IISSI 1

Tutor: Sergio Segura Rueda

Autores: Alejandro Manuel Gestoso Torres

Nicolás Sánchez Mendoza



Proyecto Bertrans

**Índice**

[Historial de versiones 2](#_Toc24909686)

[Introducción al problema 3](#_Toc24909687)

[Glosario de términos 4](#_Toc24909688)

[Visión general del sistema 5](#_Toc24909689)

[Catálogo de requisitos 6](#_Toc24909690)

[**Requisitos generales** 6](#_Toc24909691)

[**Requisitos de información** 8](#_Toc24909692)

[**Reglas de negocio y pruebas de aceptación** 10](#_Toc24909693)

[**Requisitos funcionales** 17](#_Toc24909694)

[**Requisitos no funcionales** 23](#_Toc24909695)

[Modelado conceptual 24](#_Toc24909696)

[**Diagramas de clases UML con restricciones** 24](#_Toc24909697)

[**Escenarios de prueba con descripción textual y diagrama de objetos UML** 27](#_Toc24909698)

[Matrices de trazabilidad 30](#_Toc24909699)

[Modelo relacional en 3FN 34](#_Toc24909700)

[Modelo tecnológico 35](#_Toc24909701)

Enlace del repositorio: <https://github.com/AlexD6/ProyectoBertrans.git>

Historial de versiones

|  |  |  |
| --- | --- | --- |
| **Versión** | **Detalles** | **Participantes** |
| Entregable v1 (septiembre) | * Introducción al problema * Glosario de términos * Visión general del sistema * Catálogo de requisitos * Requisitos generales * Requisitos de información * Reglas de negocio y pruebas de aceptación * Requisitos funcionales * Requisitos no funcionales * Modelado conceptual * Diagramas de clases UML con restricciones * Escenarios de prueba con descripción textual y diagrama de objetos UML * Matrices de trazabilidad * Modelo relacional en 3FN * Modelo tecnológico | Alejandro Gestoso Torres  Nicolás Sánchez Mendoza |
| Entregable v2 (diciembre) | * Corrección de errores de la v1:   + Mejora de la redacción en general   + Arreglar RG, RI, RN, RF y RnF   + Mejora de prueba de aceptación y modelado conceptual   + Inclusión de IDs en las matrices de trazabilidad | Alejandro Gestoso Torres  Nicolás Sánchez Mendoza |

Introducción al problema

Nuestro cliente es el dueño de una empresa de Sevilla, “Bertrans”, empresa de transporte de diferentes tipos de productos por toda Andalucía. El funcionamiento es simple: se reciben una serie de encargos cada día y se ha de gestionar el reparto y envío de dichos pedidos.

Tras nuestra entrevista con el cliente, estudiar el dominio del problema y conocer cuáles son las necesidades que se quieren satisfacer, hemos podido concluir la necesidad de una herramienta web que permita aportar más ventajas a la hora de realizar la gestión el negocio.

El principal problema que tiene nuestro cliente es la falta de una herramienta que permita gestionar de una forma eficiente los pedidos y repartos a realizar, de forma que cada pedido se asigne a un trabajador y un tipo de mercancía a un camión específico.

Otro de los problemas a solventar es la asignación de camiones a los trabajadores, la herramienta web a desarrollar deberá ser capaz de asignar a cada trabajador un camión específico para el tipo de mercancía que éste vaya a llevar.

A esto se le suma un sistema de penalizaciones que será usado con cada conductor de manera que permita llevar una cuenta de las incidencias producidas y la gestión de estas.

Y por último también la idea de que nuestro cliente pueda llevar a cabo una correcta gestión de los clientes (sean personas o empresas) que realizan los pedidos.

Por tanto, el objetivo de nuestro proyecto es llevar a cabo la creación de un sistema funcional que satisfaga todas las necesidades mostradas anteriormente, con esta herramienta web, nuestro cliente quiere conseguir una mayor eficiencia y accesibilidad debido a que a día de hoy el método utilizado para llevar a cabo estas tareas es a mano, mediante papel y bolígrafo.

Glosario de términos

**Administrador**: Persona encargada del desarrollo del sistema informático y del mantenimiento de este.

**Conductor:** Persona encargada de la conducción y reparto mediante uno de los camiones.

**Datos personales**: Datos con los cuales podemos identificar a una persona, por ejemplo: DNI, nombre, apellidos, teléfono, etc.

**Datos laborales**: Datos que incluyen todo lo relacionado con el trabajo del personal de la empresa.

**Expediente**: Registro documental del personal de la empresa, que incluye datos personales y laborales.

**Horario**: Tiempo durante el cual se encuentra abierta la tienda y se realizan actividades en la empresa.

**Identificador**: Código alfanumérico asignado a productos y trabajadores que los distingue a unos de otros.

**Logística**: Conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa o de un servicio, especialmente de distribución.

**Plataforma**: Edificación donde se encuentra la mercancía que será repartida. Hay de dos tipos: frío (que engloba mercancía fría y congelada) y seco.

**Propietario:** Persona dueña del negocio.

**Tacógrafo:** Aparato que se utiliza para medir la distancia recorrida y la velocidad de cada camión.

**Termógrafo:** Aparato usado para registrar los cambios de temperatura producidos en los camiones.

**Time-out**: Tiempo de respuesta de la aplicación para mostrar los resultados obtenidos en una búsqueda en el sistema de información.

Visión general del sistema

En este proyecto proponemos una solución dentro de una plataforma web, que sirva como herramienta de fácil acceso, consulta y actualización online, que integre todo lo relacionado con el negocio.

Para conseguir lo anteriormente nombrado en la introducción al problema, el sistema debe tener una base de datos que gestione los repartos a realizar, para poder tener controlado que todos los pedidos lleguen a su destino en perfectas condiciones, en un camión u otro dependiendo de su tipo de transporte requerido (frío, seco o congelado), a la hora acordada con el comercio para su descarga.

Además, una de las funciones que ha de realizar es la de gestionar los clientes de la empresa, es decir, a quién hacemos el envío si se trata de una persona o una empresa en sí.

Otra de las funciones a realizar es la de la gestión y mantenimiento de los camiones. Ver si están disponibles o necesitan pasar alguna revisión. También pueden haberse averiado y necesitarla sin que sea su fecha requerida.

También tendría que llevar a cabo una gestión de la mercancía. Conocer su tipo, cantidad y tipo de conserva para una correcta manipulación de esta.

 Y por último será necesario implementar un sistema de gestión de conductores y sus penalizaciones. Esto abarcará desde las horas que trabajen y días libres de cada uno hasta un sistema de puntos que, dependiendo de la incidencia cometida podría llegar a ser desde una llamada de atención hasta un despido.

Catálogo de requisitos

**Requisitos generales**

**RG-01:** Empleados

**COMO** propietario,

**QUIERO** un sistema para administrar los empleados

**PARA** poder llevar una correcta gestión de estos.

**RG-02:** Incidencias

**COMO** propietario,

**QUIERO** un sistema para administrar las incidencias

**PARA** poder gestionar de forma correcta las incidencias causadas por cada conductor.

**RG-03:** Camiones

**COMO** propietario,

**QUIERO** un sistema que permita administrar correctamente los camiones

**PARA** poder llevar una correcta gestión de estos, así como los empleados asignados a cada uno.

**RG-04:** Pedidos

**COMO** propietario,

**QUIERO** un sistema para administrar los pedidos

**PARA** poder llevar una correcta gestión de estos, así como del conductor encargado del mismo

**RG-05:** Mercancías

**COMO** propietario,

**QUIERO** un sistema para administrar las mercancías

**PARA** poder llevar una correcta gestión de estas, además de saber si requieren algún tratamiento especial.

**RG-06:** Clientes

**COMO** propietario,

**QUIERO** un sistema para administrar los clientes

**PARA** poder llevar una correcta gestión de los mismos.

**Requisitos de información**

**RI\_01:** Como propietario de la empresa quiero disponer de la siguiente información sobre los trabajadores:

-Datos personales como nombre, apellidos, DNI…

-Su sueldo

-Tipo de trabajador

-Horas que ha conducido

-Puntos acumulados

**RI\_02:** Como propietario de la empresa quiero disponer de la siguiente información sobre las incidencias:

-ID de la incidencia

-Causa

-Gravedad

-Fecha de la penalización

-Puntos que supone

**RI\_03:** Como propietario de la empresa quiero disponer de la siguiente información sobre los pedidos:

-ID del pedido

-Fecha de aviso

-Fecha de pedido

-Hora de salida, hora de llegada y la hora de llegada prevista

-Dirección del destino

**RI\_04:** Como dueño de la empresa quiero disponer de la siguiente información acerca de las mercancías:

-ID de la mercancía

-Tipo de conserva

-Nombre de la mercancía

-Cantidad de la mercancía

- Tipo de mercancía que es (textil, alimentación, jardinería).

**RI\_05:** Como dueño de la empresa quiero disponer de la siguiente información sobre los camiones:

-Matrícula

-Fecha de revisión

-Renovación

-Tipo de camión

-Disponibilidad del camión

-Fecha de disponibilidad

**RI\_06:** Como dueño de la empresa quiero disponer de la siguiente información sobre los clientes:

-Nombre de cliente

-Tipo de cliente

-DNI/NIF

-Domicilio de facturación

-Forma de pago

-Persona de contacto

-Teléfono de contacto

-Correo de contacto

-Observaciones

**Reglas de negocio y pruebas de aceptación**

RN-01: Fecha de incidencia

**COMO** propietario,

**QUIERO** que la fecha introducida durante el proceso de registro de la incidencia debe ser igual o más antigua que la fecha actual,

**PARA** garantizar que la fecha introducida se corresponde a una válida.

RN-02: Tipo camión/mercancía

**COMO** propietario,

**QUIERO** que se utilice el camión oportuno dependiendo del tipo de mercancía que se tenga que transportar,

**PARA** el correcto transporte de esta.

RN-03: Empleados e incidencias

**COMO** propietario

**QUIERO** que se cumpla lo siguiente: Un empleado puede acumular hasta cierta cantidad de puntos. Las incidencias leves desaparecerán con el tiempo, pero una incidencia grave podrá ser motivo de despido,

**PARA** un correcto funcionamiento de la empresa.

RN-04: Revisiones

**COMO** propietario

**QUIERO** que cada mes pase un camión por revisión a no ser que éste esté averiado,

**PARA** el correcto funcionamiento de estos.

RN-05: Hora de llegada

**COMO** propietario,

**QUIERO** que la hora de llegada tiene que ser mayor que la hora de salida,

**PARA** garantizar una hora de llegada válida.

RN-06: Hora de llegada prevista

**COMO** propietario,

**QUIERO** que la hora de llegada prevista tiene que ser mayor que la hora de salida

**PARA** garantizar una hora de llegada prevista válida.

RN-07: Fecha de aviso

**COMO** propietario,

**QUIERO** que la fecha de aviso sea menor que la fecha de pedido

**PARA** garantizar una fecha de aviso válida.

RN-08: Fecha de revisión

**COMO** propietario,

**QUIERO** que la fecha de revisión sea menor que la fecha actual

**PARA** garantizar una fecha de revisión válida.

RN-09: Fecha disponible

**COMO** propietario,

**QUIERO** que la fecha de disponibilidad sea mayor que la fecha actual

**PARA** garantizar una fecha de disponibilidad válida.

**Requisitos funcionales**

RF-01: Consulta de puntos

**COMO** propietario,

**QUIERO** poder consultar la puntuación,

**PARA** saber de cuantos puntos dispone un empleado en concreto.

RF-02: Lista de puntos

**COMO** propietario,

**QUIERO** poder obtener una lista de puntuaciones,

**PARA** tener una lista de menos a mayos de los puntos acumulados

RF-03: Dar puntuación

**COMO** propietario,

**QUIERO** poder darle puntuación a un empleado,

**PARA** registrar de alguna manera algún problema causado por el mismo.

RF-04: Quitar puntuación

**COMO** propietario,

**QUIERO** poder quitarle puntuación a un empleado,

**PARA** zanjar de alguna manera el problema ya solucionado.

RF-05: Crear asignación

**COMO** propietario,

**QUIERO** poder asignar un conductor a un camión,

**PARA** poder asignarlo de forma manual.

RF-06: Eliminar asignación

**COMO** propietario,

**QUIERO** poder eliminar una asignación de un conductor a un camión,

**PARA** poder eliminar dicha asignación de forma manual.

RF-07: Conductor asignado

**COMO** propietario,

**QUIERO** que, si le doy una matrícula a la base de datos, me devuelva el DNI del conductor,

**PARA** saber qué conductor está asignado al camión.

RF-08: Camión asignado

**COMO** propietario,

**QUIERO** que, si le doy un DNI a la base de datos, me devuelva la matricula del camión,

**PARA** saber qué camión está asignado al conductor.

RF-09: Acceso horas

**COMO** propietario,

**QUIERO** poder consultar el número de horas que ha conducido un empleado introduciendo su DNI,

**PARA** saber cuántas horas ha conducido.

RF-10: Acceso sueldo

**COMO** propietario,

**QUIERO** tener la opción de consultar el sueldo de un empleado mediante su DNI,

**PARA** estar al tanto de los pagos y saber que se realizan correctamente.

RF-11: Conductores por sueldo

**COMO** propietario,

**QUIERO** poder obtener una lista ordenada de los sueldos de mis trabajadores,

**PARA** mayor comodidad a la hora de gestionarlos.

RF-12: Conductores por horas

**COMO** propietario,

**QUIERO** poder obtener una lista ordenada de las horas trabajadas por mis trabajadores,

**PARA** mayor comodidad a la hora de gestionar viajes.

RF-13: Camiones no disponibles

**COMO** propietario,

**QUIERO** una función que me proporcione la lista de camiones que no están disponibles y cuando lo estarán,

**PARA** saber la disponibilidad de camiones que tengo.

**Requisitos no funcionales**

RNF-01 Funcionamiento correcto

**COMO** propietario

**QUIERO** que se permita la visualización y el correcto funcionamiento en cualquier navegador

**PARA** que todo el mundo pueda disfrutar del sistema.

RNF-02 Disponibilidad

**COMO** propietario

**QUIERO** que el sistema disponga de una buena disponibilidad

**PARA** su correcto funcionamiento.

 RNF-03 Facilidad de uso

**COMO** propietario

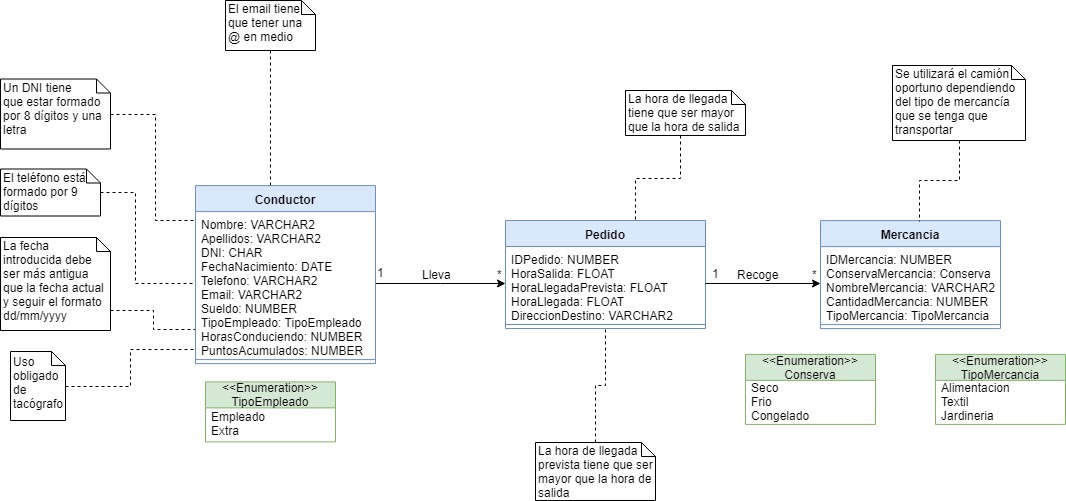
**QUIERO** que el sistema sea sencillo y cualquier persona sea capaz de usarlo

**PARA** una mayor comodidad y facilidad.

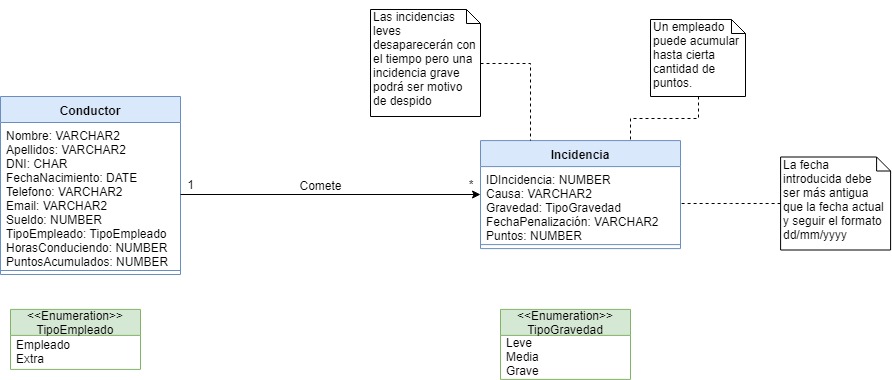
Modelado conceptual

**Diagramas de clases UML con restricciones**

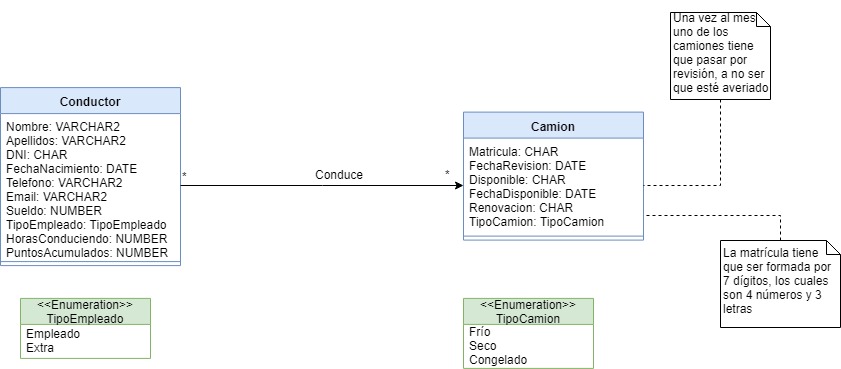
Pedidos



Incidencias



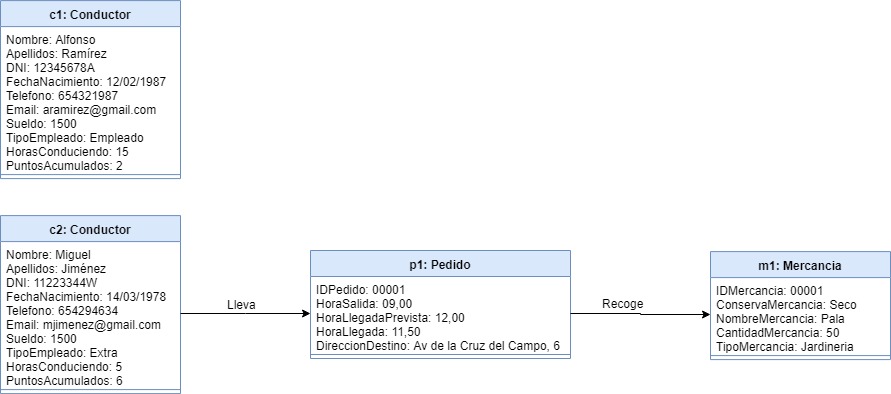
Asignación de camiones



**Escenarios de prueba con descripción textual y diagrama de objetos UML**

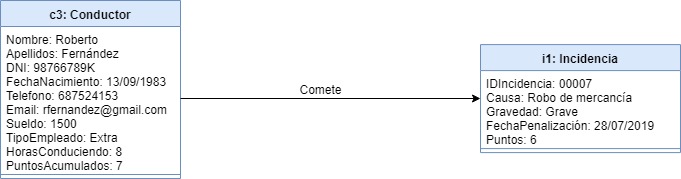
Prueba de pedidos

Aquí tenemos 2 conductores, c1 simplemente nos mostraría como quedaría el registro de un nuevo empleado en la empresa y c2 nos muestra a un conductor ya registrado que llevaría el pedido p1 cuya mercancía sería la m1.



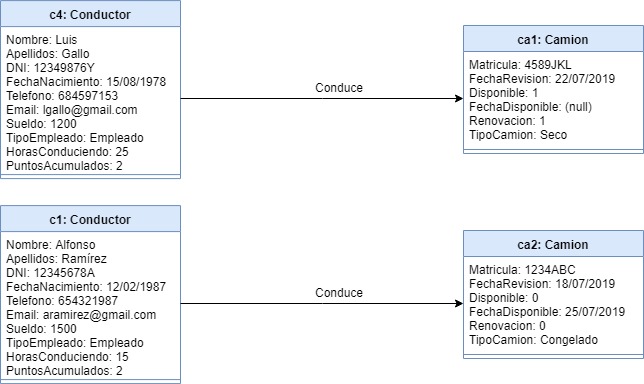
Prueba de incidencia

En este caso, se nos muestra al conductor c3, que ha cometido una incidencia grave (llamada i1) que consistiría en robo de mercancía.



Prueba de asignación de camiones

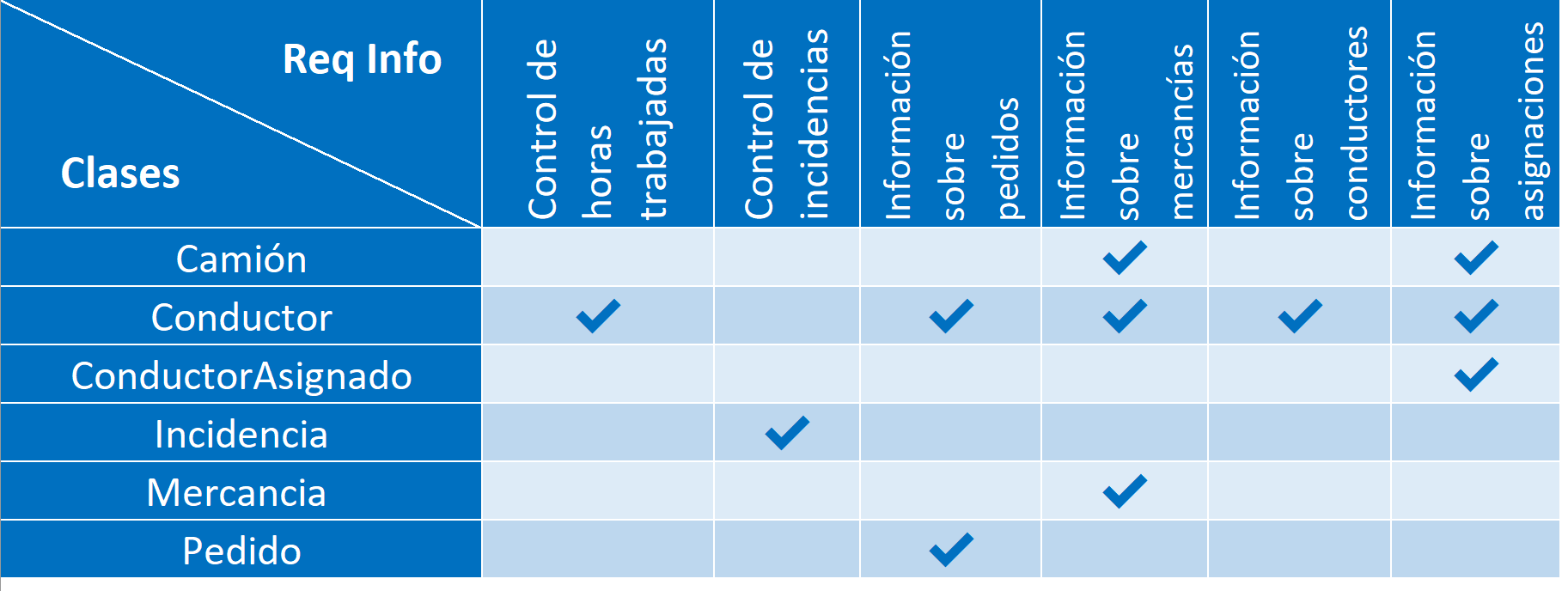
Como podemos observar, el conductor c4 tiene asignado el camión ca1, que está en perfecto estado de uso, sin embargo, como podemos ver en el c1, su camión (ca2) no está disponible y tal y como se puede ver, estará disponible en la fecha marcada.



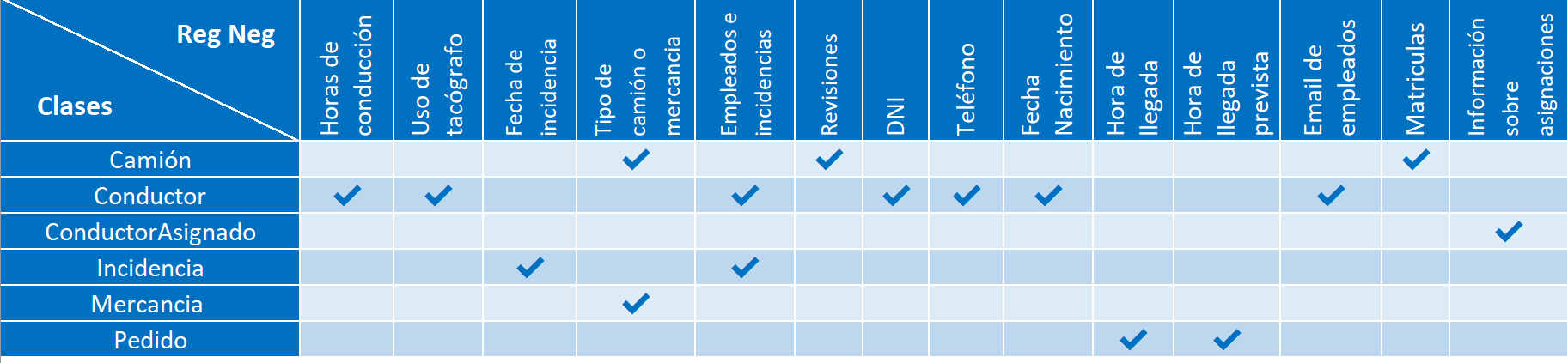
Matrices de trazabilidad

**Matrices de clase**

Clases → Requisitos de información

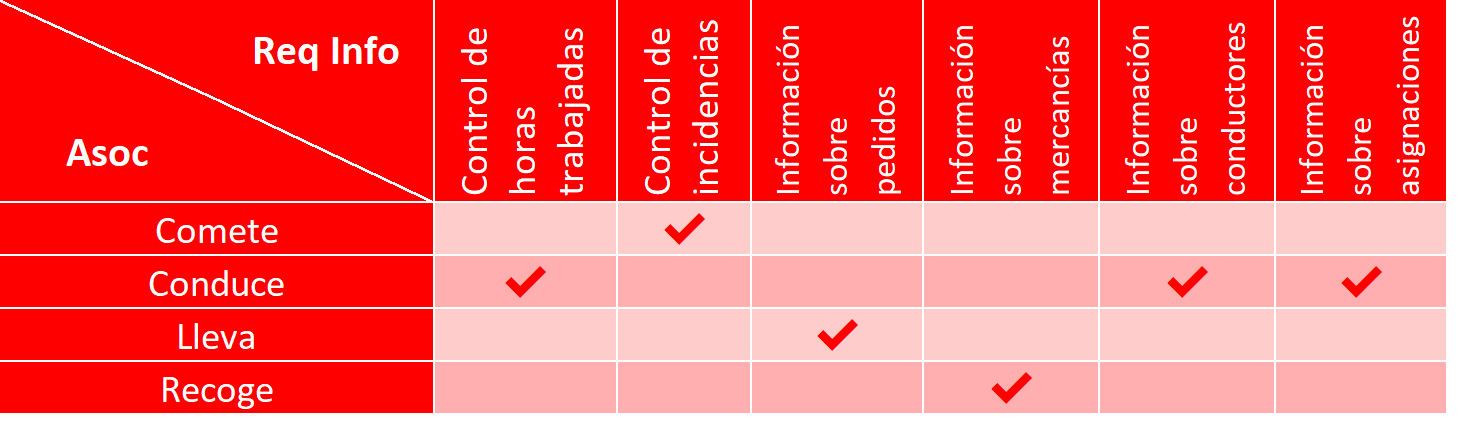


Clases → Reglas de negocio

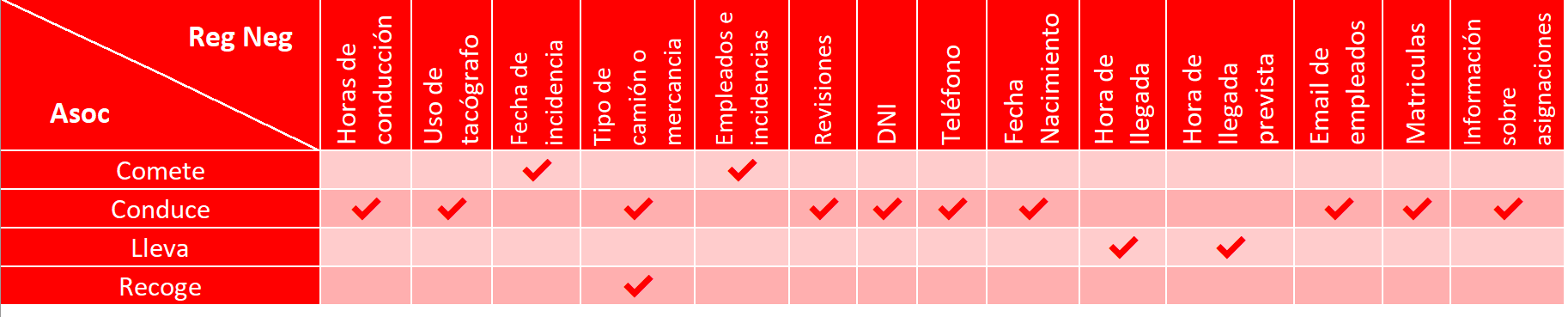


**Matrices de asociaciones**

Asociaciones → Requisitos de información

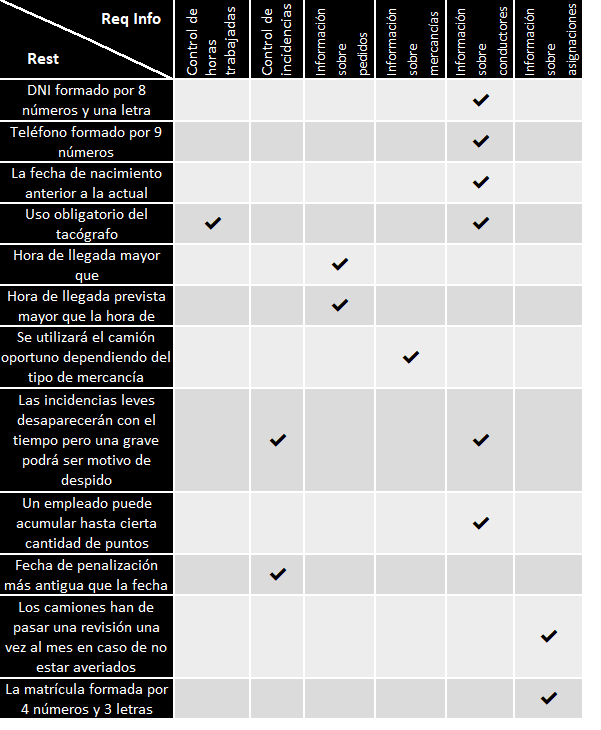


Asociaciones → Reglas de negocio

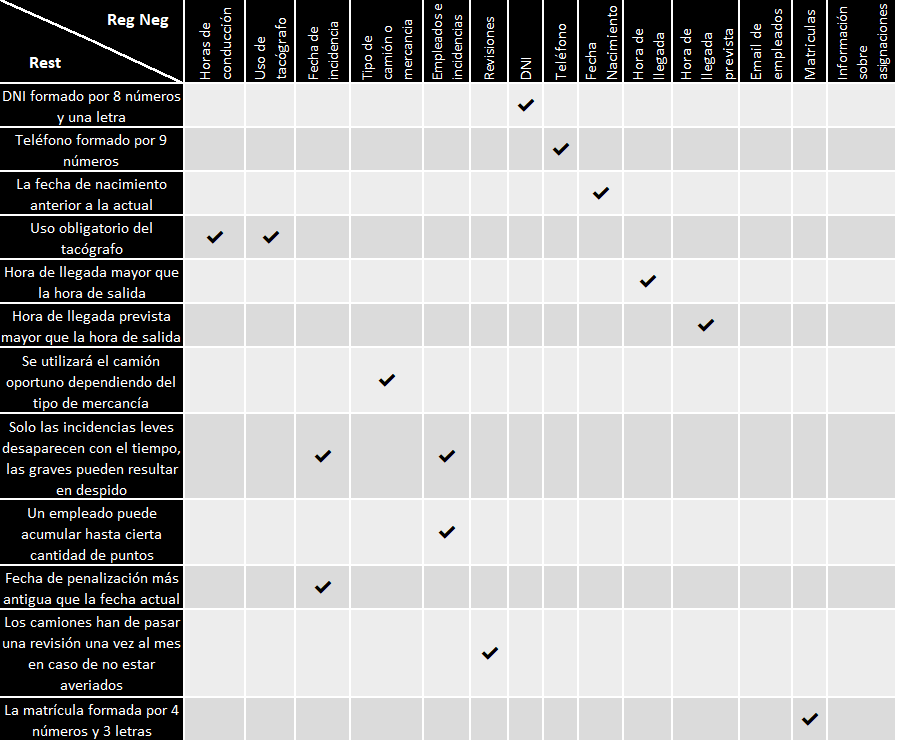


**Matrices de restricciones**

Restricciones → Requisitos de información



Restricciones → Reglas de negocio



Modelo relacional en 3FN



Modelo tecnológico

**Scripts de creación de base de datos**

/\* TABLAS \*/

CREATE TABLE Camion (

Matricula CHAR(7),

FechaRevision DATE,

Renovacion CHAR(1),

TipoCamion VARCHAR2(50),

Disponible CHAR(1),

FechaDisponible DATE

);

/

CREATE TABLE Conductor (

Nombre VARCHAR2(25),

Apellidos VARCHAR2(50),

DNI CHAR(9),

FechaNacimiento DATE,

Telefono NUMBER(9),

Email VARCHAR2(50),

Sueldo NUMBER,

TipoEmpleado VARCHAR2(50),

HorasConduciendo NUMBER,

PuntosAcumulados NUMBER

);

/

CREATE TABLE ConductorAsignado (

IDConductorAsignado NUMBER,

Conductor CHAR(9),

Camion CHAR(7)

);

/

CREATE TABLE Incidencia (

IDIncidencia NUMBER,

Causa VARCHAR2(50),

Gravedad VARCHAR2(50),

FechaPenalizacion DATE,

Puntos NUMBER,

Conductor CHAR(9)

);

/

CREATE TABLE Mercancia (

IDMercancia NUMBER,

ConservaMercancia VARCHAR2(50),

NombreMercancia VARCHAR2(50),

CantidadMercancia NUMBER,

TipoMercancia VARCHAR2(50),

Pedido NUMBER

);

/

CREATE TABLE Pedido (

IDPedido NUMBER,

HoraSalida FLOAT,

HoraLlegadaPrevista FLOAT,

HoraLlegada FLOAT,

DireccionDestino VARCHAR2(50),

Conductor CHAR(9)

);

/

/\*Enums\*/

ALTER TABLE Camion ADD CONSTRAINT CK\_TipoCamion CHECK (TipoCamion

IN('Seco','Frio','Congelado'));

/

ALTER TABLE Conductor ADD CONSTRAINT CK\_TipoEmpleado CHECK (TipoEmpleado

IN('Empleado','Extra'));

/

ALTER TABLE Incidencia ADD CONSTRAINT CK\_Gravedad CHECK (Gravedad IN('Leve','Media','Grave'));

/

ALTER TABLE Mercancia ADD CONSTRAINT CK\_ConservaMercancia CHECK (ConservaMercancia

IN('Seco','Frio','Congelado'));

/

ALTER TABLE Mercancia ADD CONSTRAINT CK\_TipoMercancia CHECK (TipoMercancia

IN('Alimentacion','Textil','Jardineria'));

/

/\*Booleans\*/

ALTER TABLE Camion ADD CONSTRAINT CK\_Disponible CHECK (Disponible IN('0','1'));

/

ALTER TABLE Camion ADD CONSTRAINT CK\_Renovacion CHECK (Renovacion IN('0','1'));

/

/\*Not nulls\*/

ALTER TABLE Camion MODIFY (Matricula NOT NULL);

ALTER TABLE Camion MODIFY (FechaRevision NOT NULL);

ALTER TABLE Camion MODIFY (Disponible NOT NULL);

ALTER TABLE Camion MODIFY (Renovacion NOT NULL);

ALTER TABLE Camion MODIFY (TipoCamion NOT NULL);

/

ALTER TABLE Conductor MODIFY (Nombre NOT NULL);

ALTER TABLE Conductor MODIFY (Apellidos NOT NULL);

ALTER TABLE Conductor MODIFY (DNI NOT NULL);

ALTER TABLE Conductor MODIFY (FechaNacimiento NOT NULL);

ALTER TABLE Conductor MODIFY (Telefono NOT NULL);

ALTER TABLE Conductor MODIFY (Email NOT NULL);

ALTER TABLE Conductor MODIFY (Sueldo NOT NULL);

ALTER TABLE Conductor MODIFY (TipoEmpleado NOT NULL);

ALTER TABLE Conductor MODIFY (HorasConduciendo NOT NULL);

ALTER TABLE Conductor MODIFY (PuntosAcumulados NOT NULL);

/

ALTER TABLE ConductorAsignado MODIFY (IDConductorAsignado NOT NULL);

/

ALTER TABLE Incidencia MODIFY (IDIncidencia NOT NULL);

ALTER TABLE Incidencia MODIFY (Causa NOT NULL);

ALTER TABLE Incidencia MODIFY (Gravedad NOT NULL);

ALTER TABLE Incidencia MODIFY (FechaPenalizacion NOT NULL);

ALTER TABLE Incidencia MODIFY (Puntos NOT NULL);

/

ALTER TABLE Mercancia MODIFY (IDMercancia NOT NULL);

ALTER TABLE Mercancia MODIFY (ConservaMercancia NOT NULL);

ALTER TABLE Mercancia MODIFY (NombreMercancia NOT NULL);

ALTER TABLE Mercancia MODIFY (CantidadMercancia NOT NULL);

ALTER TABLE Mercancia MODIFY (TipoMercancia NOT NULL);

/

ALTER TABLE Pedido MODIFY (IDPedido NOT NULL);

ALTER TABLE Pedido MODIFY (HoraSalida NOT NULL);

ALTER TABLE Pedido MODIFY (HoraLlegadaPrevista NOT NULL);

ALTER TABLE Pedido MODIFY (HoraLlegada NOT NULL);

ALTER TABLE Pedido MODIFY (DireccionDestino NOT NULL);

/

/\*Primary keys\*/

ALTER TABLE Camion ADD CONSTRAINT PK\_Camion PRIMARY KEY (Matricula);

/

ALTER TABLE Conductor ADD CONSTRAINT PK\_Conductor PRIMARY KEY (DNI);

/

ALTER TABLE ConductorAsignado ADD CONSTRAINT PK\_ConductorAsignado PRIMARY KEY

(IDConductorAsignado);

/

ALTER TABLE Incidencia ADD CONSTRAINT PK\_Incidencia PRIMARY KEY (IDIncidencia);

/

ALTER TABLE Mercancia ADD CONSTRAINT PK\_Mercancia PRIMARY KEY (IDMercancia);

/

ALTER TABLE Pedido ADD CONSTRAINT PK\_Pedido PRIMARY KEY (IDPedido);

/

/\*Foreign Keys\*/

ALTER TABLE ConductorAsignado ADD CONSTRAINT FK\_ConductorAsignado\_Conductor FOREIGN KEY

(Conductor)

REFERENCES Conductor (DNI) ON DELETE SET NULL;

/

ALTER TABLE ConductorAsignado ADD CONSTRAINT FK\_ConductorAsignado\_Camion FOREIGN KEY (Camion)

REFERENCES Camion (MATRICULA) ON DELETE SET NULL;

/

ALTER TABLE Incidencia ADD CONSTRAINT FK\_Incidencia\_Conductor FOREIGN KEY (Conductor)

REFERENCES Conductor (DNI) ON DELETE SET NULL;

/

ALTER TABLE Mercancia ADD CONSTRAINT FK\_Mercancia\_Pedido FOREIGN KEY (Pedido)

REFERENCES Pedido (IDPedido) ON DELETE SET NULL;

/

ALTER TABLE Pedido ADD CONSTRAINT FK\_Pedido\_Conductor FOREIGN KEY (Conductor)

REFERENCES Conductor (DNI) ON DELETE SET NULL;

/

/\*Alternative Keys\*/

ALTER TABLE Conductor ADD CONSTRAINT AK\_Conductor UNIQUE (Email);

/

ALTER TABLE ConductorAsignado ADD CONSTRAINT AK\_ConductorAsignado UNIQUE (Conductor, Camion);

/

ALTER TABLE Mercancia ADD CONSTRAINT AK\_Mercancia UNIQUE (NombreMercancia);

/

/\*Restricciones\*/

ALTER TABLE Camion ADD CONSTRAINT CK\_Matricula\_Camion CHECK (REGEXP\_LIKE(Matricula,

'[0-9][0-9][0-9][0-9][A-Z][A-Z][A-Z]'));

/

ALTER TABLE Conductor ADD CONSTRAINT CK\_DNI\_Conductor CHECK (REGEXP\_LIKE(DNI,

'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][A-Z]'));

/

ALTER TABLE Conductor ADD CONSTRAINT CK\_Telefono\_Conductor CHECK (REGEXP\_LIKE(Telefono,

'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'));

/

/\*Secuencias\*/

CREATE SEQUENCE SEC\_ConductorAsignado

INCREMENT BY 1

START WITH 1

MAXVALUE 99999;

/

CREATE SEQUENCE SEC\_Incidencia

INCREMENT BY 1

START WITH 1

MAXVALUE 99999;

/

CREATE SEQUENCE SEC\_Mercancia

INCREMENT BY 1

START WITH 1

MAXVALUE 99999;

/

CREATE SEQUENCE SEC\_Pedido

INCREMENT BY 1

START WITH 1

MAXVALUE 99999;

/

/\*Triggers asociados a secuencias\*/

CREATE OR REPLACE TRIGGER TR\_SEC\_ConductorAsignado

BEFORE INSERT ON ConductorAsignado

FOR EACH ROW

DECLARE

valorSec NUMBER := 0;

BEGIN

SELECT SEC\_ConductorAsignado.NEXTVAL INTO valorSec FROM DUAL;

:NEW.IDConductorAsignado := valorSec;

END;

/

CREATE OR REPLACE TRIGGER TR\_SEC\_Incidencia

BEFORE INSERT ON Incidencia

FOR EACH ROW

DECLARE

valorSec NUMBER := 0;

BEGIN

SELECT SEC\_Incidencia.NEXTVAL INTO valorSec FROM DUAL;

:NEW.IDIncidencia := valorSec;

END;

/

CREATE OR REPLACE TRIGGER TR\_SEC\_Mercancia

BEFORE INSERT ON Mercancia

FOR EACH ROW

DECLARE

valorSec NUMBER := 0;

BEGIN

SELECT SEC\_Mercancia.NEXTVAL INTO valorSec FROM DUAL;

:NEW.IDMercancia := valorSec;

END;

/

CREATE OR REPLACE TRIGGER TR\_SEC\_Pedido

BEFORE INSERT ON Pedido

FOR EACH ROW

DECLARE

valorSec NUMBER := 0;

BEGIN

SELECT SEC\_Pedido.NEXTVAL INTO valorSec FROM DUAL;

:NEW.IDPedido := valorSec;

END;

/

/\*Procedimientos y funciones\*/

/\*RF-21\*/

CREATE OR REPLACE FUNCTION FN\_CONSULTA\_PUNTOS

(v\_DNI IN Conductor.DNI%TYPE)

RETURN VARCHAR2

IS v\_PuntosAcumulados Conductor.PuntosAcumulados%TYPE;

BEGIN

SELECT PuntosAcumulados INTO v\_PuntosAcumulados

FROM Conductor

WHERE DNI = v\_DNI;

RETURN v\_PuntosAcumulados;

END;

/

/\*RF-22\*/

CREATE OR REPLACE PROCEDURE PR\_Conductores\_por\_Puntos

IS

CURSOR C IS

SELECT DNI, PuntosAcumulados FROM Conductor ORDER BY

PuntosAcumulados;

v\_DNI Conductor.DNI%TYPE;

v\_PuntosAcumulados Conductor.PuntosAcumulados%TYPE;

BEGIN

OPEN C;

FETCH C INTO v\_DNI, v\_PuntosAcumulados;

DBMS\_OUTPUT.PUT\_LINE(RPAD('DNI:' , 25)|| RPAD('PuntosAcumulados:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 50 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_DNI , 25)|| RPAD(v\_PuntosAcumulados , 25));

FETCH C INTO v\_DNI, v\_PuntosAcumulados;

END LOOP;

CLOSE C;

END;

/

/\*RF-23\*/

CREATE OR REPLACE PROCEDURE PR\_Dar\_Puntuacion (v\_DNI IN Conductor.DNI%TYPE,

v\_PuntosAcumulados IN Conductor.PuntosAcumulados%TYPE, puntosObtenidos NUMBER)

IS

BEGIN

UPDATE Conductor SET PUNTOSACUMULADOS = v\_PuntosAcumulados + puntosObtenidos WHERE DNI =

v\_DNI;

COMMIT;

END;

/

/\*RF-24\*/

CREATE OR REPLACE PROCEDURE PR\_Quitar\_Puntuacion (v\_DNI IN Conductor.DNI%TYPE,

v\_PuntosAcumulados IN Conductor.PuntosAcumulados%TYPE, puntosQuitados NUMBER)

IS

BEGIN

UPDATE Conductor SET PUNTOSACUMULADOS = v\_PuntosAcumulados - puntosQuitados WHERE DNI = v\_DNI;

COMMIT;

END;

/

/\*RF-25\*/

CREATE OR REPLACE PROCEDURE PR\_CrearAsignacion(v\_Conductor IN

CONDUCTORASIGNADO.CONDUCTOR%TYPE , v\_Camion IN CONDUCTORASIGNADO.CAMION%TYPE)

IS

BEGIN

INSERT INTO CONDUCTORASIGNADO (CONDUCTOR , CAMION) VALUES (v\_Conductor , v\_Camion);

COMMIT;

END;

/

/\*RF-26\*/

CREATE OR REPLACE PROCEDURE PR\_EliminarAsignacion(v\_Conductor IN

CONDUCTORASIGNADO.CONDUCTOR%TYPE , v\_Camion IN CONDUCTORASIGNADO.CAMION%TYPE)

IS

BEGIN

DELETE FROM CONDUCTORASIGNADO WHERE CONDUCTOR = v\_Conductor AND CAMION = v\_Camion;

COMMIT;

END;

/

/\*RF-27\*/

create or replace FUNCTION FN\_Consulta\_Conductor (v\_Camion IN ConductorAsignado.Camion%type)

RETURN VARCHAR2

IS v\_Conductor ConductorAsignado.Conductor%TYPE;

BEGIN

SELECT Conductor INTO v\_Conductor

FROM ConductorAsignado

WHERE Camion = v\_Camion;

RETURN v\_Conductor;

END;

/

/\*RF-28\*/

create or replace FUNCTION FN\_Consulta\_Camiones (v\_Conductor IN

ConductorAsignado.Conductor%TYPE)

RETURN VARCHAR2

IS v\_Camion ConductorAsignado.Camion%TYPE;

BEGIN

SELECT Camion INTO v\_Camion

FROM ConductorAsignado

WHERE Conductor = v\_Conductor;

RETURN v\_Camion;

END;

/

/\*Triggers\*/

CREATE OR REPLACE TRIGGER TR\_Fecha\_Nacimiento

AFTER INSERT OR UPDATE OF FechaNacimiento ON Conductor

FOR EACH ROW

DECLARE

v\_FechaNacimiento DATE := :NEW.FechaNacimiento;

BEGIN

IF v\_FechaNacimiento > SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-00001, 'La fecha de nacimiento ha de ser anterior o igual a la

actual.');

END IF;

END;

/

CREATE OR REPLACE TRIGGER TR\_Email

AFTER INSERT OR UPDATE OF Email ON Conductor

FOR EACH ROW

DECLARE

v\_Email VARCHAR2(40) := :NEW.Email;

BEGIN

IF v\_Email NOT LIKE '%@%' THEN

RAISE\_APPLICATION\_ERROR(-00002, 'El email ha de tener un formato determinado.');

END IF;

END;

/

CREATE OR REPLACE TRIGGER TR\_HoraLlegada\_HoraSalida

AFTER INSERT OR UPDATE OF HoraLlegada ON Pedido

FOR EACH ROW

DECLARE

v\_HoraLlegada FLOAT := :NEW.HoraLlegada;

v\_HoraSalida FLOAT := :NEW.HoraSalida;

BEGIN

IF v\_HoraLlegada < v\_HoraSalida THEN

RAISE\_APPLICATION\_ERROR(-00003, 'La hora de llegada ha de ser superior a la hora de salida.');

END IF;

END;

/

CREATE OR REPLACE TRIGGER TR\_HoraLlegadaP\_HoraSalida

AFTER INSERT OR UPDATE OF HoraLlegadaPrevista ON Pedido

FOR EACH ROW

DECLARE

v\_HoraLlegadaPrevista FLOAT := :NEW.HoraLlegadaPrevista;

v\_HoraSalida FLOAT := :NEW.HoraSalida;

BEGIN

IF v\_HoraLlegadaPrevista < v\_HoraSalida THEN

RAISE\_APPLICATION\_ERROR(-00004, 'La hora de llegada prevista ha de ser superior a la hora de

salida.');

END IF;

END;

/

CREATE OR REPLACE TRIGGER TR\_Fecha\_Incidencia

AFTER INSERT OR UPDATE OF FechaPenalizacion ON Incidencia

FOR EACH ROW

DECLARE

v\_FechaPenalizacion DATE := :NEW.FechaPenalizacion;

BEGIN

IF v\_FechaPenalizacion > SYSDATE THEN

RAISE\_APPLICATION\_ERROR(-00005, 'La fecha de la incidencia ha de ser anterior o igual a la

actual.');

END IF;

END;

/

/\*Funcion auxiliar\*/

CREATE OR REPLACE FUNCTION ASSERT\_EQUALS (v\_Salida BOOLEAN, salidaEsperada BOOLEAN)

RETURN VARCHAR2

IS

BEGIN

IF v\_Salida = salidaEsperada THEN

RETURN 'EXITO';

ELSE

RETURN 'FALLO';

END IF;

END;

/

/\*Cabeceras\*/

CREATE OR REPLACE PACKAGE PCK\_Camion

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_Matricula IN Camion.Matricula%TYPE,

v\_FechaRevision IN Camion.FechaRevision%TYPE, v\_Renovacion IN Camion.Renovacion%TYPE,

v\_TipoCamion IN Camion.TipoCamion%TYPE, v\_Disponible IN Camion.Disponible%TYPE,

v\_FechaDisponible IN Camion.FechaDisponible%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_Matricula IN Camion.Matricula%TYPE,

v\_FechaRevision IN Camion.FechaRevision%TYPE, v\_Renovacion IN Camion.Renovacion%TYPE,

v\_TipoCamion IN Camion.TipoCamion%TYPE, v\_Disponible IN Camion.Disponible%TYPE,

v\_FechaDisponible IN Camion.FechaDisponible%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_Matricula IN Camion.Matricula%TYPE,

salidaEsperada BOOLEAN);

END;

/

CREATE OR REPLACE PACKAGE PCK\_Conductor

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_nombre IN Conductor.Nombre%TYPE, v\_Apellidos IN

Conductor.Apellidos%TYPE, v\_DNI IN Conductor.DNI%TYPE,

v\_FechaNacimiento IN Conductor.FechaNacimiento%TYPE, v\_Telefono IN Conductor.Telefono%TYPE,

v\_Email IN Conductor.Email%TYPE, v\_Sueldo IN Conductor.SUELDO%TYPE,

v\_TipoEmpleado IN Conductor.TipoEmpleado%TYPE,

v\_HorasConduciendo IN Conductor.HorasConduciendo%TYPE, v\_PuntosAcumulados IN

Conductor.PuntosAcumulados%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_nombre IN Conductor.Nombre%TYPE, v\_Apellidos

IN Conductor.Apellidos%TYPE, v\_DNI IN Conductor.DNI%TYPE,

v\_FechaNacimiento IN Conductor.FechaNacimiento%TYPE, v\_Telefono IN Conductor.Telefono%TYPE,

v\_Email IN Conductor.Email%TYPE, v\_Sueldo IN Conductor.SUELDO%TYPE, v\_TipoEmpleado IN

Conductor.TipoEmpleado%TYPE,

v\_HorasConduciendo IN Conductor.HorasConduciendo%TYPE, v\_PuntosAcumulados IN

Conductor.PuntosAcumulados%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_DNI IN Conductor.DNI%TYPE, salidaEsperada

BOOLEAN);

END;

/

CREATE OR REPLACE PACKAGE PCK\_ConductorAsignado

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_Conductor IN ConductorAsignado.Conductor%TYPE,

v\_Camion IN ConductorAsignado.Camion%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDConductorAsignado IN

ConductorAsignado.IDConductorAsignado%TYPE, salidaEsperada BOOLEAN);

END;

/

CREATE OR REPLACE PACKAGE PCK\_Incidencia

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_Causa IN Incidencia.Causa%TYPE, v\_Gravedad IN

Incidencia.Gravedad%TYPE,

v\_FechaPenalizacion IN Incidencia.FechaPenalizacion%TYPE, v\_Puntos IN

Incidencia.Puntos%TYPE, v\_Conductor IN Incidencia.Conductor%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_IDIncidencia IN Incidencia.IDIncidencia%TYPE,

v\_Causa IN Incidencia.Causa%TYPE, v\_Gravedad IN Incidencia.Gravedad%TYPE,

v\_FechaPenalizacion IN Incidencia.FechaPenalizacion%TYPE, v\_Puntos IN

Incidencia.Puntos%TYPE, v\_Conductor IN Incidencia.Conductor%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDIncidencia IN Incidencia.IDIncidencia%TYPE,

salidaEsperada BOOLEAN);

END;

/

CREATE OR REPLACE PACKAGE PCK\_Mercancia

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_ConservaMercancia IN

Mercancia.ConservaMercancia%TYPE, v\_NombreMercancia IN Mercancia.NombreMercancia%TYPE,

v\_CantidadMercancia IN Mercancia.CantidadMercancia%TYPE, v\_TipoMercancia IN

Mercancia.TipoMercancia%TYPE, v\_Pedido IN Mercancia.Pedido%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_IDMercancia IN Mercancia.IDMercancia%TYPE,

v\_ConservaMercancia IN Mercancia.ConservaMercancia%TYPE, v\_NombreMercancia IN

Mercancia.NombreMercancia%TYPE,

v\_CantidadMercancia IN Mercancia.CantidadMercancia%TYPE, v\_TipoMercancia IN

Mercancia.TipoMercancia%TYPE, v\_Pedido IN Mercancia.Pedido%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDMercancia IN Mercancia.IDMercancia%TYPE,

salidaEsperada BOOLEAN);

END;

/

CREATE OR REPLACE PACKAGE PCK\_Pedido

IS

PROCEDURE Inicializar;

PROCEDURE Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2, v\_HoraSalida IN Pedido.HoraSalida%TYPE,

v\_HoraLlegadaPrevista IN Pedido.HoraLlegadaPrevista%TYPE,

v\_HoraLlegada IN Pedido.HoraLlegada%TYPE, v\_DireccionDestino IN

Pedido.DireccionDestino%TYPE, v\_Conductor IN Pedido.Conductor%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_IDPedido IN Pedido.IDPedido%TYPE,

v\_HoraSalida IN Pedido.HoraSalida%TYPE, v\_HoraLlegadaPrevista IN

Pedido.HoraLlegadaPrevista%TYPE,

v\_HoraLlegada IN Pedido.HoraLlegada%TYPE, v\_DireccionDestino IN

Pedido.DireccionDestino%TYPE, v\_Conductor IN Pedido.Conductor%TYPE, salidaEsperada BOOLEAN);

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDPedido IN Pedido.IDPedido%TYPE,

salidaEsperada BOOLEAN);

END;

/

/\*Cuerpos\*/

--Tabla Camion

CREATE OR REPLACE PACKAGE BODY PCK\_Camion

IS

CURSOR C IS

SELECT \* FROM Camion ;

v\_salida BOOLEAN := TRUE;

v\_Camion Camion%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM Camion;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_Camion;

DBMS\_OUTPUT.PUT\_LINE( RPAD('Matricula:' , 25)||RPAD('FechaRevision:' , 25)||

RPAD('Renovacion:' , 25)|| RPAD('TipoCamion:' , 25)||RPAD('Disponible:' , 25)||

RPAD('FechaDisponible:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 150 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_Camion.Matricula , 25)||RPAD(v\_Camion.FechaRevision , 25)||

RPAD(v\_Camion.Renovacion , 25)||

RPAD(v\_Camion.TipoCamion , 25)|| RPAD(v\_Camion.Disponible , 25)||

RPAD(v\_Camion.FechaDisponible , 25));

FETCH C INTO v\_Camion;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,

v\_Matricula IN Camion.Matricula%TYPE,

v\_FechaRevision IN Camion.FechaRevision%TYPE,

v\_Renovacion IN Camion.Renovacion%TYPE,

v\_TipoCamion IN Camion.TipoCamion%TYPE,

v\_Disponible IN Camion.Disponible%TYPE,

v\_FechaDisponible IN Camion.FechaDisponible%TYPE,

salidaEsperada BOOLEAN)

IS

BEGIN

INSERT INTO Camion (Matricula, FechaRevision , Renovacion , TipoCamion, Disponible,

FechaDisponible) VALUES (v\_Matricula, v\_FechaRevision , v\_Renovacion , v\_TipoCamion,

v\_Disponible , v\_FechaDisponible );

SELECT \* INTO v\_Camion FROM Camion WHERE Matricula = v\_Matricula;

IF v\_Camion.FechaRevision != v\_FechaRevision AND

v\_Camion.Renovacion != v\_Renovacion AND

v\_Camion.TipoCamion != v\_TipoCamion AND

v\_Camion.Disponible != v\_Disponible AND

v\_Camion.FechaDisponible != v\_FechaDisponible

THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2,v\_Matricula IN Camion.Matricula%TYPE,

v\_FechaRevision IN Camion.FechaRevision%TYPE,

v\_Renovacion IN Camion.Renovacion%TYPE, v\_TipoCamion IN Camion.TipoCamion%TYPE,v\_Disponible

IN Camion.Disponible%TYPE, v\_FechaDisponible IN Camion.FechaDisponible%TYPE, salidaEsperada

BOOLEAN)

IS

BEGIN

UPDATE Camion SET FechaRevision = v\_FechaRevision , Renovacion = v\_Renovacion , TipoCamion

= v\_TipoCamion, Disponible = v\_Disponible , FechaDisponible = v\_FechaDisponible WHERE

Matricula= v\_Matricula;

SELECT \* INTO v\_Camion FROM Camion WHERE Matricula= v\_Matricula;

IF v\_Camion.FechaRevision != v\_FechaRevision AND

v\_Camion.Renovacion != v\_Renovacion AND

v\_Camion.TipoCamion != v\_TipoCamion AND

v\_Camion.Disponible != v\_Disponible AND

v\_Camion.FechaDisponible != v\_FechaDisponible

THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_Matricula IN Camion.Matricula%TYPE,

salidaEsperada BOOLEAN)

IS

v\_NumCamion NUMBER := 0;

BEGIN

DELETE FROM Camion WHERE Matricula = v\_Matricula;

IF v\_NumCamion != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

--Tabla Conductor--

CREATE OR REPLACE PACKAGE BODY PCK\_Conductor

IS

CURSOR C IS

SELECT \* FROM CONDUCTOR ;

v\_salida BOOLEAN := TRUE;

v\_Conductor CONDUCTOR%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM CONDUCTOR;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_Conductor;

DBMS\_OUTPUT.PUT\_LINE(RPAD('Nombre:' , 25)||RPAD('Apellidos:' , 25)|| RPAD('DNI:' , 25)||

RPAD('FechaNacimiento:' , 25)|| RPAD('Telefono:' , 25)||

RPAD('Email:' , 25)|| RPAD('Sueldo:' , 25)||RPAD('TipoEmpleado:' , 25)||

RPAD('HorasConduciendo:' , 25)|| RPAD('PuntosAcumulados:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 250 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_Conductor.Nombre , 25)|| RPAD(v\_Conductor.Apellidos , 25)||

RPAD(v\_Conductor.DNI , 25)|| RPAD(v\_Conductor.FechaNacimiento , 25)||

RPAD(v\_Conductor.Telefono , 25)|| RPAD(v\_Conductor.Email , 25)|| RPAD(v\_Conductor.Sueldo ,

25)|| RPAD(v\_Conductor.TipoEmpleado , 25)|| RPAD(v\_Conductor.HorasConduciendo , 25)||

RPAD(v\_Conductor.PuntosAcumulados , 25));

FETCH C INTO v\_Conductor;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,

v\_Nombre IN Conductor.NOMBRE%TYPE,

v\_Apellidos IN Conductor.Apellidos%TYPE,

v\_DNI IN Conductor.DNI%TYPE,

v\_FechaNacimiento IN Conductor.FechaNacimiento%TYPE,

v\_Telefono IN Conductor.Telefono%TYPE,

v\_Email IN Conductor.Email%TYPE,

v\_Sueldo IN Conductor.Sueldo%TYPE,

v\_TipoEmpleado IN Conductor.TipoEmpleado%TYPE,

v\_HorasConduciendo IN Conductor.HorasConduciendo%TYPE,

v\_PuntosAcumulados IN Conductor.PuntosAcumulados%TYPE,

salidaEsperada BOOLEAN)

IS

BEGIN

INSERT INTO Conductor (Nombre , Apellidos, DNI, FechaNacimiento , Telefono , Email , Sueldo

, TipoEmpleado , HorasConduciendo, PuntosAcumulados) VALUES (v\_Nombre , v\_Apellidos , v\_DNI

, v\_FechaNacimiento

, v\_Telefono , v\_Email , v\_Sueldo , v\_TipoEmpleado , v\_HorasConduciendo , v\_PuntosAcumulados);

SELECT \* INTO v\_Conductor FROM Conductor WHERE DNI = v\_DNI;

IF v\_Conductor.Nombre != v\_Nombre AND

v\_Conductor.Apellidos != v\_Apellidos AND

v\_Conductor.DNI != v\_DNI AND

v\_Conductor.FechaNacimiento != v\_FechaNacimiento AND

v\_Conductor.Telefono != v\_Telefono AND

v\_Conductor.Email != v\_Email AND

v\_Conductor.Sueldo != v\_Sueldo AND

v\_Conductor.TipoEmpleado != v\_TipoEmpleado AND

v\_Conductor.HorasConduciendo != v\_HorasConduciendo AND

v\_Conductor.PuntosAcumulados != v\_PuntosAcumulados

THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2, v\_nombre IN Conductor.NOMBRE%TYPE, v\_Apellidos

IN Conductor.Apellidos%TYPE, v\_DNI IN Conductor.DNI%TYPE,

v\_FechaNacimiento IN Conductor.FechaNacimiento%TYPE, v\_Telefono IN Conductor.Telefono%TYPE,

v\_Email IN Conductor.Email%TYPE, v\_Sueldo IN Conductor.Sueldo%TYPE,

v\_TipoEmpleado IN Conductor.TipoEmpleado%TYPE,

v\_HorasConduciendo IN Conductor.HorasConduciendo%TYPE, v\_PuntosAcumulados IN

Conductor.PuntosAcumulados%TYPE, salidaEsperada BOOLEAN)

IS

BEGIN

UPDATE Conductor SET Nombre = v\_Nombre , Apellidos = v\_Apellidos , FechaNacimiento =

v\_FechaNacimiento , Telefono = v\_Telefono , Email = v\_Email , Sueldo = v\_Sueldo ,

TipoEmpleado = v\_TipoEmpleado ,

HorasConduciendo = v\_HorasConduciendo , PuntosAcumulados = v\_PuntosAcumulados WHERE DNI=

v\_DNI;

SELECT \* INTO v\_Conductor FROM Conductor WHERE DNI= v\_DNI;

IF v\_Conductor.Nombre != v\_Nombre AND

v\_Conductor.Apellidos != v\_Apellidos AND

v\_Conductor.FechaNacimiento != v\_FechaNacimiento AND

v\_Conductor.Telefono != v\_Telefono AND

v\_Conductor.Email != v\_Email AND

v\_Conductor.Sueldo != v\_Sueldo AND

v\_Conductor.TipoEmpleado != v\_TipoEmpleado AND

v\_Conductor.HorasConduciendo != v\_HorasConduciendo AND

v\_Conductor.PuntosAcumulados != v\_PuntosAcumulados

THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_DNI IN Conductor.DNI%TYPE, salidaEsperada

BOOLEAN)

IS

v\_NumConductor NUMBER := 0;

BEGIN

DELETE FROM Conductor WHERE DNI = v\_DNI;

IF v\_NumConductor != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

--Tabla ConductorAsignado

CREATE OR REPLACE PACKAGE BODY PCK\_ConductorAsignado

IS

CURSOR C IS

SELECT \* FROM ConductorAsignado ;

v\_salida BOOLEAN := TRUE;

v\_ConductorAsignado ConductorAsignado%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM ConductorAsignado;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_ConductorAsignado;

DBMS\_OUTPUT.PUT\_LINE( RPAD('IDConductorAsignado:' , 25)||RPAD('Conductor:' ,

25)||RPAD('Camion:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 75 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_ConductorAsignado.IDConductorAsignado ,

25)||RPAD(v\_ConductorAsignado.Conductor , 25)|| RPAD(v\_ConductorAsignado.Camion , 25));

FETCH C INTO v\_ConductorAsignado;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,

v\_Conductor IN ConductorAsignado.Conductor%TYPE,

v\_Camion IN ConductorAsignado.Camion%TYPE,

salidaEsperada BOOLEAN)

IS

v\_IDConductorAsignado ConductorAsignado.IDConductorAsignado%TYPE;

BEGIN

INSERT INTO ConductorAsignado (Conductor , Camion) VALUES (v\_Conductor , v\_Camion);

v\_IDConductorAsignado := SEC\_ConductorAsignado.CURRVAL;

SELECT \* INTO v\_ConductorAsignado FROM ConductorAsignado WHERE IDConductorAsignado =

v\_IDConductorAsignado;

IF v\_ConductorAsignado.Conductor != v\_Conductor AND v\_ConductorAsignado.Camion != v\_Camion

THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDConductorAsignado IN

ConductorAsignado.IDConductorAsignado%TYPE, salidaEsperada BOOLEAN)

IS

v\_NumConductorAsignado NUMBER := 0;

BEGIN

DELETE FROM ConductorAsignado WHERE IDConductorAsignado = v\_IDConductorAsignado;

SELECT COUNT(\*) INTO v\_NumConductorAsignado FROM ConductorAsignado WHERE

v\_IDConductorAsignado = v\_IDConductorAsignado;

IF v\_NumConductorAsignado != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

--Tabla Incidencia

CREATE OR REPLACE PACKAGE BODY PCK\_Incidencia

IS

CURSOR C IS

SELECT \* FROM Incidencia ;

v\_salida BOOLEAN := TRUE;

v\_Incidencia Incidencia%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM Incidencia;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_Incidencia;

DBMS\_OUTPUT.PUT\_LINE(RPAD('IDIncidencia:' , 25)||RPAD('Causa:' , 50)||RPAD('Gravedad:' ,

25)|| RPAD('FechaPenalizacion:' , 25)|| RPAD('Puntos:' , 25)|| RPAD('Conductor:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 175 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_Incidencia.IDIncidencia , 25)||RPAD(v\_Incidencia.Causa , 50)||

RPAD(v\_Incidencia.Gravedad , 25)|| RPAD(v\_Incidencia.FechaPenalizacion , 25)||

RPAD(v\_Incidencia.Puntos , 25)|| RPAD(v\_Incidencia.Conductor , 25));

FETCH C INTO v\_Incidencia;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,v\_Causa IN Incidencia.Causa%TYPE,v\_Gravedad IN

Incidencia.Gravedad%TYPE, v\_FechaPenalizacion IN Incidencia.FechaPenalizacion%TYPE,v\_Puntos

IN Incidencia.Puntos%TYPE, v\_Conductor IN Incidencia.Conductor%TYPE,salidaEsperada BOOLEAN)

IS

v\_IDIncidencia Incidencia.IDIncidencia%TYPE;

BEGIN

INSERT INTO Incidencia ( Causa, Gravedad, FechaPenalizacion , Puntos , Conductor)

VALUES (v\_Causa , v\_Gravedad , v\_FechaPenalizacion , v\_Puntos, v\_Conductor);

v\_IDIncidencia:= SEC\_INCIDENCIA.CURRVAL;

SELECT \* INTO v\_Incidencia FROM Incidencia WHERE IDIncidencia = v\_IDIncidencia;

IF v\_Incidencia.Causa != v\_Causa AND v\_Incidencia.Gravedad != v\_Gravedad AND

v\_Incidencia.FechaPenalizacion != v\_FechaPenalizacion

AND v\_Incidencia.Puntos != v\_Puntos AND v\_Incidencia.Conductor != v\_Conductor THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Actualizar(nombrePrueba VARCHAR2,v\_IDIncidencia IN

Incidencia.IDIncidencia%TYPE,v\_Causa IN Incidencia.Causa%TYPE,v\_Gravedad IN

Incidencia.Gravedad%TYPE, v\_FechaPenalizacion IN Incidencia.FechaPenalizacion%TYPE,

v\_Puntos IN Incidencia.Puntos%TYPE, v\_Conductor IN Incidencia.Conductor%TYPE, salidaEsperada

BOOLEAN)

IS

BEGIN

UPDATE Incidencia SET Causa = v\_Causa , Gravedad = v\_Gravedad , FechaPenalizacion =

v\_FechaPenalizacion , Puntos = v\_Puntos , Conductor = v\_Conductor WHERE IDIncidencia =

v\_IDIncidencia;

SELECT \* INTO v\_Incidencia FROM Incidencia WHERE IDIncidencia = v\_IDIncidencia;

IF v\_Incidencia.Causa != v\_Causa AND v\_Incidencia.Gravedad != v\_Gravedad AND

v\_Incidencia.FechaPenalizacion != v\_FechaPenalizacion AND v\_Incidencia.Puntos != v\_Puntos

AND v\_Incidencia.Conductor != v\_Conductor THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDIncidencia IN Incidencia.IDIncidencia%TYPE,

salidaEsperada BOOLEAN )

IS

v\_NumIncidencia NUMBER := 0;

BEGIN

DELETE FROM Incidencia WHERE IDIncidencia = v\_IDIncidencia;

SELECT COUNT(\*) INTO v\_NumIncidencia FROM Incidencia WHERE IDIncidencia = v\_IDIncidencia;

IF v\_NumIncidencia != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

--Tabla Mercancia

CREATE OR REPLACE PACKAGE BODY PCK\_Mercancia

IS

CURSOR C IS

SELECT \* FROM Mercancia ;

v\_salida BOOLEAN := TRUE;

v\_Mercancia Mercancia%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM Mercancia;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_Mercancia;

DBMS\_OUTPUT.PUT\_LINE(RPAD('IDMercancia:' , 25)||RPAD('ConservaMercancia:' ,

25)||RPAD('NombreMercancia:' , 25)|| RPAD('CantidadMercancia:' , 25)|| RPAD('TipoMercancia:'

, 25)|| RPAD('Pedido:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 150 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_Mercancia.IDMercancia , 25)||RPAD(v\_Mercancia.ConservaMercancia

, 25)|| RPAD(v\_Mercancia.NombreMercancia , 25)|| RPAD(v\_Mercancia.CantidadMercancia , 25)||

RPAD(v\_Mercancia.TipoMercancia , 25)|| RPAD(v\_Mercancia.Pedido , 25));

FETCH C INTO v\_Mercancia;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,v\_ConservaMercancia IN

Mercancia.ConservaMercancia%TYPE,v\_NombreMercancia IN Mercancia.NombreMercancia%TYPE,

v\_CantidadMercancia IN Mercancia.CantidadMercancia%TYPE,v\_TipoMercancia IN

Mercancia.TipoMercancia%TYPE, v\_Pedido IN Mercancia.Pedido%TYPE,salidaEsperada BOOLEAN)

IS

v\_IDMercancia Mercancia.IDMercancia%TYPE;

BEGIN

INSERT INTO Mercancia ( ConservaMercancia,NombreMercancia, CantidadMercancia , TipoMercancia

, Pedido)

VALUES (v\_ConservaMercancia , v\_NombreMercancia , v\_CantidadMercancia , v\_TipoMercancia

, v\_Pedido);

v\_IDMercancia:= SEC\_MERCANCIA.CURRVAL;

SELECT \* INTO v\_Mercancia FROM Mercancia WHERE IDMercancia = v\_IDMercancia;

IF v\_Mercancia.ConservaMercancia != v\_ConservaMercancia AND v\_Mercancia.NombreMercancia !=

v\_NombreMercancia AND v\_Mercancia.CantidadMercancia != v\_CantidadMercancia

AND v\_Mercancia.TipoMercancia != v\_TipoMercancia AND v\_Mercancia.Pedido != v\_Pedido THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Actualizar(nombrePrueba VARCHAR2,v\_IDMercancia IN

Mercancia.IDMercancia%TYPE,v\_ConservaMercancia IN

Mercancia.ConservaMercancia%TYPE,v\_NombreMercancia IN Mercancia.NombreMercancia%TYPE,

v\_CantidadMercancia IN Mercancia.CantidadMercancia%TYPE,

v\_TipoMercancia IN Mercancia.TipoMercancia%TYPE, v\_Pedido IN Mercancia.Pedido%TYPE,

salidaEsperada BOOLEAN)

IS

BEGIN

UPDATE Mercancia SET ConservaMercancia = v\_ConservaMercancia , NombreMercancia =

v\_NombreMercancia , CantidadMercancia = v\_CantidadMercancia , TipoMercancia =

v\_TipoMercancia , Pedido = v\_Pedido WHERE IDMercancia = v\_IDMercancia;

SELECT \* INTO v\_Mercancia FROM Mercancia WHERE IDMercancia = v\_IDMercancia;

IF v\_Mercancia.ConservaMercancia != v\_ConservaMercancia AND v\_Mercancia.NombreMercancia !=

v\_NombreMercancia AND v\_Mercancia.CantidadMercancia != v\_CantidadMercancia AND

v\_Mercancia.TipoMercancia != v\_TipoMercancia AND v\_Mercancia.Pedido != v\_Pedido THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDMercancia IN Mercancia.IDMercancia%TYPE,

salidaEsperada BOOLEAN )

IS

v\_NumMercancia NUMBER := 0;

BEGIN

DELETE FROM Mercancia WHERE IDMercancia = v\_IDMercancia;

SELECT COUNT(\*) INTO v\_NumMercancia FROM Mercancia WHERE IDMercancia = v\_IDMercancia;

IF v\_NumMercancia != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

--Tabla Pedido

CREATE OR REPLACE PACKAGE BODY PCK\_Pedido

IS

CURSOR C IS

SELECT \* FROM Pedido ;

v\_salida BOOLEAN := TRUE;

v\_Pedido Pedido%ROWTYPE;

PROCEDURE Inicializar

IS

BEGIN

DELETE FROM Pedido;

END Inicializar;

PROCEDURE Consultar

IS

BEGIN

OPEN C;

FETCH C INTO v\_Pedido;

DBMS\_OUTPUT.PUT\_LINE(RPAD('IDPedido:' , 25)||RPAD('HoraSalida:' ,

25)||RPAD('HoraLlegadaPrevista:' , 25)|| RPAD('HoraLlegada:' , 25)||

RPAD('DireccionDestino:' , 25)|| RPAD('Conductor:' , 25));

DBMS\_OUTPUT.PUT\_LINE(LPAD( '-', 150 , '-'));

WHILE C%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE( RPAD(v\_Pedido.IDPedido , 25)||RPAD(v\_Pedido.HoraSalida , 25)||

RPAD(v\_Pedido.HoraLlegadaPrevista , 25)|| RPAD(v\_Pedido.HoraLlegada , 25)||

RPAD(v\_Pedido.DireccionDestino , 25)|| RPAD(v\_Pedido.Conductor , 25));

FETCH C INTO v\_Pedido;

END LOOP;

CLOSE C;

END Consultar;

PROCEDURE Insertar (nombrePrueba VARCHAR2,v\_HoraSalida IN

Pedido.HoraSalida%TYPE,v\_HoraLlegadaPrevista IN Pedido.HoraLlegadaPrevista%TYPE,

v\_HoraLlegada IN Pedido.HoraLlegada%TYPE,v\_DireccionDestino IN Pedido.DireccionDestino%TYPE,

v\_Conductor IN Pedido.Conductor%TYPE,salidaEsperada BOOLEAN)

IS

v\_IDPedido Pedido.IDPedido%TYPE;

BEGIN

INSERT INTO Pedido ( HoraSalida, HoraLlegadaPrevista, HoraLlegada , DireccionDestino ,

Conductor)

VALUES (v\_HoraSalida , v\_HoraLlegada , v\_HoraLlegadaPrevista , v\_DireccionDestino,

v\_Conductor);

v\_IDPedido:= SEC\_PEDIDO.CURRVAL;

SELECT \* INTO v\_Pedido FROM Pedido WHERE IDPedido = v\_IDPedido;

IF v\_Pedido.HoraSalida != v\_HoraSalida AND v\_Pedido.HoraLlegadaPrevista !=

v\_HoraLlegadaPrevista AND v\_Pedido.HoraLlegada != v\_HoraLlegada

AND v\_Pedido.DireccionDestino != v\_DireccionDestino AND v\_Pedido.Conductor != v\_Conductor THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Insertar;

PROCEDURE Actualizar (nombrePrueba VARCHAR2,v\_IDPedido IN Pedido.IDPedido%TYPE,v\_HoraSalida

IN Pedido.HoraSalida%TYPE,v\_HoraLlegadaPrevista IN Pedido.HoraLlegadaPrevista%TYPE,

v\_HoraLlegada IN Pedido.HoraLlegada%TYPE,

v\_DireccionDestino IN Pedido.DireccionDestino%TYPE, v\_Conductor IN Pedido.Conductor%TYPE,

salidaEsperada BOOLEAN)

IS

BEGIN

UPDATE Pedido SET HoraSalida = v\_HoraSalida , HoraLlegadaPrevista = v\_HoraLlegadaPrevista ,

HoraLlegada = v\_HoraLlegada , DireccionDestino = v\_DireccionDestino , Conductor =

v\_Conductor WHERE IDPedido = v\_IDPedido;

SELECT \* INTO v\_Pedido FROM Pedido WHERE IDPedido = v\_IDPedido;

IF v\_Pedido.HoraSalida != v\_HoraSalida AND v\_Pedido.HoraLlegadaPrevista !=

v\_HoraLlegadaPrevista AND v\_Pedido.HoraLlegada != v\_HoraLlegada AND

v\_Pedido.DireccionDestino != v\_DireccionDestino AND v\_Pedido.Conductor != v\_Conductor THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Actualizar;

PROCEDURE Eliminar (nombrePrueba VARCHAR2, v\_IDPedido IN Pedido.IDPedido%TYPE,

salidaEsperada BOOLEAN )

IS

v\_NumPedido NUMBER := 0;

BEGIN

DELETE FROM Pedido WHERE IDPedido = v\_IDPedido;

SELECT COUNT(\*) INTO v\_NumPedido FROM Pedido WHERE IDPedido = v\_IDPedido;

IF v\_NumPedido != 0 THEN

v\_Salida := FALSE;

END IF;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(v\_Salida , salidaEsperada) );

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(nombrePrueba || ':' || ASSERT\_EQUALS(FALSE , salidaEsperada) );

ROLLBACK;

END Eliminar;

END;

/

**Scripts de casos de prueba (packages)**

/\*Casos de prueba\*/

SET SERVEROUTPUT ON

/

DECLARE

ID\_CA1 NUMBER;

ID\_CA2 NUMBER;

ID\_CA3 NUMBER;

ID\_CA4 NUMBER;

ID\_CA5 NUMBER;

ID\_CA6 NUMBER;

ID\_I1 NUMBER;

ID\_I2 NUMBER;

ID\_I3 NUMBER;

ID\_I4 NUMBER;

ID\_I5 NUMBER;

ID\_I6 NUMBER;

ID\_P1 NUMBER;

ID\_P2 NUMBER;

ID\_P3 NUMBER;

ID\_P4 NUMBER;

ID\_P5 NUMBER;

ID\_P6 NUMBER;

ID\_P7 NUMBER;

ID\_M1 NUMBER;

ID\_M2 NUMBER;

ID\_M3 NUMBER;

ID\_M4 NUMBER;

ID\_M5 NUMBER;

ID\_M6 NUMBER;

BEGIN

/\*Camion\*/

PCK\_Camion.Inicializar;

PCK\_Camion.Insertar ('prueba normal 1','1234DEF',TO\_DATE('12/02/2019',

'DD/MM/YYYY'),'0','Frio','1',null,TRUE);

PCK\_Camion.Insertar ('prueba normal 2','2345DEF',TO\_DATE('11/02/2019',

'DD/MM/YYYY'),'0','Seco','1',null,TRUE);

PCK\_Camion.Insertar ('prueba normal 3','3456DEF',TO\_DATE('15/02/2019',

'DD/MM/YYYY'),'0','Congelado','1',null,TRUE);

PCK\_Camion.Insertar ('prueba normal 4','9876GST',TO\_DATE('14/02/2019',

'DD/MM/YYYY'),'0','Frio','0',TO\_DATE('25/12/2019', 'DD/MM/YYYY'),TRUE);

PCK\_Camion.Insertar ('prueba normal 5','0000QWE',TO\_DATE('17/02/2019',

'DD/MM/YYYY'),'0','Congelado','1',null,TRUE);

PCK\_Camion.Insertar ('prueba normal 6','3485DHR',TO\_DATE('21/02/2019',

'DD/MM/YYYY'),'0','Seco','1',null,TRUE);

PCK\_Camion.Insertar ('prueba matricula mal','1234DDEF',TO\_DATE('12/02/2019',

'DD/MM/YYYY'),'0','Frio','1',null,FALSE);

PCK\_Camion.Insertar ('prueba fallo tipocamion','0987DEF',TO\_DATE('18/04/2019',

'DD/MM/YYYY'),'0','Congeladoo','1',null,FALSE);

PCK\_Camion.Actualizar ('prueba actualizacion de tipocamion','1234DEF',TO\_DATE('12/02/2019',

'DD/MM/YYYY'),'0','Congelado','1',null,TRUE);

PCK\_Camion.Actualizar ('prueba actualizacion de fecha','2345DEF',TO\_DATE('15/02/2019',

'DD/MM/YYYY'),'0','Frio','1',null,TRUE);

PCK\_Camion.Eliminar('prueba eliminacion','3485DHR',TRUE);

PCK\_Camion.Consultar ;

/\*Conductor\*/

PCK\_Conductor.Inicializar;

PCK\_Conductor.Insertar ('prueba normal

1','Alfonso','Ramirez','12345678A',TO\_DATE('12/02/1987',

'DD/MM/YYYY'),'654321987','aramirez@gmail.com','1500','Empleado','15','2',TRUE);

PCK\_Conductor.Insertar ('prueba normal

2','Miguel','Jimenez','11223344W',TO\_DATE('14/03/1978',

'DD/MM/YYYY'),'654294634','mjimenez@gmail.com','1500','Extra','5','6',TRUE);

PCK\_Conductor.Insertar ('prueba normal

3','Roberto','Fernandez','98766789K',TO\_DATE('13/09/1983',

'DD/MM/YYYY'),'687524153','rfernandez@gmail.com','1500','Extra','8','7',TRUE);

PCK\_Conductor.Insertar ('prueba normal 4','Luis','Gallo','12349876Y',TO\_DATE('15/08/1978',

'DD/MM/YYYY'),'684597153','lgallo@gmail.com','1200','Empleado','25','2',TRUE);

PCK\_Conductor.Insertar ('prueba normal

5','Manuel','Rodriguez','99887766H',TO\_DATE('16/09/1989',

'DD/MM/YYYY'),'684259743','mrodriguez@gmail.com','1500','Empleado','9','3',TRUE);

PCK\_Conductor.Insertar ('prueba normal

6','Alejandro','Perez','52486375P',TO\_DATE('25/08/1965',

'DD/MM/YYYY'),'698547123','aperez@gmail.com','1500','Empleado','17','0',TRUE);

PCK\_Conductor.Insertar ('prueba fallo

DNI','Jose','Fernando','998877661',TO\_DATE('16/09/1989',

'DD/MM/YYYY'),'684259743','mrodriguez@gmail.com','1500','Empleado','9','3',FALSE);

PCK\_Conductor.Insertar ('prueba fallo fecha

nacimiento','Enrique','Enrique','99557766H',TO\_DATE('16/09/2077',

'DD/MM/YYYY'),'684259743','mrodriguez@gmail.com','1500','Empleado','9','3',FALSE);

PCK\_Conductor.Actualizar ('prueba actualizacion de

nombre','Roberta','Fernandez','98766789K',TO\_DATE('13/09/1983',

'DD/MM/YYYY'),'687524153','rfernandez@gmail.com','1500','Extra','8','7',TRUE);

PCK\_Conductor.Actualizar ('prueba actualizacion de fallo

telefono','Luis','Gallo','12349876Y',TO\_DATE('15/08/1978',

'DD/MM/YYYY'),'6845971532','lgallo@gmail.com','1200','Empleado','25','2',FALSE);

PCK\_Conductor.Actualizar ('prueba actualizacion de fallo

email','Alfonso','Ramirez','12345678A',TO\_DATE('12/02/1987',

'DD/MM/YYYY'),'654321987','aramirezgmail.com','1500','Empleado','15','2',FALSE);

PCK\_Conductor.Eliminar('prueba eliminacion','52486375P',TRUE);

PCK\_Conductor.Consultar ;

/\*ConductorAsignado\*/

PCK\_ConductorAsignado.Inicializar;

PCK\_ConductorAsignado.Insertar ('prueba normal 1','11223344W','1234DEF',TRUE);

ID\_CA1:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Insertar ('prueba normal 2','12345678A','9876GST',TRUE);

ID\_CA2:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Insertar ('prueba normal 3','12349876Y','0000QWE',TRUE);

ID\_CA3:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Insertar ('prueba normal 4','99887766H','2345DEF',TRUE);

ID\_CA4:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Insertar ('prueba normal 5','98766789K','3456DEF',TRUE);

ID\_CA5:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Insertar ('prueba normal 6','98766789K','2345DEF',TRUE);

ID\_CA6:=SEC\_CONDUCTORASIGNADO.currval;

PCK\_ConductorAsignado.Eliminar('prueba eliminacion',ID\_CA6,TRUE);

PCK\_ConductorAsignado.Consultar;

/\*Incidencia\*/

PCK\_Incidencia.Inicializar;

PCK\_Incidencia.Insertar ('prueba normal 1','Robo de material','Grave',TO\_DATE('12/02/2019',

'DD/MM/YYYY'),'7','11223344W',TRUE);

ID\_I1:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Insertar ('prueba normal 2','Retraso en entrega','Leve',TO\_DATE('15/03/2019',

'DD/MM/YYYY'),'2','11223344W',TRUE);

ID\_I2:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Insertar ('prueba normal 3','Entrega cancelada por tardar

demasiado','Grave',TO\_DATE('18/04/2019', 'DD/MM/YYYY'),'9','11223344W',TRUE);

ID\_I3:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Insertar ('prueba normal 4','Robo de material','Media',TO\_DATE('25/04/2019',

'DD/MM/YYYY'),'5','12349876Y',TRUE);

ID\_I4:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Insertar ('prueba normal 5','Robo de neumatico','Grave',TO\_DATE('03/01/2019',

'DD/MM/YYYY'),'7','12349876Y',TRUE);

ID\_I5:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Insertar ('prueba normal 6','Robo de neumatico','Grave',TO\_DATE('03/01/2019',

'DD/MM/YYYY'),'7','12349876Y',TRUE);

ID\_I6:=SEC\_INCIDENCIA.currval;

PCK\_Incidencia.Actualizar ('prueba actualizacion',ID\_I1,'Robo de

material','Grave',TO\_DATE('12/02/2018', 'DD/MM/YYYY'),'7','11223344W',TRUE);

PCK\_Incidencia.Actualizar ('prueba fallo actualizacion en fecha',ID\_I1,'Robo de

material','Grave',TO\_DATE('12/02/2077', 'DD/MM/YYYY'),'7','11223344W',FALSE);

PCK\_Incidencia.Eliminar('prueba eliminacion',ID\_I6,TRUE);

PCK\_Incidencia.Consultar ;

--Pedido

PCK\_Pedido.Inicializar;

PCK\_Pedido.Insertar ('prueba normal 1','05,15','07,15','06,45','C/ Vaticano,

28','11223344W',TRUE);

ID\_P1:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba normal 2','12,45','15,45','15,37','C/ Pepe,

14','12345678A',TRUE);

ID\_P2:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba normal 3','08,25','10,25','10,45','C/ Francisco,

54','11223344W',TRUE);

ID\_P3:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba normal 4','07,45','09,45','09,37','C/ Roberto,

21','12345678A',TRUE);

ID\_P4:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba normal 5','19,45','21,15','21,14','C/ Imagen,

80','98766789K',TRUE);

ID\_P5:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba normal 6','18,45','20,15','20,06','C/ Imagen,

80','98766789K',TRUE);

ID\_P6:=SEC\_PEDIDO.currval;

PCK\_Pedido.Insertar ('prueba fallo insertar hora llegada

prevista','19,45','18,15','21,14','C/ Imagen, 80','12345678A',FALSE);

ID\_P7:=SEC\_PEDIDO.currval;

PCK\_Pedido.Actualizar ('prueba actualizacion',ID\_P2,'12,45','15,45','15,37','C/ Pepe,

19','12345678A',TRUE);

PCK\_Pedido.Actualizar ('prueba fallo actualizacion hora

llegada',ID\_P4,'07,45','09,45','6,42','C/ Roberto, 21','12345678A',FALSE);

PCK\_Pedido.Eliminar('prueba eliminacion',ID\_P6,TRUE);

PCK\_Pedido.Consultar;

--Mercancia

PCK\_Mercancia.Inicializar;

PCK\_Mercancia.Insertar ('prueba normal 1','Seco','Pala','50','Jardineria',ID\_P1,TRUE);

ID\_M1:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Insertar ('prueba normal 2','Frio','Pizzas','10','Alimentacion',ID\_P2,TRUE);

ID\_M2:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Insertar ('prueba normal 3','Congelado','Mariscos','10','Alimentacion',ID\_P3,TRUE);

ID\_M3:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Insertar ('prueba normal 4','Frio','Refrescos','600','Alimentacion',ID\_P4,TRUE);

ID\_M4:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Insertar ('prueba normal 5','Seco','Macetas','15','Jardineria',ID\_P5,TRUE);

ID\_M5:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Insertar ('prueba normal 6','Seco','Semillas','15','Jardineria',ID\_P4,TRUE);

ID\_M6:=SEC\_MERCANCIA.currval;

PCK\_Mercancia.Actualizar ('prueba

actualizacion',ID\_M1,'Seco','Camisetas','150','Textil',ID\_P1,TRUE);

PCK\_Mercancia.Actualizar ('prueba fallo actualizacion

conserva',ID\_M4,'Secooo','Refrescos','600','Alimentacion',ID\_P4,FALSE);

PCK\_Mercancia.Eliminar('prueba eliminacion',ID\_M6,TRUE);

PCK\_Mercancia.Consultar ;

END;

/

**Scripts de casos de prueba (functions y procedures)**

SET SERVEROUTPUT ON;

/\*RF-21\*/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(fn\_consulta\_puntos('12345678A'));

END;

//

\*RF-22\*/

EXEC pr\_conductores\_por\_puntos;

/\*RF-23\*/

EXEC PR\_dar\_puntuacion('12345678A', '2', '1');

/\*RF-24\*/

EXEC PR\_Quitar\_puntuacion('11223344W', '6', '1');

/\*RF-25\*/

EXEC PR\_CrearAsignacion('12345678A', '1234DEF');

/\*RF-26\*/

EXEC PR\_EliminarAsignacion('12345678A', '1234DEF');

/\*RF-27\*/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(fn\_consulta\_conductor('1234DEF'));

END;

/

/\*RF-28\*/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(fn\_consulta\_camiones('12345678A'));

END;

/