodato IS
[DECLARE]
      [<variables locales>]
BEGIN
      <código de la función>
RETURN <valor>;
[EXCEPTION]
END [nombre_función];
```

Disparador en PL/SQL

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
{BEFORE|AFTER|INSTEAD OF}
{INSERT|DELETE|UPDATE [OF <atributo]} ON <tabla>
[FOR EACH ROW|STATEMENT]
[WHEN condición]
[DECLARE]
      [<variables locales>]
BEGIN
      <código del trigger>
[EXCEPTION]
END [nombre_trigger];
```

Declaración y manejo de un cursor en PL/SQL

```
CURSOR prueba IS
Sentencia de consulta;
```

```
LOOP
FETCH prueba INTO mi_registro;
EXIT WHEN prueba%NOTFOUND;
  --- se procesa el registro
END LOOP;
```

```
FETCH prueba INTO mi_registro;
WHILE prueba%FOUND LOOP
   DBMS_OUTPUT.PUT_LINE('Hola');
FETCH prueba INTO mi_registro;
END LOOP;
```

```
FOR loop_contador IN [ REVERSE ]
lim_inf .. lim_sup LOOP
   Instruccion_fulana ;
   DBMS_OUTPUT.PUT_LINE('Hola');
END LOOP;
```

Estructura de control en PL/SQL

```
IF condicion THEN
      DBMS_OUTPUT.PUT_LINE('lo que sea');
ELSE
      DBMS_OUTPUT.PUT_LINE(una_variable);
END IF;
```