Análisis, Diseño e Implementación de Base de Datos de una Almazara

Trabajo de Base de Datos Avanzadas

Ingeniería Informática - Especialidad de Computación

Escuela Politécnica Superior de Córdoba

Universidad de Córdoba

Curso académico: 2023-2024

Alumno:

Manuel Casas Castro -31875931R - i72cascm@uco.es

Profesores:

Gonzalo Cerruela Garcia

Manuel Mendoza Hurtado

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

Este documento recoge el análisis, diseño e implementación de la base de datos de una almazara a modo de proyecto final para la asignatura Base de Datos Avanzadas.

El proyecto se divide en una serie de fases que serán indicadas y desarrolladas en esta memoria. De manera adicional, acompañando las explicaciones, se podrá observar una serie de capturas de pantalla tanto del código fuente utilizado en las diferentes fases como de su funcionamiento.

Para la creación de la base de datos y su manipulación en las diferentes fases del proyecto, se utilizará SQL Developer y la cuenta Oracle proporcionada por la UCO a sus estudiantes.

DESCRIPCIÓN DEL PROBLEMA

A continuación se realizará una descripción detallada del problema junto con sus requisitos y reglas de negocio. Esta descripción es la que en una primera instancia será utilizada para la fase 1 del proyecto, pero podrá ser susceptible a cambios si las siguientes fases lo requieren:

Enunciado:

El encargado de una almazara requiere llevar un registro de los agricultores que llevan sus cosechas de aceitunas, de los diversos trabajadores que supervisan el correcto funcionamiento de la maquinaria, las entregas realizadas por los agricultores y la producción obtenida en cada lote obtenido.

Descripción detallada:

1.- Se reciben cosechas de diversos agricultores. De cada agricultor se almacena: DNI, nombre, apellidos, teléfono, correo, dirección y empresa.

2.- En cada entrega, se registra la fecha de recepción, peso de las aceitunas, su calidad y tipos de aceituna.

3.- Tras el funcionamiento de la almazara, se registra la cantidad de aceite producido, fecha, rendimiento y tipo de aceite.

4.- El funcionamiento de la almazara la lleva un equipo de trabajo.

5.- Registro del equipo de empleados, cada empleado tiene: Nombre, apellidos, DNI, teléfono, dirección, correo y puesto.

6.- Los agricultores pueden consultar información sobre sus entregas de aceitunas y producción.

Supuestos:

1.- Agricultores y trabajadores con DNI único.

2.- En agricultores, los atributos dni, nombre completo, fecha de nacimiento, dirección y empresa no pueden ser nulos.

3.- En trabajadores, los atributos dni, nombre completo, fecha de nacimiento, dirección y puesto no pueden ser nulos.

4.- Las aceitunas se clasifican en los tipos: picual, hojiblanca, royal y cornicabra.

5.- La calidad de las aceitunas es evaluada en una escala numérica entre 0 y 10 (0 = baja calidad, 10 = alta calidad).

6.- Los tipos de aceite son: Virgen, virgen extra y orujo.

7.- Los puestos de trabajo son: Recepción, limpieza, batidora y almacenamiento.

8.- Cada trabajador esta asignado a un único puesto de trabajo.

9.- En cada etapa de la almazara debe existir como mínimo un trabajador asignado.

10.- La edad de agricultores y trabajadores debe ser mayor o igual a 18 años.

11.- Para realizar una entrega, esta debe pesar más de 500 kg.

12.- Un agricultor no puede realizar más de 3 entregas al año.

13.- Los agricultores tienen acceso a la información de sus entregas.

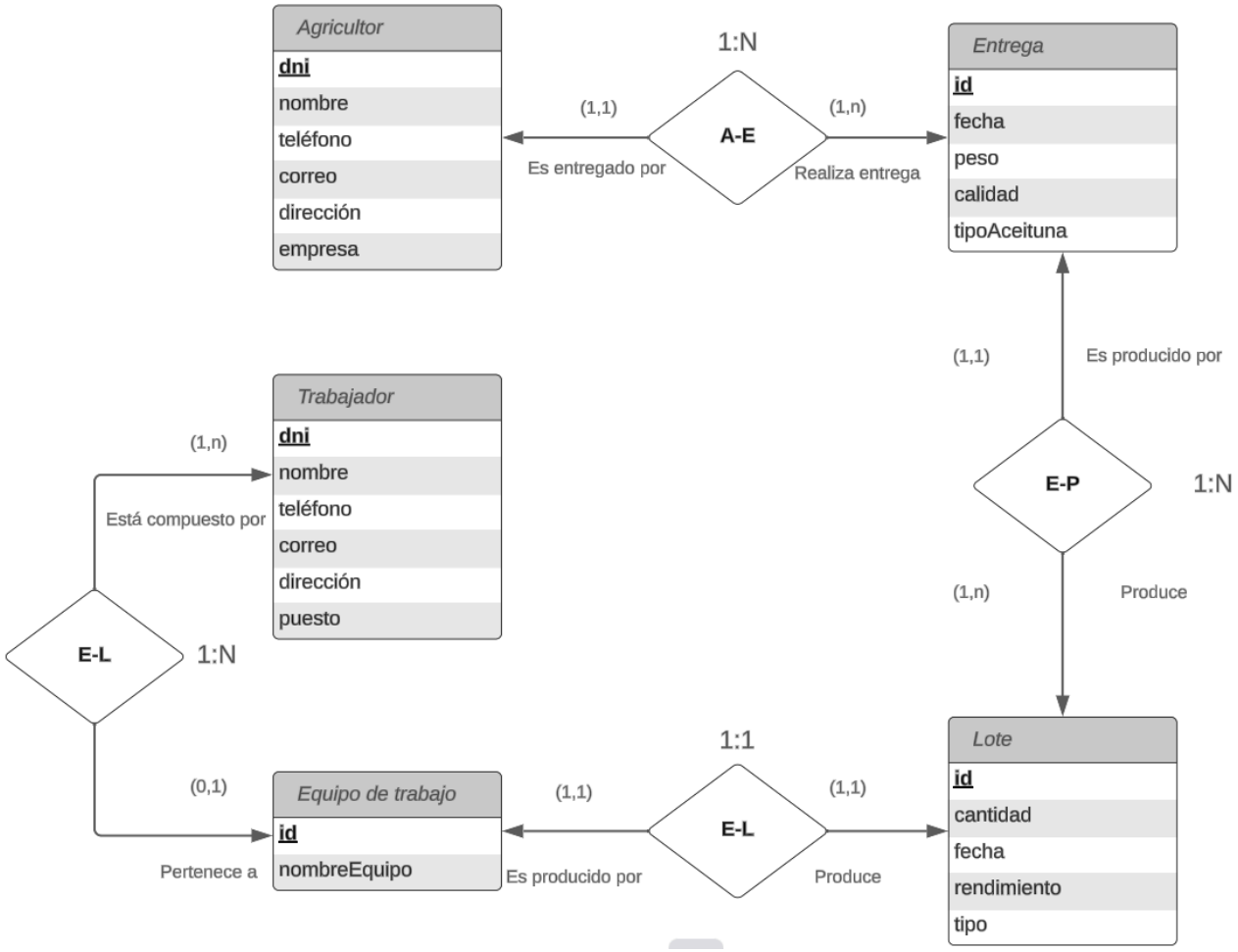
14.- En los fines de semana (sábados y domingos en la fecha del sistema) no se pueden realizar registros en las bases de datos de entrega y producción.

FASE 1

Una vez definido con detalle el problema que va a ser tratado en este proyecto, es momento de comenzar a elaborar el esquema de la base de datos

MODELO CONCEPTUAL

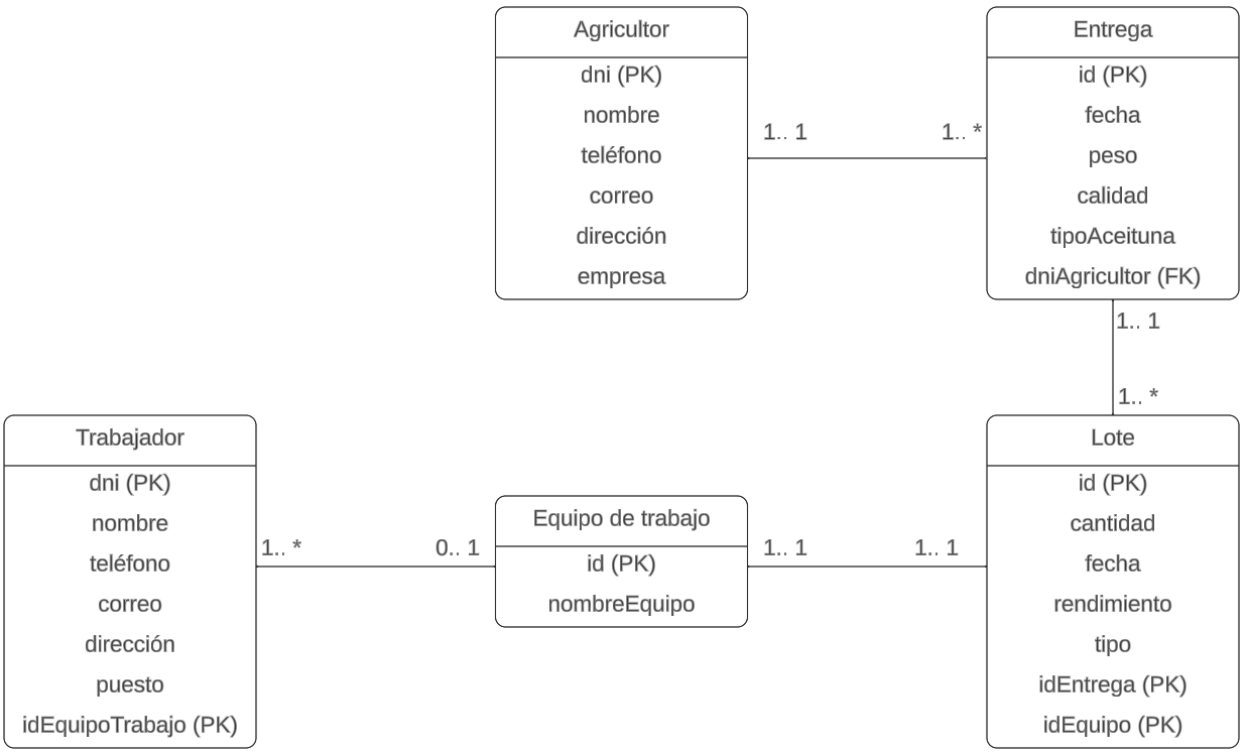
A continuación, se presenta el modelo entidad-relación de este problema:



Se puede observar que el modelo tiene en total cinco entidades y cuatro interrelaciones que las unen.

MODELO RELACIONAL

Se muestra ahora el modelo relacional de esta base de datos, el cual ha sido elaborado a partir del modelo entidad-relación:



Agricultor(<u>dni</u>, nombre, teléfono, correo, dirección, empresa)

Entrega(<u>id</u>, fecha, peso, calidad, tipoAceituna, **dniAgricultor**)

Lote(<u>id</u>, cantidad, fecha, rendimiento, tipo, **idEntrega**, **idEquipo**)

EquipoTrabajo(<u>id</u>, nombreEquipo)

Trabajador(<u>dni</u>, nombre, teléfono, correo, dirección puesto, **idEquipoTrabajo**)

Con estos dos modelos ya creados, se puede proceder a la creación y validación de la base de datos mediante la herramienta **SQL Developer** de **Oracle** mediante las cuentas dadas por la uco a los alumnos de esta asignatura.

SCRIPT DE CREACIÓN DE LA BASE DE DATOS

En los archivos que han sido entregados en este trabajo, se encuentra uno nombrado "**scriptCreacion.sql**" el cual contiene todo el código necesario para generar las tablas y algunas de sus restricciones (el resto son codificadas en la fase 2 mediante triggers) atendiendo a los descrito en el enunciado del problema y sus supuestos.

A continuación, se muestran capturas de pantalla de la creación de las tablas en **SQL Developer**.

- Tablas agricultor y entrega:

```
-- Tabla agricultor --
create table agricultor
    (dni number(8) primary key,
    nombreCompleto varchar2(64) not null,
    fechaNacimiento DATE not null,
    telefono number(14),
    correo varchar2(32),
    direccion varchar2(64) not null,
    empresa varchar2(64) not null
);

-- Tabla entrega --
create table entrega (
    idEntrega number(10) primary key,
    fecha date not null,
    peso number(10, 2) not null,
    calidad number(5),
    tipoAceituna varchar2(50) not null,
    dniAgricultor number(8),
    constraint fk_entrega_agricultor foreign key (dniAgricultor) references agricultor(dni),
    constraint ck_tipoAceituna CHECK (tipoAceituna IN ('Picual','Hojiblanca','Royal','Cornicabra')),
    constraint ck_calidad CHECK (calidad BETWEEN 0 AND 10),
    CONSTRAINT ck_peso CHECK (peso >= 500)
);
```

- Tablas equipoTrabajo, produccionAceite y trabajador:

```
-- Tabla equipoTrabajo --
create table equipoTrabajo (
    idEquipo number(10) primary key,
    nombreEquipo varchar2(100) not null
);

-- Tabla produccionAceite --
create table produccionAceite (
    idProduccion number(10) primary key,
    litros number(10, 2) not null,
    fecha date not null,
    rendimiento number(5, 2),
    tipoAceite varchar2(50) not null,
    idEntrega number(10),
    idEquipo number(10),
    constraint fk_produccion_entrega foreign key (idEntrega) references entrega(idEntrega),
    constraint fk_produccion_equipo foreign key (idEquipo) references equipoTrabajo(idEquipo),
    constraint ck_tipoAceite CHECK (tipoAceite IN ('Virgen','Virgen Extra','Orujo'))
);

-- Tabla trabajador --
create table trabajador
    (dni number(8) primary key,
    nombreCompleto VARCHAR2(64) not null,
    fechaNacimiento DATE not null,
    telefono number(14),
    correo varchar2(32),
    direccion varchar2(64) not null,
    puestoTrabajo varchar2(64) not null,
    idEquipoTrabajo number(10),
    constraint fk_trabajador_equipo foreign key (idEquipoTrabajo) references equipoTrabajo(idEquipo),
    constraint ck_puesto CHECK (puestoTrabajo IN ('Recepción','Limpieza','Batidora','Almacenamiento'))
);
```

Se pueden observar las restricciones codificadas para que algunos de los atributos sean no nulos, la creación de claves foráneas para establecer las relaciones entre las tablas y algunas restricciones de dominio que fueron mencionadas en el enunciado, como pueden ser los tipos de aceituna y aceite, los valores que puede tomar la calidad del aceite o la restricción de que una entrega sea de mínimo 500 kilogramos o más.

Como último paso de esta primera fase, se procede a realizar una carga de datos suficiente en las distintas tablas creadas para validar su funcionamiento y así probar que sus restricciones funcionan correctamente. Esta carga de datos se encuentra en el propio script de creación "**scriptCreacion.sql**" y aquí se muestran alguno de las inserciones a modo de ejemplo:

```
-- Valores agricultor --
insert into agricultor values (12345678, 'Juan Pérez', TO_DATE('01-01-2001', 'DD-MM-YYYY'), 987654321, 'juan.p
insert into agricultor values (23456789, 'Ana Gómez', TO_DATE('15-03-2002', 'DD-MM-YYYY'), 987654322, 'ana.gom
insert into agricultor values (34567890, 'Carlos López', TO_DATE('23-05-1980', 'DD-MM-YYYY'), 987654323, 'carl
insert into agricultor values (45678901, 'Laura Martínez', TO_DATE('07-07-1995', 'DD-MM-YYYY'), 987654324, 'la
```

```
-- Valores entrega --
insert into entrega values (1, TO_DATE('15-01-2023', 'DD-MM-YYYY'), 500.50, 8, 'Picual', 12345678);
insert into entrega values (2, TO_DATE('16-01-2023', 'DD-MM-YYYY'), 550.00, 7, 'Hojiblanca', 23456789);
insert into entrega values (3, TO_DATE('17-01-2023', 'DD-MM-YYYY'), 600.75, 9, 'Cornicabra', 34567890);
```

```
-- Valores equipoTrabajo --
insert into equipoTrabajo values (1, 'Equipo Alfa');
insert into equipoTrabajo values (2, 'Equipo Bravo');
insert into equipoTrabajo values (3, 'Equipo Charlie');
```

```
-- Valores produccionAceite --
insert into produccionAceite values (1, 1000.00, TO_DATE('15-02-2023', 'DD-MM-YYYY'), 0.8, 'Virgen Extra', 1, 1);
insert into produccionAceite values (2, 950.50, TO_DATE('16-02-2023', 'DD-MM-YYYY'), 0.75, 'Virgen', 2, 2);
insert into produccionAceite values (3, 1100.75, TO_DATE('17-02-2023', 'DD-MM-YYYY'), 0.82, 'Orujo', 3, 3);
```

```
-- Valores trabajadores --
-- Equipo 1
insert into trabajador values (12345678, 'Carlos González', TO_DATE('02-03-1985', 'DD-MM-YYYY'), 987654321, 'carlos.go
insert into trabajador values (23456789, 'Marta Fernández', TO_DATE('15-05-1980', 'DD-MM-YYYY'), 987654322, 'marta.fer
```

	DNI	NOMBRECOMPLETO	FECHANACIMIENTO	TELEFONO	CORREO	DIRECCION	EMPRESA
1	12345678	Juan Pérez	01/01/01	987654321	juan.perez@exampl...	Calle Falsa 123	Aceitunas Pérez S.L.
2	23456789	Ana Gómez	15/03/02	987654322	ana.gomez@example...	Avenida Olivos 45	Olivos Gómez S.A.
3	34567890	Carlos López	23/05/80	987654323	carlos.lopez@exam...	Plaza del Aceite 9	López y Aceites

	IDENTREGA	FECHA	PESO	CALIDAD	TIPOACEITUNA	DNIAGRICULTOR
1	1	15/01/23	500,5	8	Picual	12345678
2	2	16/01/23	550	7	Hojiblanca	23456789
3	3	17/01/23	600,75	9	Cornicabra	34567890

	IDEQUIPO	NOMBREEQUIPO
1	1	Equipo Alfa
2	2	Equipo Bravo
3	3	Equipo Charlie

FASE 2

Ahora que la base de datos ha sido creada exitosamente junto a algunas de las restricciones declaradas en el enunciado y con una carga de datos suficiente para hacer pruebas, se procede a la codificación y prueba de una serie de disparadores que generen nuevas restricciones o sustituyan algunas de las existentes creadas en la fase anterior.

Los disparadores se encuentran codificados en el script "**triggers.sql**" y la inserción y manipulación de datos usados para probar los diferentes disparadores se encuentran en "**pruebasTriggers.sql**".

DISPARADOR DE AUDITORÍA

Se han codificado dos triggers diferentes, el primero de ellos notifica cuando en la tabla de entregas se ha modificado los atributos fecha o dni de alguna de las tuplas:

```
-- Trigger que notifica cuando en la tabla entrega se han modificado los atributos fecha o dniAgricultor de alguna tupla
CREATE OR REPLACE TRIGGER auditoria_entrega
AFTER UPDATE OF fecha, dniAgricultor ON entrega
FOR EACH ROW
BEGIN
    IF :NEW.fecha != :OLD.fecha OR :NEW.dniAgricultor != :OLD.dniAgricultor THEN
        DBMS_OUTPUT.PUT_LINE('Se ha modificado la entrega con ID ' || :OLD.idEntrega ||
                             '. Cambios: Fecha de ' || :OLD.fecha || ' a ' || :NEW.fecha ||
                             ', DNI Agricultor de ' || :OLD.dniAgricultor || ' a ' || :NEW.dniAgricultor);
    END IF;
END;
/
```

```
-- Update 1
UPDATE entrega
SET fecha = TO_DATE('30-01-2050', 'DD-MM-YYYY')
WHERE idEntrega = 14;
-- Update 2
UPDATE entrega
SET fecha = TO_DATE('25-02-2030', 'DD-MM-YYYY')
WHERE idEntrega = 14;
```

```
Se ha modificado la entrega con ID 14. Cambios: Fecha de 25/02/30 a 30/01/50, DNI Agricultor de 56789013 a 56789013

1 fila actualizadas.

Se ha modificado la entrega con ID 14. Cambios: Fecha de 30/01/50 a 25/02/30, DNI Agricultor de 56789013 a 56789013

1 fila actualizadas.
```

El segundo trigger se encarga de notificar si ha sido modificado cualquier tupla de la tabla de trabajadores, independientemente del atributo que haya sido modificado:

```
-- Triger que notifica si ha sido modificado cualquier valor de la tabla trabajador
CREATE OR REPLACE TRIGGER auditoria_trabajador
AFTER UPDATE ON trabajador
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('El trabajador con DNI ' || :OLD.dni || ' ha sido actualizado con éxito.');
END;
/
```

```
-- Updates para comprobar funcionalidad trabajadores
-- Update 1
UPDATE trabajador
SET correo = 'pablo.aguire_new@example.com',
    telefono = 987655555
WHERE dni = 45678902;
-- Update 2
UPDATE trabajador
SET puestoTrabajo = 'Limpieza',
    direccion = 'Avenida Principal'
WHERE dni = 56789013;
-- Update 3
UPDATE trabajador
SET direccion = 'Calle Secunda 78',
    nombreCompleto = 'Óscar Sánchez Gómez'
WHERE dni = 67890124;
```

```
El trabajador con DNI 45678902 ha sido actualizado con éxito.


1 fila actualizadas.

El trabajador con DNI 56789013 ha sido actualizado con éxito.


1 fila actualizadas.

El trabajador con DNI 67890124 ha sido actualizado con éxito.


1 fila actualizadas.
```

DISPARADOR DE SEGURIDAD SEGÚN FECHA

Se tratan de dos disparadores que impiden realizar inserciones de datos en las tablas de entrega y lotes (producción) según la fecha actual del sistema. En este caso, impide que se hagan inserciones en los fines de semana (sábados y domingos):

```
-- Trigger que impide insertar datos en la tabla de entregas en sábados y domingos
CREATE OR REPLACE TRIGGER fechas_laborales_entrega
BEFORE INSERT ON entrega
BEGIN
    IF(TO_CHAR(SYSDATE, 'DY') IN ('SÁB', 'DOM'))
        THEN RAISE_APPLICATION_ERROR(-20500, 'ERROR: No se encuentra en un día laboral.');
    END IF;
END;
/

-- Trigger que impide insertar datos en la tabla de produccion en sábados y domingos
CREATE OR REPLACE TRIGGER fechas_laborales_produccion
BEFORE INSERT ON produccionaceite
BEGIN
    IF(TO_CHAR(SYSDATE, 'DY') IN ('SÁB', 'DOM'))
        THEN RAISE_APPLICATION_ERROR(-20500, 'ERROR: No se encuentra en un día laboral.');
    END IF;
END;
/
```

```
-- Ejercicio 2 --
-- Insert para comprobar que no se insertan datos en entrega los sabados y domingos (es importante que la fech
insert into entrega values (16, TO_DATE('28-01-2024', 'DD-MM-YYYY'), 49.00, 7, 'Royal', 56789013);
insert into entrega values (17, TO_DATE('29-01-2025', 'DD-MM-YYYY'), 50.50, 8, 'Hojiblanca', 67890124);
-- Insert para comprobar que no se insertan datos en produccion los sabados y domingos (es importante que la f
insert into produccionAceite values (16, 995.00, TO_DATE('28-02-2023', 'DD-MM-YYYY'), 0.78, 'Orujo', 14, 3);
insert into produccionAceite values (17, 1025.50, TO_DATE('01-03-2023', 'DD-MM-YYYY'), 0.8, 'Virgen', 15, 2);
```

```
Error que empieza en la línea: 49 del comando -
insert into produccionAceite values (16, 995.00, TO_DATE('28-02-2023', 'DD-MM-YYYY'), 0.78, 'Orujo', 14, 3)
Error en la línea de comandos : 49 Columna : 13
Informe de error -
Error SQL: ORA-20500: ERROR: No se encuentra en un día laboral.
ORA-06512: en "I72CASCM.FECHAS_LABORALES_PRODUCCION", línea 3
ORA-04088: error durante la ejecución del disparador 'I72CASCM.FECHAS_LABORALES_PRODUCCION'
```

DISPARADOR DE RESTRICCIÓN DE DOMINIO

Se trata de una serie de disparadores que completan algunas de las restricciones y lógicas de negocio descritas en el enunciado y que no fueron añadidas en la primera fase del proyecto.

Los dos primeros disparadores se encargan de calcular la edad que tiene el agricultor o trabajador introducido/modificado e impedir que la acción se realice si este es menor de edad:

```
-- Trigger para comprobar que el trabajador/a es mayor de edad
CREATE OR REPLACE TRIGGER trabajadores_mayor_edad
BEFORE INSERT OR UPDATE ON trabajador
FOR EACH ROW
BEGIN
    IF :NEW.fechaNacimiento >  add_months(sysdate , -12*18)  then
       RAISE_APPLICATION_ERROR(-20001, 'El trabajador/a no puede ser menor de edad.');
    END IF;
END;
/

-- Trigger para comprobar que el agricultor/a es mayor de edad
CREATE OR REPLACE TRIGGER agricultores_mayor_edad
BEFORE INSERT OR UPDATE ON agricultor
FOR EACH ROW
BEGIN
    IF :NEW.fechaNacimiento >  add_months(sysdate , -12*18)  then
       RAISE_APPLICATION_ERROR(-20001, 'El agricultor/a no puede ser menor de edad.');
    END IF;
END;
/
```

```
-- Modificaro añadir algun agricultor/a con edad menor a 18 años
UPDATE agricultor
SET fechaNacimiento = TO_DATE('30-04-2009', 'DD-MM-YYYY')
WHERE dni = 56789012;
UPDATE agricultor
SET fechaNacimiento = TO_DATE('30-12-2001', 'DD-MM-YYYY')
WHERE dni = 56789012;
insert into agricultor values (45644444, 'Roberto Navarro', TO_DATE('08-08-2030', 'DD-MM-YYYY'), 987
insert into agricultor values (56789044, 'Luisa Moreno', TO_DATE('14-02-1992', 'DD-MM-YYYY'), 987654
```

```
Error que empieza en la línea: 74 del comando -
insert into agricultor values (45644444, 'Roberto Navarro', TO_DATE('08-08-2030', 'DD-MM-YYYY'), 987654333, 'rober
Error en la línea de comandos : 74 Columna : 13
Informe de error -
Error SQL: ORA-20001: El agricultor/a no puede ser menor de edad.
ORA-06512: en "I72CASCM.AGRICULTORES_MAYOR_EDAD", línea 3
ORA-04088: error durante la ejecución del disparador 'I72CASCM.AGRICULTORES_MAYOR_EDAD'
```

El siguiente disparador de restricción hace que cualquier inserción o modificación de una de las entregas que tengan un peso menor a 500 kilogramos, sea modificada por el disparador para que en la base de datos conste con un peso de 500 kilogramos (aunque el peso real sea menor):

```
-- Trigger para hacer que el peso mínimo de entrega sea de 500kg mínimo
CREATE OR REPLACE TRIGGER cambiar_peso_entrega
BEFORE INSERT OR UPDATE ON entrega
FOR EACH ROW
BEGIN
    IF :NEW.peso < 500 THEN
        :NEW.peso := 500;
    END IF;
END;
/
```

```
insert into entrega values (16, TO_DATE('22-01-2023', 'DD-MM-YYYY'), 200.00, 7, 'Royal', 56789013);
insert into entrega values (17, TO_DATE('24-01-2023', 'DD-MM-YYYY'), 430.50, 8, 'Hojiblanca', 67890124);
UPDATE entrega
SET peso = 0
WHERE idEntrega = 9;
UPDATE entrega
SET peso = 499.99
WHERE idEntrega = 14;
```

	IDENTREGA	FECHA	PESO	CALIDAD	TIPOACEITUNA	DNIAGRICULTOR
1	1	15/01/23	500,5	8	Picual	12345678
2	2	16/01/23	550	7	Hojiblanca	23456789
3	3	17/01/23	600,75	9	Cornicabra	34567890
4	4	18/01/23	530,2	6	Royal	45678901
5	5	19/01/23	770	7	Hojiblanca	56789012
6	6	20/01/23	560,5	8	Royal	67890123
7	7	21/01/23	890,3	5	Picual	78901234
8	8	22/01/23	510	7	Cornicabra	89012345
9	9	23/01/23	500	8	Picual	90123456
10	10	24/01/23	500	6	Royal	12345679
11	11	25/01/23	680,5	7	Hojiblanca	23456780
12	12	26/01/23	520,75	9	Cornicabra	34567891
13	13	27/01/23	505,2	6	Picual	45678902
14	14	28/01/23	500	7	Royal	56789013
15	15	29/01/23	530,5	8	Hojiblanca	67890124
16	16	22/01/23	500	7	Royal	56789013
17	17	24/01/23	500	8	Hojiblanca	67890124

El último disparador de este tipo se encarga de comprobar que las inserciones o modificaciones de las tuplas de agricultores no tengan el campo teléfono como nulo. Esta restricción no era contemplada en el script de creación en la primera fase:

```
-- Trigger para hacer que el campo teléfono en agricultor no pueda ser nulo
CREATE OR REPLACE TRIGGER telefono_agricultor_not_null
BEFORE INSERT OR UPDATE ON agricultor
FOR EACH ROW
BEGIN
    IF :NEW.telefono IS null THEN
        RAISE_APPLICATION_ERROR(-20001, 'El campo teléfono no puede ser nulo.');
    END IF;
END;
/
```

```
-- Que no sea posible insertar agricultores con teléfono nulo
insert into agricultor values (45678111, 'Roberto Navarro', TO_DATE('08-08-1978', 'DD-MM-YYYY'), '', '
insert into agricultor values (56789222, 'Luisa Moreno', TO_DATE('14-02-1992', 'DD-MM-YYYY'), 98765444
```

```
Error que empieza en la línea: 88 del comando -
insert into agricultor values (45678111, 'Roberto Navarro', TO_DATE('08-08-1978', 'DD-MM-YYYY'), '', 'roberto.
Error en la línea de comandos : 88 Columna : 13
Informe de error -
Error SQL: ORA-20001: El campo teléfono no puede ser nulo.
ORA-06512: en "I72CASCM.TELEFONO_AGRICULTOR_NOT_NULL", línea 3
ORA-04088: error durante la ejecución del disparador 'I72CASCM.TELEFONO_AGRICULTOR_NOT_NULL'


1 fila insertadas.
```

DISPARADOR DE RESTRICCIÓN: INTEGRIDAD DE REFERENCIA

Este disparador se encargará de cumplir el papel de clave foránea definido en el script de creación de base de datos en la fase 1 entre las tablas de equipo de trabajo y lote (producción). Para ello, primero es necesario deshabilitar la clave foránea y tras esto, crear el disparador:

```
-- En primer lugar es necesario eliminar de clave foránea
ALTER TABLE produccionaceite drop CONSTRAINT fk_produccion_equipo;
/

-- Trigger que sustituye la restricción de integridad referencial entre las tablas equipoTrabajo y produccionAceite
CREATE OR REPLACE TRIGGER relacion_equipo_produccion
BEFORE INSERT OR UPDATE ON produccionaceite
FOR EACH ROW
DECLARE
    i number;
BEGIN
    SELECT count(*)
    INTO i
    FROM equipotrabajo
    WHERE idequipo = :NEW.idequipo;
    IF i = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Equipo de Trabajo inexistente.');
    END IF;
END;
/
```

```
-- Ejercicio 4 --
-- Inserciones para comprobar el trigger de restricción de integridad referencial entre equipoTrabajo y produccionAceite
insert into produccionAceite values (16, 1030.75, TO_DATE('26-02-2023', 'DD-MM-YYYY'), 0.81, 'Virgen', 12, 4);
insert into produccionAceite values (17, 1010.20, TO_DATE('27-02-2023', 'DD-MM-YYYY'), 0.79, 'Virgen Extra', 13, 5);
insert into produccionAceite values (18, 995.00, TO_DATE('28-02-2023', 'DD-MM-YYYY'), 0.78, 'Orujo', 14, 6);
insert into produccionAceite values (19, 1025.50, TO_DATE('01-03-2023', 'DD-MM-YYYY'), 0.8, 'Virgen', 15, 10);
```

```
Error que empieza en la línea: 99 del comando -
insert into produccionAceite values (18, 995.00, TO_DATE('28-02-2023', 'DD-MM-YYYY'), 0.78, 'Orujo', 14, 6)
Error en la línea de comandos : 99 Columna : 13
Informe de error -
Error SQL: ORA-20001: Equipo de Trabajo inexistente.
ORA-06512: en "I72CASCM.RELACION_EQUIPO_PRODUCCION", línea 9
ORA-04088: error durante la ejecución del disparador 'I72CASCM.RELACION_EQUIPO_PRODUCCION'


1 fila insertadas.
```

Al ejecutar las inserciones de prueba, podemos comprobar que las tres primeras no funcionan ya que los equipos de trabajo 4, 5 y 6 no fueron insertados en el script de creación, pero la última inserción se realiza con éxito debido a que el equipo 2 si existe.

DISPARADOR EN FUNCIÓN DE EXTENSIÓN DE OTRA TABLA

Este disparador se encarga de cubrir el supuesto de que un agricultor no pueda realizar más de 3 entregas en un mismo año. Para ello, el disparador compara el año de la tupla introducida con los existentes de ese mismo agricultor en la base de datos. Si cuenta 3 o más veces ese mismo año en la base de datos, da un error e impide realizar la inserción:

```
-- Trigger para que un agricultor no pueda realizar más de 3 entregas el mismo año
CREATE OR REPLACE TRIGGER limitar_entregas_agricultor
BEFORE INSERT ON entrega
FOR EACH ROW
DECLARE
    i number;
BEGIN
    SELECT count(*)
    INTO i
    FROM entrega
    WHERE dniagricultor = :NEW.dniagricultor AND (extract(year FROM fecha) = extract(year FROM :NEW.fecha));
    IF i >= 3 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Un agricultor no puede realizar más de 3 entregas el mismo año.');
    END IF;
END;
/
```

```
-- 2024
insert into entrega values (30, TO_DATE('26-01-2024', 'DD-MM-YYYY'), 520.75, 4, 'Cornicabra', 34567891);
insert into entrega values (31, TO_DATE('27-01-2024', 'DD-MM-YYYY'), 526.75, 5, 'Picual', 34567891);
insert into entrega values (32, TO_DATE('28-01-2024', 'DD-MM-YYYY'), 820.75, 6, 'Cornicabra', 34567891);
insert into entrega values (33, TO_DATE('29-01-2024', 'DD-MM-YYYY'), 580.75, 8, 'Picual', 34567891);
-- 2025
insert into entrega values (34, TO_DATE('30-01-2025', 'DD-MM-YYYY'), 620.75, 6, 'Cornicabra', 34567891);
insert into entrega values (36, TO_DATE('26-01-2025', 'DD-MM-YYYY'), 100.75, 9, 'Cornicabra', 34567891);
insert into entrega values (37, TO_DATE('26-01-2025', 'DD-MM-YYYY'), 102.75, 9, 'Picual', 34567891);
insert into entrega values (38, TO_DATE('26-01-2025', 'DD-MM-YYYY'), 20.75, 9, 'Cornicabra', 34567891);
```

Se puede observar que de las 4 entregas del año 2024 y de las 4 de 2025, solo han sido introducidas las tres primeras (todas son realizadas por el mismo agricultor):

18	30	26/01/24	520,75	4	Cornicabra	34567891
19	31	27/01/24	526,75	5	Picual	34567891
20	32	28/01/24	820,75	6	Cornicabra	34567891
21	34	30/01/25	620,75	6	Cornicabra	34567891
22	36	26/01/25	500	9	Cornicabra	34567891
23	37	26/01/25	500	9	Picual	34567891

FASE 3

En esta fase, se procederá a realizar una base de datos objeto-relacional a partir del enunciado descrito en la primera fase. El script "**objetoRelacional.sql**" es el encargado de realizar la creación de los objetos y tablas, mientras que "**manipulacion_objetoRealcional.sql**" es el script de pruebas de la base de datos. En este último script se encuentra una serie de altas, modificaciones y borrados de la base de datos para comprobar su funcionamiento.

Los objetos y las tablas en esta base de datos han sido creadas de la siguiente forma.

En primer lugar se define el objeto:

```
CREATE TYPE entrega_t AS OBJECT (
    idEntrega number(10),
    fecha date,
    peso number(10, 2),
    calidad number(5),
    tipoAceituna varchar2(50),
    rf_agricultor REF agricultor_t
);
/
```

Y posteriormente, se crea la tabla a partir del tipo:

```
CREATE TABLE entrega OF entrega_t(
    idEntrega primary key,
    fecha CONSTRAINT nn_entrega_fecha NOT NULL,
    peso CONSTRAINT ck_entrega_peso CHECK (peso >= 500) NOT NULL,
    calidad CONSTRAINT ck_entrega_calidad CHECK (calidad BETWEEN 0 AND 10),
    tipoAceituna CONSTRAINT ck_entrega_tipoAceituna CHECK (tipoAceituna IN ('Picual', 'Hojiblanca',
    rf_agricultor SCOPE IS agricultor
);
/
```

OBJETO PARA DEFINICIÓN DE LA ESTRUCTURA DE UNA TABLA

Para este ejercicio, se ha creado el objeto del tipo direccion_t, el cual tendrá una serie de atributos que ayuden con la localización de la dirección:

```
-- Ejercicio 1 --
CREATE TYPE direccion_t AS OBJECT(
    ciudad varchar2(64),
    calle varchar2(64),
    numero number(20),
    codigoPostal number(20)
);
/
```

Una vez se ha realizado la definición, se procede a crear las definiciones de los tipo objeto de agricultor y trabajador:

```
-- Creación de la base de datos de Al
CREATE TYPE agricultor_t AS OBJECT (
    dni number(8),
    nombreCompleto varchar2(64),
    fechaNacimiento date,
    telefono number(14),
    correo lista_correo_t,
    direccion direccion_t,
    empresa varchar2(64)
);
/
```

```
CREATE TYPE trabajador_t AS OBJECT (
    dni number(8),
    nombreCompleto VARCHAR2(64),
    fechaNacimiento DATE,
    telefono number(14),
    correo lista_correo_t,
    direccion direccion_t,
    puestoTrabajo varchar2(64),
    rf_equipoTrabajo REF equipoTrabajo_t
);
/
```

Se puede observar que el atributo dirección en estos tipos está definido a partir de la estructura creada anteriormente. Por último, solo queda crear sus tablas:

```
CREATE TABLE agricultor OF agricultor_t(
    dni primary key,
    nombreCompleto CONSTRAINT nn_agricultor_nombreCompleto NOT NULL,
    fechaNacimiento CONSTRAINT nn_agricultor_fechaNacimiento NOT NULL,
    empresa CONSTRAINT nn_agricultor_empresa NOT NULL
);
/
```

```
CREATE TABLE trabajador OF trabajador_t (
    dni primary key,
    nombreCompleto CONSTRAINT nn_trabajador_nombreCompleto NOT NULL,
    fechaNacimiento CONSTRAINT nn_trabajador_fechaNacimiento NOT NULL,
    puestoTrabajo CONSTRAINT ck_trabajador_puestoTrabajo CHECK (puestoTrabajo IN ('Recepción',
    rf_equipoTrabajo SCOPE IS equipoTrabajo
);
```

TABLA CON ATRIBUTO QUE A SU VEZ ES UNA TABLA

En este apartado, se procede a la creación de una tabla anidada. Se seguirá el siguiente planteamiento para llevar a cabo esta implementación:

Se quiere crear un atributo en los equipo de trabajo que a su vez sea una tabla con las entregas que cada equipo de trabajo debe realizar.

Para llevar a cabo esto, en primer lugar crea un tipo tabla de referencias a partir del tipo entrega_t y se introduce dicho tipo en la definición del tipo equipo de trabajo:

```
CREATE TYPE entregasList AS TABLE OF REF entrega_t;
/

CREATE TYPE equipoTrabajo_t AS OBJECT(
    idEquipo number(10),
    nombreEquipo varchar2(100),
    entregas EntregasList
);
/
```

Y tras esto, se crea la tabla equipoTrabajo junto con la tabla anidada para poder introducir datos posteriormente:

```
CREATE TABLE equipoTrabajo OF equipoTrabajo_t (
    idEquipo PRIMARY KEY,
    nombreEquipo CONSTRAINT nn_equipoTrabajo_nombreEquipo NOT NULL
) NESTED TABLE entregas STORE AS nestedEntregas
/
```

TABLA CON ATRIBUTO ARRAY

A continuación, se muestra la creación de un atributo de tipo array, en este caso correos, que permitirá que tanto los agricultores como los trabajadores puedan tener más de un correo electrónico asignado.

En primer lugar, se define el tipo array de tipo varchar2:

```
-- Ejercicio 3
CREATE TYPE lista_correo_t AS VARRAY(10) OF VARCHAR2(64);
```

Y posteriormente, este es introducido en el atributo correo de los agricultores y trabajadores de la siguiente forma:

```
-- Creación de la base de datos de Alma:
CREATE TYPE agricultor_t AS OBJECT (
    dni number(8),
    nombreCompleto varchar2(64),
    fechaNacimiento date,
    telefono number(14),
    correo lista_correo_t,
    direccion direccion_t,
    empresa varchar2(64)
);
/
```

```
CREATE TYPE trabajador_t AS OBJECT (
    dni number(8),
    nombreCompleto VARCHAR2(64),
    fechaNacimiento DATE,
    telefono number(14),
    correo lista_correo_t,
    direccion direccion_t,
    puestoTrabajo varchar2(64),
    rf_equipoTrabajo REF equipoTrabajo_t
);
/
```

Por último, se crean sus tablas con sus correspondientes restricciones:

```
CREATE TABLE agricultor OF agricultor_t(
    dni primary key,
    nombreCompleto CONSTRAINT nn_agricultor_nombreCompleto NOT NULL,
    fechaNacimiento CONSTRAINT nn_agricultor_fechaNacimiento NOT NULL,
    empresa CONSTRAINT nn_agricultor_empresa NOT NULL
);
/
```

```
CREATE TABLE trabajador OF trabajador_t (
    dni primary key,
    nombreCompleto CONSTRAINT nn_trabajador_nombreCompleto NOT NULL,
    fechaNacimiento CONSTRAINT nn_trabajador_fechaNacimiento NOT NULL,
    puestoTrabajo CONSTRAINT ck_trabajador_puestoTrabajo CHECK (puestoTrabajo IN ('Re
    rf_equipoTrabajo SCOPE IS equipoTrabajo
);
```

DEFINIR UN MÉTODO PARA UN OBJETO

Para realizar este ejercicio, se ha definido que si la producción de un lote es menor a 600 litros, se considera “pequeña”, si es entre 600 y 700 litros se considera “Mediana” y si es mayor a 700 se considera “grande”. A continuación, se proporciona una captura de pantalla de la función de “produccionAceite_t” que hace posible realizar esta clasificación:

```
CREATE OR REPLACE TYPE BODY produccionAceite_t AS
    MEMBER FUNCTION clasificarProduccion return varchar2 IS
    BEGIN
        IF litros < 600 THEN
            return 'Pequeña';
        ELSIF litros between 600 and 700 then
            return 'Mediana';
        ELSE
            return 'Grande';
        END IF;
    END;
END;
/
```

Para poder probarla, se realiza una serie de inserciones en la tabla produccionAceite para poder realizar esta clasificación:

```
CREATE TYPE produccionAceite_t AS OBJECT (
    idProduccion number(10),
    litros number(10, 2),
    fecha date,
    rendimiento number(5, 2),
    tipoAceite varchar2(50),
    rf_entrega REF entrega_t,
    rf_equipoTrabajo REF equipoTrabajo_t,
    MEMBER FUNCTION clasificarProduccion return varchar2
);
/
```

```
CREATE TABLE produccionAceite OF produccionAceite_t (
    idProduccion primary key,
    litros CONSTRAINT nn_produccionAceite_litros NOT NULL,
    fecha CONSTRAINT nn_produccionAceite_fecha NOT NULL,
    tipoAceite CONSTRAINT ck_produccionAceite_tipoAceite CHECK (tipoAceite IN ('Virger
    rf_entrega SCOPE IS entrega,
    rf_equipoTrabajo SCOPE IS equipoTrabajo
);
/
```

```
INSERT INTO produccionAceite VALUES (
    1,
    500.00,
    TO_DATE('2023-06-01', 'YYYY-MM-DD'),
    0.80,
    'Virgen Extra',
    (SELECT REF(e) FROM entrega e WHERE e.idEntrega = 1001),
    (SELECT REF(et) FROM equipoTrabajo et WHERE et.idEquipo = 1)
);
```

IDPRODUCCION	LITROS	FECHA	RENDIMIENTO	TIPOACEITE	RF_ENTREGA	RF_EQUIPOTRABAJO
1	500	01/06/23	0,8	Virgen Extra	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
2	600	02/06/23	0,85	Virgen	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
3	700	03/06/23	0,75	Orujo	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
4	700	04/06/23	0,77	Virgen Extra	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
5	900	05/06/23	0,78	Virgen	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]

Ahora, se realiza una consulta para mostrar los datos de esta tabla y así poder comprobar el correcto funcionamiento de este método:

```
SELECT p.idProduccion, p.litros, p.clasificarProduccion() AS Clasificacion
FROM produccionAceite p;
```

	IDPRODUCCION	LITROS	CLASIFICACION
1	1	500	Pequeña
2	2	600	Mediana
3	3	700	Mediana
4	4	700	Mediana
5	5	900	Grande

De esta forma, se pueden realizar clasificaciones de los litros de aceite en esta base de datos.

SCRIPT DE MANIPULACIÓN FASE 3

Para finalizar la fase 3, se va a realizar una breve revisión del script "**manipulacion_objetoRelacional.sql**". En el script se realizan todo lo relacionado con el alta, modificación y borrado de esta base de datos, se mostrará ahora algunos fragmentos del script con su correspondiente explicación.

En primer lugar, se realizan una serie de inserciones en las diferentes tablas. Se realizan inserciones en todas las tablas en este script, pero sólo se mostrarán algunas de ellas ya que se trata de un proceso repetitivo:

Tabla Agricultor, aquí se puede apreciar cómo se introduce el array de correos y la estructura de dirección:

```
INSERT INTO agricultor VALUES (
    12345678,
    'Juan Pérez',
    TO_DATE('1980-01-15', 'YYYY-MM-DD'),
    987654321,
    lista_correo_t('juan.perez@example.com', 'perez.juan@example.com'),
    direccion_t('Cordoba', 'Calle 1', 123, 100),
    'Aceitunas Pérez S.L.'
);

INSERT INTO agricultor VALUES (
    12345738,
    'Luis Pepe',
    TO_DATE('2000-01-15', 'YYYY-MM-DD'),
    987654321,
    lista_correo_t('luis.pepe@example.com', 'pepe.luis@example.com'),
    direccion_t('Sevilla', 'Costa', 123, 1030),
    'Olivos Pepe S.A.'
);
```

DNI	NOMBRECOMPLETO	FECHANACIMIENTO	TELEFONO	CORREO	DIRECCION	EMPRESA
12345678	Luis Luis	15/01/80	987654321	I72CASCM.LISTA_CO...	[I72CASCM.DIRECCION_T]	Olivos Luis
12345738	Luis Pepe	15/01/00	987654321	I72CASCM.LISTA_CO...	[I72CASCM.DIRECCION_T]	Olivos Pepe S.A.
23456789	Ana Gómez	22/05/75	987654322	I72CASCM.LISTA_CO...	[I72CASCM.DIRECCION_T]	Gómez Agricultura
34567890	Carlos Martín	30/08/88	987654333	I72CASCM.LISTA_CO...	[I72CASCM.DIRECCION_T]	Martín Olivareros
45678901	Sara López	12/12/90	987654334	I72CASCM.LISTA_CO...	[I72CASCM.DIRECCION_T]	López Aceitunas S.L.

Tabla producción de aceite, donde se puede observar cómo se tratan las interrelaciones en este tipo de tablas mediante referencias:

```
INSERT INTO produccionAceite VALUES (
    2,
    600.00,
    TO_DATE('2023-06-02', 'YYYY-MM-DD'),
    0.85,
    'Virgen',
    (SELECT REF(e) FROM entrega e WHERE e.idEntrega = 1002),
    (SELECT REF(et) FROM equipoTrabajo et WHERE et.idEquipo = 1)
);

INSERT INTO produccionAceite VALUES (
    3,
    700.00,
    TO_DATE('2023-06-03', 'YYYY-MM-DD'),
    0.75,
    'Orujo',
    (SELECT REF(e) FROM entrega e WHERE e.idEntrega = 1003),
    (SELECT REF(et) FROM equipoTrabajo et WHERE et.idEquipo = 2)
);
```

	IDPRODUCCION	LITROS	FECHA	RENDIMIENTO	TIPOACEITE	RF_ENTREGA	RF_EQUIPOTRABAJO
1	1	500	01/06/23	0,8	Virgen Extra	oracle.sql.REF@689d32d4	[I72CASCM.EQUIPOTRABAJO_T]
2	2	600	02/06/23	0,85	Virgen	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
3	3	700	03/06/23	0,75	Orujo	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
4	4	700	04/06/23	0,77	Virgen Extra	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]
5	5	900	05/06/23	0,78	Virgen	[I72CASCM.ENTREGA_T]	[I72CASCM.EQUIPOTRABAJO_T]

Tabla de trabajadores, donde nuevamente se puede observar el tipo array y estructura:

```
INSERT INTO trabajador VALUES (
    87654325,
    'Carlos Sánchez',
    TO_DATE('1995-06-05', 'YYYY-MM-DD'),
    987654349,
    lista_correo_t('carlos.sanchez@example.com'),
    direccion_t('Jaén', 'Calle 7', 92, 2005),
    'Recepción',
    (SELECT REF(e) FROM equipoTrabajo e WHERE e.idEquipo = 2)
);
```

	DNI	NOMBRECOMPLETO	FECHANACIMIENTO	TELEFONO	CORREO	DIRECCION	PUESTOTRABAJO	RF_EQUIPOTRABAJO
1	87654321	Pedro García	15/02/85	987654345	I72CASCM.LISTA_CORREO_T('pedro.garcia@example.com')	[I72CASCM.DIRECCION_T]	Recepción	[I72CASCM.EQUIPOTRABAJO_T]
2	87654322	María Rodríguez	20/03/90	987654346	I72CASCM.LISTA_CORREO_T('maria.rodriguez@example.com')	[I72CASCM.DIRECCION_T]	Limpieza	[I72CASCM.EQUIPOTRABAJO_T]
3	87654323	José López	25/04/88	987654347	I72CASCM.LISTA_CORREO_T('jose.lopez@example.com')	[I72CASCM.DIRECCION_T]	Batidora	[I72CASCM.EQUIPOTRABAJO_T]
4	87654324	Ana Torres	30/05/92	987654348	I72CASCM.LISTA_CORREO_T('ana.torres@example.com')	[I72CASCM.DIRECCION_T]	Almacenamiento	[I72CASCM.EQUIPOTRABAJO_T]
5	87654325	Carlos Sánchez	05/06/95	987654349	I72CASCM.LISTA_CORREO_T('carlos.sanchez@example.com')	[I72CASCM.DIRECCION_T]	Recepción	[I72CASCM.EQUIPOTRABAJO_T]

A continuación se realiza una serie de modificaciones a los datos anteriormente introducidos:

```sql
-- MODIFICAR DATOS --
-- Agricultor --
UPDATE agricultor
SET
    nombreCompleto = 'Luis Luis',
    empresa = 'Olivos Luis'
WHERE dni = 12345678;

-- Entrega --
UPDATE entrega
SET peso = 2000
WHERE idEntrega = 1003;

-- Equipo Trabajo --
UPDATE equipoTrabajo
SET
    idEquipo = 3,
    nombreEquipo = 'Equipo Charlie'
WHERE idEquipo = 2;

-- Producion aceite --
UPDATE produccionAceite
SET
    litros = 700,
    rf_equipoTrabajo = (SELECT REF(et) FROM equipoTrabajo et WHERE et.idEquipo = 1)
WHERE idProduccion = 4;

-- Trabajador --
UPDATE trabajador
SET
    direccion = direccion_t('Cordoba', 'Calle Juderia', 45, 14005)
WHERE dni = 87654321;
```

Estos pueden ser consultados para comprobar que han sido modificados con éxito:

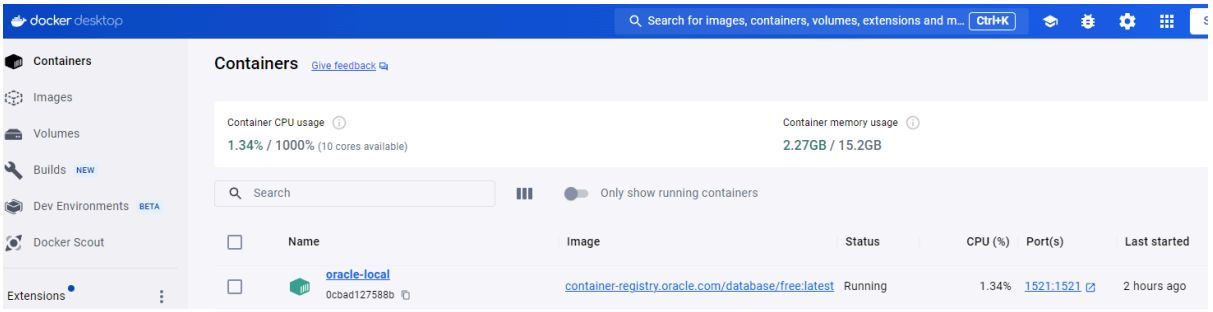
```sql
-- CONSULTAR DATOS --
SELECT * FROM agricultor;
SELECT * FROM entrega;
SELECT * FROM equipoTrabajo;
SELECT * FROM produccionAceite;
SELECT * FROM trabajador;
```

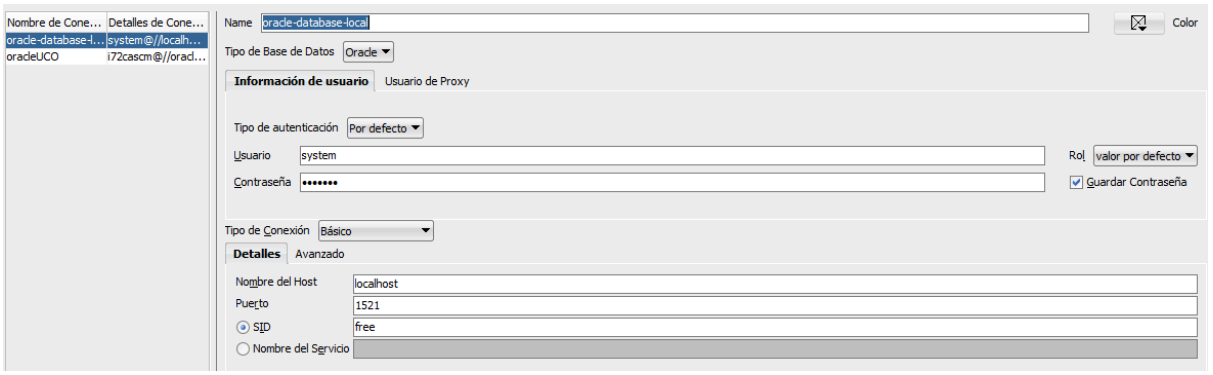
Por último, se realiza una serie de borrado de las distintas tablas a modo de prueba:

```sql
-- BORRAR DATOS --
-- Agricultor --
DELETE FROM agricultor
WHERE dni = 23456789;
-- Entrega --
DELETE FROM entrega
WHERE idEntrega = 1004;
-- Equipo trabajo --
DELETE FROM equipoTrabajo
WHERE idEquipo = 1;
-- Produccion aceite --
DELETE FROM produccionAceite
WHERE idProduccion = 4;
-- Trabajador --
DELETE FROM trabajador
WHERE dni = 87654323;
```

FASE 4

Para la creación de un middleware que pueda realizar consultas y modificar la base de datos ha sido desarrollada durante la **fase 1** y **fase 2**, se ha optado por el uso del lenguaje de programación **Python**, el cual es empleado para la creación de diversas aplicaciones web. Adicionalmente, se ha utilizado la aplicación **Docker** para la creación de una base de datos en oracle y, de esta manera, realizar la conexión y manipulación de la base de datos:





A continuación, se procede a mostrar capturas y explicar el script programado en python llamado "**middleware.py**", donde se encuentran codificadas todas las funciones de consulta y manipulación de las tablas "**ENTREGA**" y "**PRODUCCIONACEITE**" a modo de back-end.

CONEXIÓN:

Mediante las siguientes líneas de código se puede realizar la conexión a la base de datos alojada en Docker:

```python
# Conexión con la base de datos
def conectar_a_oracle():
    try:
        cnx = oracledb.connect(user="system", password="Ora1234", dsn="localhost/free")
        return cnx
    except oracledb.DatabaseError as e:
        print("Error al conectarse a la base de datos:", e)
        return None
```

MENÚ PRINCIPAL Y SUBMENÚS:

Permite la navegación en la aplicación entre las tablas de entregas y lotes y sus diversas opciones:

```python
# Menu principal del programa
def main():
    cnx = conectar_a_oracle()
    if cnx is not None:
        while True:
            print("1. Ir a los datos de entregas")
            print("2. Ir a los datos de lotes")
            print("3. Salir")
            opc = input("Ingrese su opción: ")
            if opc == '1':
                opc = submenu_entregas(cnx)
            elif opc == '2':
                submenu_lotes(cnx)
            elif opc == '3':
                print("Saliendo del programa.")
                break
            else:
                print("ERROR: Esa opción no es válida, intentelo nuevamente.")

        cnx.close()
```

```python
# Submenu de la tabla entregas
def submenu_entregas(cnx):
    while True:
        print("\nSubmenú - Datos de Entregas")
        print("1. Consultar tabla de datos de entregas")
        print("2. Insertar dato")
        print("3. Modificar dato")
        print("4. Borrar dato")
        print("5. Salir del submenú")
        opcion = input("Ingrese su opción: ")

        if opcion == '1':
            consultar_entregas(cnx)
            break
        elif opcion == '2':
            insercion_entrega(cnx)
            break
        elif opcion == '3':
            modificar_entrega(cnx)
            break
        elif opcion == '4':
            borrar_entrega(cnx)
            break
        elif opcion == '5':
            print("Regresando al menú principal...")
            break
        else:
            print("ERROR: Esa opción no es válida, intentelo nuevamente.")
```

```python
# Submenu de la tabla lotes (produccionAcetite)
def submenu_lotes(cnx):
    while True:
        print("\nSubmenú - Datos de Lotes")
        print("1. Consultar tabla de datos de lotes")
        print("2. Insertar dato")
        print("3. Modificar dato")
        print("4. Borrar dato")
        print("5. Salir del submenú")
        opc = input("Ingrese su opción: ")

        if opc == '1':
            consultar_lotes(cnx)
            break
        elif opc == '2':
            insercion_lote(cnx)
            break
        elif opc == '3':
            modificar_lote(cnx)
            break
        elif opc == '4':
            borrar_lote(cnx)
            break
        elif opc == '5':
            print("Regresando al menú principal...")
            break
        else:
            print("ERROR: Esa opción no es válida, intentelo nuevamente.")
```

INSERCIONES

A partir de este punto se va a explicar el código referente a la manipulación de la tabla entregas, puesto a que el código y funcionamiento es similar para la tabla de lotes.

En la siguiente imagen se puede apreciar cómo en primer lugar se pide al usuario que introduzca el valor de los atributos de la tabla entrega para posteriormente realizar la inserción en la tabla:

```python
# Insertar en tabla entrega
def insercion_entrega(cnx):
    id_entrega = input("Introduzca el ID de la entrega: ")
    fecha = input("Introduzca la fecha en el formato DD-MM-YYYY: ")
    peso = input("Introduzca el peso de la entrega en kilogramos: ")
    calidad = input("Introduzca la calidad de la entrega (0-10): ")
    tipo = input("Introduzca el tipo de aceituna (Picual, Hojiblanca, Royal, Cornicabra): ")
    dni = input("Introduzca el DNI del agricultor: ")

    SQL = '''INSERT INTO entrega (idEntrega, fecha, peso, calidad, tipoAceituna, dniAgricultor)
             VALUES (:id_entrega, TO_DATE(:fecha, 'DD-MM-YYYY'), :peso, :calidad, :tipo_aceituna, :dni_agricultor)'''

    try:
        cursor = cnx.cursor()
        cursor.execute(SQL, [id_entrega, fecha, peso, calidad, tipo, dni])
        cnx.commit()
        print("Entrega insertada con éxito.")
    except oracledb.DatabaseError as e:
        print("Error al insertar en la base de datos:", e)
    finally:
        cursor.close()
```

A continuación se muestra un ejemplo de inserción (los campos deben ser escritos de manera correcta y el dni del agricultor debe existir previamente en la tabla de agricultores):

```
Submenú - Datos de Entregas
1. Consultar tabla de datos de entregas
2. Insertar dato
3. Modificar dato
4. Borrar dato
5. Salir del submenú
Ingrese su opción: 2
Introduzca el ID de la entrega: 22
Introduzca la fecha en el formato DD-MM-YYYY: 05-03-2023
Introduzca el peso de la entrega en kilogramos: 900
Introduzca la calidad de la entrega (0-10): 7.6
Introduzca el tipo de aceituna (Picual, Hojiblanca, Royal, Cornicabra): Royal
Introduzca el DNI del agricultor: 12345679
Entrega insertada con éxito.
```

13	13	27/01/23	505,2	6	Picual	45678902
14	14	28/01/23	1095	7	Royal	56789013
15	15	29/01/23	530,5	8	Hojiblanca	67890124
16	22	05/03/23	900	8	Royal	12345679

MODIFICACIONES

Es la más compleja de las funciones, en primer lugar se pide al usuario que introduzca el ID de la entrega que se desea modificar, junto con los nuevos valores (todo valor que no se desee modificar, debe dejarse en blanco).

```
def modificar_entrega(cnx):
    id_entrega = input("Introduca la ID de la entrega que desea modificar.")
    print("Los valores que no sean modificados deben dejarse en blanco.")
    nueva_fecha = input("Introduzca la nueva fecha en el formato DD-MM-YYYY: ")
    nuevo_peso = input("Introduzca el nuevo peso: ")
    nueva_calidad = input("Introduzca la nueva calidad (0-10): ")
    nuevo_tipo_aceituna = input("Introduzca el nuevo tipo de aceituna (Picual, Hojiblanca, Royal, Cornicabra): ")
    nuevo_dni_agricultor = input("Introduzca el nuevo DNI del agricultor: ")
```

A continuación, se crea la consulta SQL que realizará la modificación en la tabla. En esta sentencia, "**NVL(exp1, exp2)**" comprueba si la expresión 1 es nula, en caso de que no sea nula se modifica el atributo de la tupla. Si la expresión 1 es nula, se deja ese atributo sin modificar (exp2 es el antiguo valor del atributo):

```
SQL = """
UPDATE entrega
SET fecha = NVL(TO_DATE(:nueva_fecha, 'DD-MM-YYYY'), fecha),
    peso = NVL(:nuevo_peso, peso),
    calidad = NVL(:nueva_calidad, calidad),
    tipoAceituna = NVL(:nuevo_tipo_aceituna, tipoAceituna),
    dniAgricultor = NVL(:nuevo_dni_agricultor, dniAgricultor)
WHERE idEntrega = :id_entrega
"""
```

Por último, se ejecuta la sentencia SQL para realizar la modificación en la tabla:

```
try:
    cursor = cnx.cursor()
    cursor.execute(SQL, [nueva_fecha if nueva_fecha else None,
                         nuevo_peso if nuevo_peso else None,
                         nueva_calidad if nueva_calidad else None,
                         nuevo_tipo_aceituna if nuevo_tipo_aceituna else None,
                         nuevo_dni_agricultor if nuevo_dni_agricultor else None,
                         id_entrega])
    cnx.commit()
    if cursor.rowcount > 0:
        print("Entrega modificada con éxito.")
    else:
        print("No se encontró una entrega con el ID proporcionado.")
except oracledb.DatabaseError as e:
    print("Error al modificar en la base de datos:", e)
finally:
    cursor.close()
```

Se procede a mostrar un ejemplo de modificación:

3	3	17/01/23	600,75	9	Cornicabra	34567890
4	4	18/01/23	530,2	6	Royal	45678901
5	5	19/01/23	900	1	Hojiblanca	56789012

```
Submenú - Datos de Entregas
1. Consultar tabla de datos de entregas
2. Insertar dato
3. Modificar dato
4. Borrar dato
5. Salir del submenú
Ingrese su opción: 3
Introduca la ID de la entrega que desea modificar.4
Los valores que no sean modificados deben dejarse en blanco.
Introduzca la nueva fecha en el formato DD-MM-YYYY:
Introduzca el nuevo peso: 670.54
Introduzca la nueva calidad (0-10): 8
Introduzca el nuevo tipo de aceituna (Picual, Hojiblanca, Royal, Cornicabra): Picual
Introduzca el nuevo DNI del agricultor:
Entrega modificada con éxito.
```

3	3	17/01/23	600,75	9	Cornicabra	34567890
4	4	18/01/23	670,54	8	Picual	45678901
5	5	19/01/23	900	1	Hojiblanca	56789012

BORRADOS

En esta funcionalidad, solo es necesario introducir la ID de la entrega que se desea eliminar:

```python
# Borrar tupla de la tabla entrega
def borrar_entrega(cnx):
    id_entrega = input("Introduzca la ID de la entrega que desea eliminar: ")

    SQL = '''DELETE FROM entrega WHERE idEntrega = :id_entrega'''

    try:
        cursor = cnx.cursor()
        cursor.execute(SQL, [id_entrega])
        cnx.commit()
        if cursor.rowcount > 0:
            print("Entrega borrada con éxito.")
        else:
            print("No se encontró una entrega con ese ID, cancelando borrado.")
    except oracledb.DatabaseError as e:
        print("Error al borrar en la base de datos:", e)
    finally:
        cursor.close()
```

(variable) cursor: Any

14	14	28/01/23	1095	7	Royal	56789013
15	15	29/01/23	530,5	8	Hojiblanca	67890124
16	22	05/03/23	900	8	Royal	12345679

```
Submenú - Datos de Entregas
1. Consultar tabla de datos de entregas
2. Insertar dato
3. Modificar dato
4. Borrar dato
5. Salir del submenú
Ingrese su opción: 4
Introduzca la ID de la entrega que desea eliminar: 22
Entrega borrada con éxito.
```

14	14	28/01/23	1095	7	Royal	56789013
15	15	29/01/23	530,5	8	Hojiblanca	67890124

CONSULTAS

Esta última función permite imprimir por consola toda la tabla de entregas de la base de datos para poder ir comprobando el correcto funcionamiento de los anteriores apartados:

```python
# Consultar tabla entregas
def consultar_entregas(cnx):
    cursor = cnx.cursor()
    cursor.execute("SELECT * FROM entrega")
    registros = cursor.fetchall()
    cursor.close()
    if not registros:
        print("No hay registros en la tabla de entregas.")
    else:
        for x in registros:
            print(x)
```

```
Submenú - Datos de Entregas
1. Consultar tabla de datos de entregas
2. Insertar dato
3. Modificar dato
4. Borrar dato
5. Salir del submenú
Ingrese su opción: 1
(1, datetime.datetime(2023, 1, 15, 0, 0), 500.5, 8, 'Picual', 12345678)
(2, datetime.datetime(2023, 1, 16, 0, 0), 550.0, 7, 'Hojiblanca', 23456789)
(3, datetime.datetime(2023, 1, 17, 0, 0), 600.75, 9, 'Cornicabra', 34567890)
(4, datetime.datetime(2023, 1, 18, 0, 0), 670.54, 8, 'Picual', 45678901)
(5, datetime.datetime(2023, 1, 19, 0, 0), 900.0, 1, 'Hojiblanca', 56789012)
(6, datetime.datetime(2023, 1, 20, 0, 0), 560.5, 8, 'Royal', 67890123)
(7, datetime.datetime(2023, 1, 21, 0, 0), 890.3, 5, 'Picual', 78901234)
(8, datetime.datetime(2023, 1, 22, 0, 0), 510.0, 7, 'Cornicabra', 89012345)
(9, datetime.datetime(2023, 1, 23, 0, 0), 550.25, 8, 'Picual', 90123456)
(10, datetime.datetime(2023, 1, 24, 0, 0), 500.0, 6, 'Royal', 12345679)
(11, datetime.datetime(2023, 1, 25, 0, 0), 680.5, 7, 'Hojiblanca', 23456780)
(12, datetime.datetime(2023, 1, 26, 0, 0), 520.75, 9, 'Cornicabra', 34567891)
(13, datetime.datetime(2023, 1, 27, 0, 0), 505.2, 6, 'Picual', 45678902)
(14, datetime.datetime(2023, 1, 28, 0, 0), 1095.0, 7, 'Royal', 56789013)
(15, datetime.datetime(2023, 1, 29, 0, 0), 530.5, 8, 'Hojiblanca', 67890124)
```