Actividad 1

Realiza los siguientes apartados:

a. Ingresa en SQL*Plus con el usuario administrador.

b. Crea un usuario nuevo, que se denomine examenTUDNI (sustituyendo TUDNI por los dígitos de tu documento).

c. Otorga permisos al nuevo usuario para conectar al SGBD y para crear objetos.

d. Cierra la sesión del usuario administrador.

e. Conecta al SGBD con el usuario creado en el apartado b.

f. Visualiza el usuario y su contenedor.

```
sqlplus sys as sysdba

create user c##examen54050674 identified by xerach default tablespace
users;

grant resource, connect, dba to c##examen54050674;

disconnect

connect c##examen54050674

show user con_name
```

Actividad 2

Se tiene el análisis de la información de las siguientes tablas:

CONCESIONARIO			
Nombre Columna	**Descripción**	**Tipo Dato**	**Restricción**
codigoConcesionario	Código de un concesionario	Numérico, tamaño a elegir	Clave Primaria
denominacion	Nombre del concesionario	Alfanumérico, tamaño a elegir	Debe tener contenido.
teléfono	Teléfono del concesionario	Alfanumérico, elegir tamaño	
codigoPostal	Código Postal del concesionario	*Decidir.*	Debe tener contenido.

VEHÍCULO			
Nombre Columna	**Descripción**	**Tipo Dato**	**Restricción**
VIN	Número de Bastidor	Alfanumérico fijo 17 caract.	Clave Primaria
matrícula	Matrícula del Vehículo	Alfanumérico, elegir tamaño	Único
marca	Marca del vehículo	Alfanumérico, elegir tamaño	Debe tener contenido.
modelo	Modelo del vehículo	Alfanumérico, elegir tamaño	
peso	peso del vehículo	Decimal	Mayor que 0.
concesionario		*Decidir*	Clave Foránea

Se pide, en SQL Developer o SQL*Plus con sentencias DDL, crear un script denominado actividad2_tudni.sql, en el que se realice cada apartado precedido de un comentario:

a. (0,05 puntos) Crear una base de datos denominada gestionVehiculosTUDNI (sustituyendo TUDNI por los dígitos de tu documento).

b. (0,40 puntos) Crear, utilizando la base de datos creada en el apartado a, la tablas especificadas anteriormente aplicando las restricciones (constraints) pedidas. Se debe cumplir la integridad referencial en cascada para la actualización y no permitir la eliminación de la clave primaria en caso de existir registros en la foránea.

c. (0,20 puntos) En la tabla vehículo, añadir una columna llamada itv que únicamente acepte los valores: S o N.

d. (0,20 puntos) Eliminar cualquier columna de la tabla que se desee.

e. (0,15 puntos) Conectar a SQL*Plus como administrador, ejecutar el script y otorgar todos los permisos al usuario generado en la actividad 1 sobre la tabla vehículo.

NOTA: EN ORACLE NO SE PERMITE LA SENTENCIA ON UPDATE CASCADE para mantener la integridad referencial en caso de actualización de registros.

```
CREATE TABLE CONCESIONARIO (
    codigoConcesionario INTEGER NOT NULL,
    denominacion VARCHAR2(50) NOT NULL,
    telefono VARCHAR2(15),
    codigoPostal VARCHAR2(5) NOT NULL
);

CREATE TABLE VEHICULO (
    VIN CHAR(17) NOT NULL,
    matricula VARCHAR2(10) NOT NULL,
    marca VARCHAR2(30) NOT NULL,
```

```
    modelo VARCHAR2(30) NOT NULL,
    peso NUMBER(7,2) NOT NULL,
    concesionario INTEGER NOT NULL
);



ALTER TABLE CONCESIONARIO ADD CONSTRAINT pk_concec
PRIMARY KEY (codigoConcesionario);

ALTER TABLE VEHICULO ADD CONSTRAINT pk_vehic
PRIMARY KEY (VIN);

ALTER TABLE VEHICULO ADD CONSTRAINT fk_concec
FOREIGN KEY (concesionario) REFERENCES CONCESIONARIO(codigoConcesionario);

ALTER TABLE VEHICULO ADD CONSTRAINT uk_matric
UNIQUE (Matricula);

ALTER TABLE VEHICULO ADD CONSTRAINT ck_peso
CHECK (peso > 0);

ALTER TABLE VEHICULO ADD
ITV CHAR CONSTRAINT ck_ITV CHECK (ITV IN ('S', 'N'));

COMMENT ON TABLE CONCESIONARIO IS 'Almacenaje de distintos concesionarios';
COMMENT ON COLUMN CONCESIONARIO.codigoConcesionario IS 'Primary key de la tabla concesionario';
COMMENT ON COLUMN CONCESIONARIO.denominacion IS 'Nombre del concesionario';
COMMENT ON COLUMN CONCESIONARIO.telefono IS 'Teléfono del concesionario';
COMMENT ON COLUMN CONCESIONARIO.codigoPostal IS 'Código postal del concesionario';

COMMENT ON TABLE VEHICULO IS 'Almacenaje de vehículos y el concesionario al que pertenecen';
```

```
COMMENT ON COLUMN VEHICULO.VIN IS 'Número de bastidor del vehículo';

COMMENT ON COLUMN VEHICULO.matricula IS 'Matrícula del vehículo';

COMMENT ON COLUMN VEHICULO.marca IS 'Marca del vehículo';

COMMENT ON COLUMN VEHICULO.modelo IS 'Modelo del vehículo';

COMMENT ON COLUMN VEHICULO.peso IS 'Peso del vehículo';

COMMENT ON COLUMN VEHICULO.concesionario IS 'Foreign key hacia concesionario';


ALTER TABLE CONCESIONARIO DROP COLUMN telefono;
```

Actividad 3

Una multinacional ofrece formación en una plataforma online a su personal laboral y ha solicitado la creación de una base de datos para gestionar, almacenar y organizar toda la información dependiente en dicha plataforma de formación. De este modo, en la plataforma online se organizan cursos en el que únicamente participa el personal de la empresa, unos como estudiantes y otros como docentes. De cada curso se desea almacenar un identificador, el nombre, una breve descripción del mismo y el númerode horas de duración.

Un curso con el mismo identificador puede impartirse en diferentes fechas de inicio y con diferente alumnado y docente, es decir, posee diferentes ediciones. En una misma de inicio sólo puede impartirse una edición de un curso, que puede tener varios estudiantes pero únicamente un docente.

Del personal laboral se debe almacenar su código de empleado, el nombre y los apellidos, la dirección, teléfonos, fecha de nacimiento, el NIF, edad, codigo postal, provincia y el salario, así como si está capacitado para impartir cursos. El personal laboral puede participar en diferentes ediciones de un curso y, obviamente, en una edición participan varios miembros del personal laboral como estudiante almacenando en ese caso el correo electrónico de éste, que será único para cada estudiante.

El personal laboral puede ser docente en una edición de un curso y estudiante en otra, pero nunca puede ocupar ambos roles al mismo tiempo. Es decir, en una misma edición de un curso o bien lo imparte, o bien lo recibe.

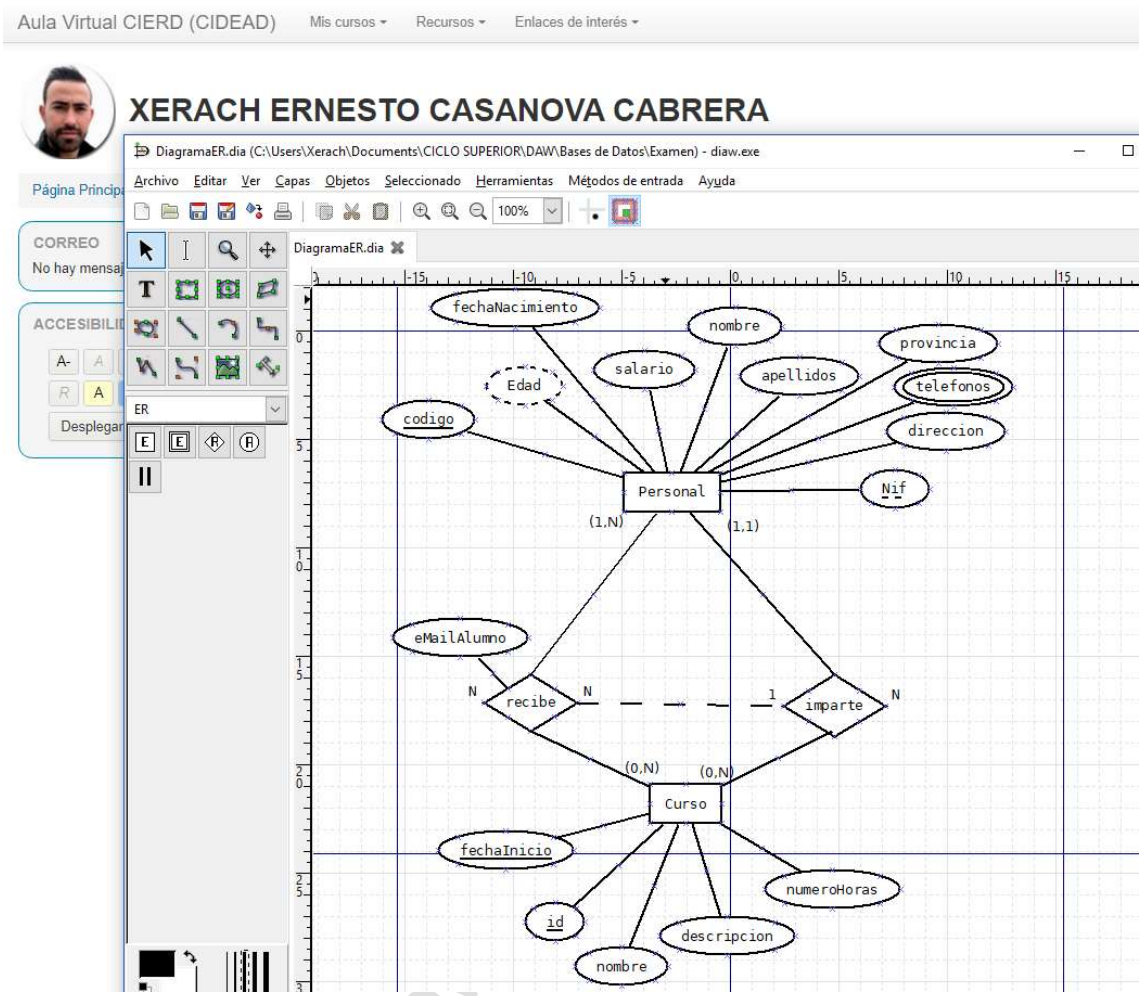
a. (0,25 puntos) Elabora un esquema empleando el modelo de datos Entidad-Relación Extendido empleando DIA u otra herramienta de elaboración de ERD (indicar cuál se ha utilizado). Ejemplo (Ejercicio 20).

b. (0,25 puntos) Añade, al ERD obtenido, una jerarquía total y exclusiva a tu elección. Además, añade un atributo derivado a la entidad Curso.

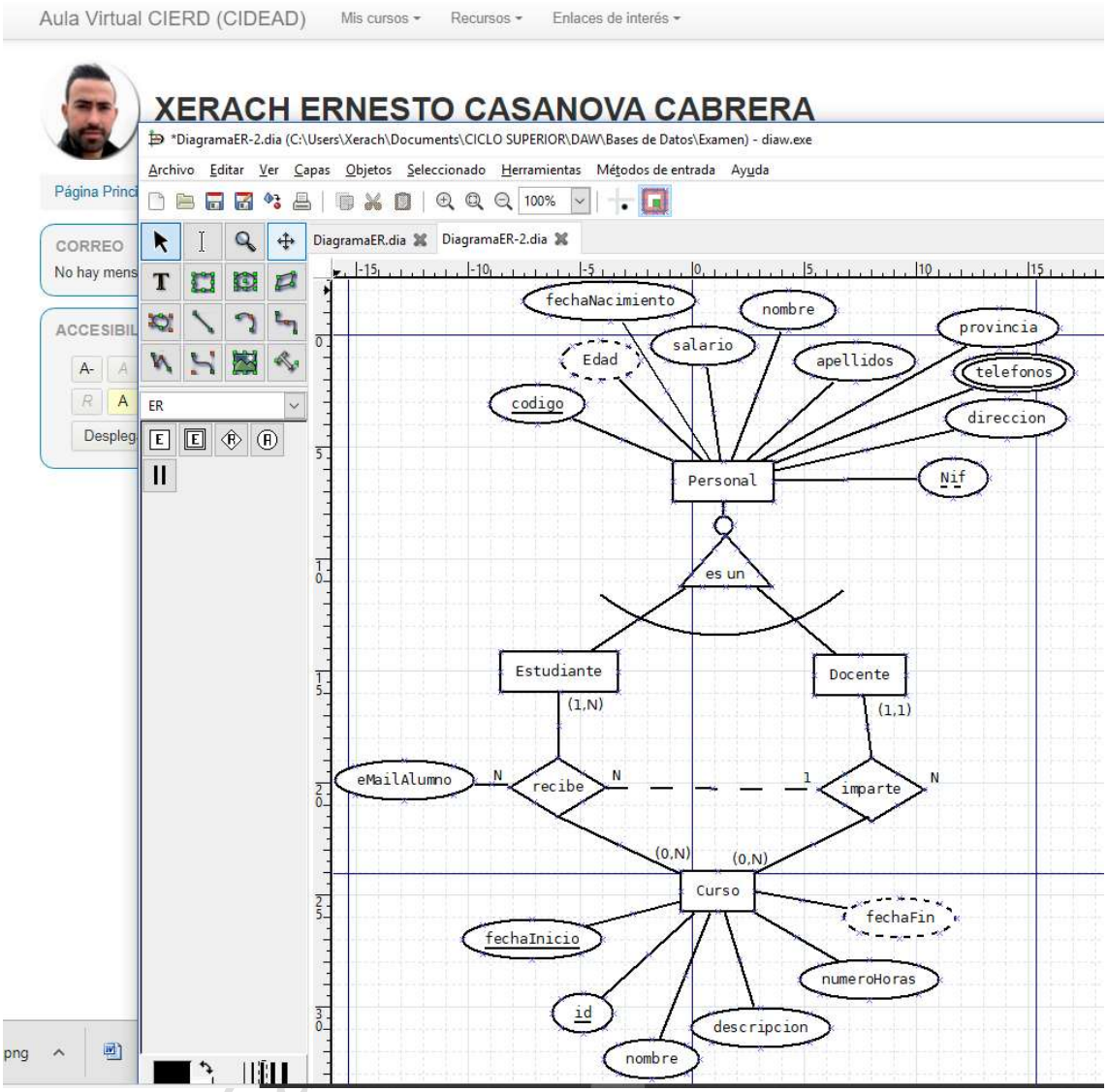
c. (0,5 puntos) Realiza el diseño lógico en el modelo relacional pasando a tablas el modelo conceptual obtenido, detallando las tablas, atributos, restricciones, claves principales y claves ajenas o foráneas. Ejemplo.

d. (0,5 puntos) Explica detalladamente, indicando las dependencias localizadas, el proceso de Normalización a las tablas obtenidas hasta la Tercera Forma Normal. Ejemplo.

Apartado a



Apartado b



Apartado c

PERSONAL (codigoPersonal, nombre, apellidos, nif*, telefonos, email*, direccion, codigoPostal, provincia, fechaNacimiento, edad, salario, esDocente)

PERSONAL_CURSO (eMailAlumno(FK), idCurso(FK), fInicio(FK))

CURSO (idCurso, fechaInicio, nombre, descripción, numeroHoras, codigoPersonal(FK))

Apartado d

Primera forma normal

Todos los campos deben ser atómicos. En el diseño lógico nos encontramos con que tanto **apellidos** como **telefonos,** no lo son.

- **Apellidos.** Se puede dividir el atributo en dos atributos distintos: apellido1 y apellido2º (apellido 2 puede aceptar valores nulos)
- **Telefonos.** Según la opción que se escoja, se puede permitir un número de campos teléfono determinado (por ejemplo dos: telefono1, telefono2) o Telefono se puede convertir en otra

entidad independiente que almacene los distintos números de teléfono de cada empleado, con una clave foránea en la entidad Personal que apunte hacia esa otra entidad.

PERSONAL (codigoPersonal, nombre, apellido1, apellido2, nif*, telefono(FK), email*, direccion, codigoPostal, provincia, fechaNacimiento, edad, salario, esDocente)

TELEFONO (telefono)

PERSONAL_CURSO (eMailAlumno(FK), idCurso(FK), fInicio(FK))

CURSO (idCurso, fechaInicio, nombre, descripción, numeroHoras, codigoPersonal(FK))

Segunda forma normal

En el caso de la segunda forma normal, debemos buscar atributos en las distintas entidades que no tengan dependencia funcional completa con la clave. Podemos así descartar las entidades que tienen como clave una clave simple y solo nos quedaría analizar la entidad CURSO, que es la única que tiene una clave compuesta.

Analizando la entidad CURSO, tanto nombre, como descripción, como numeroHoras tienen una dependencia funcional completa con la clave completa (IDCurso, fechaInicio), así que en este caso ya estaría todo el diseño lógico en 2ª forma normal.

Tercera forma normal

Debemos buscar en las distintas entidades, atributos que dependan transitivamente de la clave primaria. Nos encontramos con que la entidad PERSONAL podría tener varios casos.

- El atributo provincia depende transitivamente del atributo codigoPersonal a través de codigoPostal. Ya que a partir del código postal podemos averiguar la Provincia y a su vez, con el códigoPostal no podemos averiguar el codigoPersonal. En este caso, podríamos agregar una nueva entidad llamada Provincia.

- Otro atributo que depende tiene dependencia transitiva es edad, ya que a través del atributo fechaNacimiento podemos sacarlo. En este caso no haría falta crear otra tabla, simplemente bastaría con eliminar el campo, ya que a través del propio campo fechaNacimiento se puede calcular.

PERSONAL (codigoPersonal, nombre, apellido1, apellido2, nif*, telefono(FK), email*, direccion, codigoPostal, idProvincia, fechaNacimiento, edad, salario, esDocente)

TELEFONO (telefono)

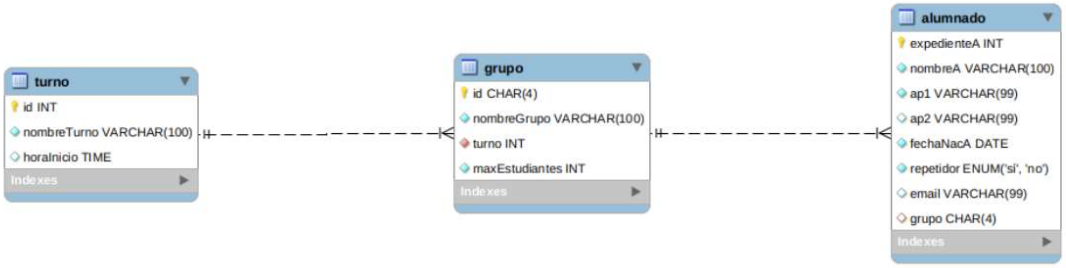
PROVINCIA (provincia)

PERSONAL_CURSO (eMailAlumno(FK), idCurso(FK), fInicio(FK))

CURSO (idCurso, fechaInicio, nombre, descripción, numeroHoras, codigoPersonal(FK))

Actividad 4

Se tiene una base de datos para la gestión del alumnado, los grupos a los que pertenecen y el turno que tienen estos grupos. De este modo, el alumnado se identifica por un expediente y tiene como característica un nombre, apellidos, fecha de nacimiento, si es o no repetidor, email y un único grupo al que pertenecen. Estos grupos poseen un id de cuatro caracteres (por ejemplo, 1DAW), un nombre un cupo máximo de estudiantes y pertenecen a un turno. Los turnos se identifican con un número, poseen un nombre y una hora de inicio.



Conocida la base de datos, elabora un script denominado actividad4_tudni.sql Realiza las siguientes consultas:

a. (0,25 puntos) Muestra un listado que muestre el nombre completo de todos el alumnado que no tenga segundo apellido, es decir, cuyo segundo apellido sea igual a un valor nulo.

b. (0,25 puntos) Muestra un listado que incluya el nombre de todos los grupos y el nombre del turno al que le corresponde de los grupos que superen un máximo de 20 estudiantes y tengan tu misma fecha de nacimiento.

c. (0,5 puntos) Mostrar el nombre del grupo que tiene mayor alumnado matriculado.

d. (0,5 puntos) Muestra el nombre de los grupos que tienen el mayor cupo máximo de estudiantes.

```
SELECT NOMBREA || ' ' ||  AP1 || ' ' || ' ' || AP2 AS "NOMBRE COMPLETO"

FROM ALUMNADO WHERE AP2 IS NULL;


SELECT DISTINCT G.NOMBREGRUPO, T.NOMBRETURNO

FROM GRUPO G INNER JOIN TURNO T ON G.TURNO = T.ID

INNER JOIN ALUMNADO A ON G.ID = A.GRUPO

WHERE G.MAXESTUDIANTES > 20 AND A.FECHANACA = '21/10/84';


SELECT G.NOMBREGRUPO AS "GRUPO CON MAYOR NÚMERO DE ALUMNOS"

FROM GRUPO G INNER JOIN ALUMNADO A ON G.ID = A.GRUPO

GROUP BY G.NOMBREGRUPO

HAVING COUNT(GRUPO) = (SELECT MAX(TOTAL)

                       FROM (SELECT COUNT(GRUPO) AS TOTAL

                       FROM ALUMNADO
```

```
                GROUP BY GRUPO));

SELECT NOMBREGRUPO
FROM GRUPO
WHERE MAXESTUDIANTES IN (SELECT MAX(MAXESTUDIANTES) FROM GRUPO);
```

Actividad 5

En base a la base de datos de la Actividad 4, elabora un script denominado actividad5_tudni.sql, que tenga los siguientes apartados:

a. (0,25 puntos) Inserta al menos tres valores en cada una de las tablas.

b. (0,5 puntos) En la tabla grupo, actualiza el campo maxEstudiantes del registro de un único grupo (el que tu quieras), asignándole el valor correspondiente al número total de estudiantes que hay en la tabla alumnado y que tienen asignado ese mismo grupo.

c. (0,25 puntos) Se tiene una tabla con que almacena el alumnado recién inscrito, denominada alumnadoReciente, que tiene como columnas las mismas que la tabla alumnado, excepto grupo. Inserta los registros de la tabla alumadoReciente en la tabla alumnado asociandoles un grupo (el que tu prefieras) en una única sentencia

```
INSERT INTO TURNO VALUES
(1, 'MAÑANA', '8:00');

INSERT INTO TURNO VALUES
(2, 'TARDE', '14:00');

INSERT INTO TURNO VALUES
(3, 'NOCHE', '19:00');

INSERT INTO GRUPO VALUES
('DAWA', 'Desarrollo de Aplicaciones Web', 1, 22);

INSERT INTO GRUPO VALUES
('DAWB', 'Desarrollo de Aplicaciones Multiplataforma', 2, 15);

INSERT INTO GRUPO VALUES
('ASIR', 'Administración de Sistemas Informáticos', 3, 25);

INSERT INTO ALUMNADO VALUES
(1, 'XERACH', 'CASANOVA', 'CABRERA', '21/10/84', 'si', 'xerach@xerach.com', 'DAWA');

INSERT INTO ALUMNADO VALUES
(2, 'JUAN', 'PÉREZ', 'RODRÍGUEZ', '20/11/88', 'no', 'juan@juan.com', 'DAWA');

INSERT INTO ALUMNADO (expedienteA, nombreA, ap1, fechaNacA, repetidor, email, grupo) VALUES
(3, 'MARCOS', 'GONZÁLEZ', '03/02/96', 'no', 'marcos@marcos.com', 'DAWA');
```

```
UPDATE GRUPO SET
MAXESTUDIANTES = (SELECT COUNT(EXPEDIENTEA) FROM ALUMNADO )
WHERE ID = 'DAWB';

INSERT INTO ALUMNADO (expedienteA, nombreA, ap1, fechaNacA, repetidor, email, grupo)
SELECT expedienteA, nombreA, ap1, fechaNacA, repetidor, email, 'DAWA' FROM ALUMNADORECIENTE;
```

Actividad 6

¿Qué diferencias existen entre una función y un disparador? Explica, con un ejemplo propio, para qué utilizarías cada uno de ellos.

Una de diferencia entre una función o un procedimiento almacenado y un disparador es que el disparador se ejecuta automáticamente cuando se ejecutan sentencias de UPDATE, INSERT O DELETE.

Los disparadores no devuelven resultados a partir de consultas, en cambio las funciones y procedimientos si lo hacen.

Otra de las características de las funciones que no tienen los disparadores es que pueden recibir parámetros de entrada y salida.

Ejemplo de procedimiento en las tablas del ejercicio 5. Es un procedimiento que cuando se ejecuta, se le pasan dos parámetros: el grupo de origen y el grupo de destino. Al hacerlo, todos los alumnos que sean del grupo de origen se actualizarán con el grupo destino.

```
CREATE OR REPLACE PROCEDURE AlumnadoGrupo(
                        id_grupoOrigen grupo.id%TYPE,
                        id_grupoDestino grupo.id%TYPE)
AS

    vGrupoOrigen grupo.id%TYPE;
    vGrupoDestino grupo.id%TYPE;

BEGIN


    SELECT id into vGrupoOrigen FROM grupo WHERE id = id_grupoOrigen;
    SELECT id into vGrupoDestino FROM grupo WHERE id = id_grupoOrigen;

    UPDATE alumnado SET grupo = vGrupoDestino WHERE grupo = vGrupoOrigen;
    DBMS_OUTPUT.PUT_LINE('Se han trasladado todos los alumnos del grupo' || vGrupoOrigen || ' al
grupo ' || vGrupoDestino);


    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Grupos no válidos');
END;
/

DECLARE

    p_id_grupoOrigen grupo.id%TYPE := &grupoOrigen;
    p_id_grupoDestino grupo.id%TYPE := &grupoDestino;
```

```
BEGIN


   AlumnadoGrupo(p_id_grupoOrigen, p_id_grupoDestino);

END;
/
```

Además para probar, he añadido un bloque de código:

```
DECLARE

   p_id_grupoOrigen grupo.id%TYPE := &grupoOrigen;
   p_id_grupoDestino grupo.id%TYPE := &grupoDestino;

BEGIN

   AlumnadoGrupo(p_id_grupoOrigen, p_id_grupoDestino);

END;
/
```

Ejemplo de disparador en las tablas del ejercicio 5. Cuando se inserta o se actualiza un grupo, no se permite que el turno sea distinto a 1, 2 o 3. Tampoco se permite que el máximo de usuarios permitidos sea menor que 1

```
CREATE OR REPLACE TRIGGER integridad_grupo
BEFORE INSERT OR UPDATE ON grupo
FOR EACH ROW

BEGIN

  IF (:NEW.maxestudiantes < 1) THEN
      RAISE_APPLICATION_ERROR(-20205, 'No pueden haber cursos con menos de un estudiante máximo permitido.');
   END IF;
   IF :NEW.turno < 1 OR :NEW.turno  > 3 OR :NEW.turno IS NULL THEN
      RAISE_APPLICATION_ERROR(-20206, 'El turno debe tener valor 1, 2 o 3');
   END IF;

END;
/
```

Actividad 7

Crea, en un script denominado actividad7_tudni.sql, un objeto relacionado con un curso, con los atributos que estimes oportuno (al menos tres).

```
CREATE OR REPLACE TYPE alumno AS OBJECT(
   expediente INTEGER,
   nombre VARCHAR2(100),
   apellidos VARCHAR2(99),
   fechaNac DATE,
   repetidor VARCHAR(2),
   email VARCHAR2(99),

   CONSTRUCTOR FUNCTION alumno(expediente INTEGER,
                        nombre VARCHAR2,
                        ap1 VARCHAR2,
                        ap2 VARCHAR2,
                        fechaNac DATE,
                        repetidor VARCHAR2,
                        email VARCHAR2)
                        RETURN SELF AS RESULT
);
/

CREATE OR REPLACE TYPE BODY alumno AS
   CONSTRUCTOR FUNCTION alumno(expediente INTEGER,
                        nombre VARCHAR2,
                        ap1 VARCHAR2,
                        ap2 VARCHAR2,
                        fechaNac DATE,
                        repetidor VARCHAR2,
                        email VARCHAR2)
                        RETURN SELF AS RESULT
   IS
      BEGIN

         SELF.expediente := expediente;
         SELF.nombre := nombre;
         SELF.apellidos := ap1 || ' ' || ap2;
         SELF.fechaNac := fechaNac;
         SELF.repetidor := repetidor;
```

```
        SELF.email := email;

        RETURN;
    END;
END;
/
```