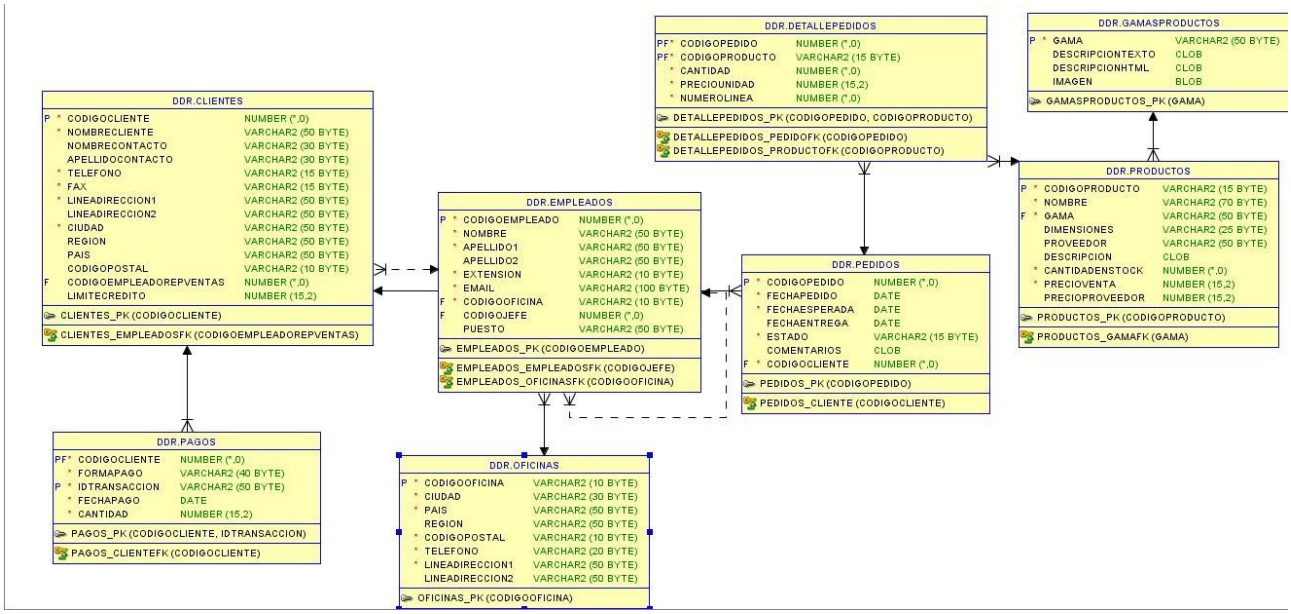

EJERCICIOS RESUELTOS SOBRE BASE DE DATOS DE LA IMAGEN

1. Mostrar el nombre de un cliente dado su código.

```
declare
  v_codigocliente clientes.codigocliente%type := &codigo;
  v_nombrecliente clientes.nombrecliente%type;
begin

  select nombrecliente into v_nombrecliente
  from clientes
  where codigocliente = v_codigocliente;

  DBMS_OUTPUT.PUT_LINE('El nombre del cliente es ' || v_nombrecliente);

end;
```

2. Mostrar el precioVenta y la gama de un producto dado su código.

```
declare
  v_codigoproducto productos.codigoproducto%type := &codigo;
  v_nombreproducto productos.nombre%type;
  v_gamaproducto productos.gama%type;
begin

  select nombre, gama into v_nombreproducto, v_gamaproducto
  from productos
  where codigoproducto = v_codigoproducto;

  DBMS_OUTPUT.PUT_LINE('El nombre del producto es ' || v_nombreproducto
```

```
                || ' y su gama es '  || v_gamaproducto);

end;
```

3. Mostrar toda la información de un pedido dado su código (fechaEsperada, fechaEntrega, fechapedido, estado, comentarios)

```
declare
  v_codigopedido pedidos.codigopedido%type := &codigo;
  v_pedido pedidos%rowtype;
begin

  select * into v_pedido
  from pedidos
  where codigopedido = v_codigopedido;

  DBMS_OUTPUT.PUT_LINE('La fecha de pedido es ' || v_pedido.fechapedido
                || ', la fecha esperada es ' || v_pedido.fechaesperada
                || ', la fecha de entrega es ' || v_pedido.fechaentrega
                || ', el estado es ' || v_pedido.estado
                || ' y los comentarios son ' || v_pedido.comentarios
                );

end;
```

4. Realizar una función que me devuelva la suma de pagos que ha realizado. Pasa el código por parámetro.

```
create or replace function Pagos_cliente(v_codigocliente clientes.codigocliente%type)
return Number
as
  v_sumapagos pagos.cantidad%type := 0;
begin

  select sum(cantidad) into v_sumapagos
  from pagos
  where codigocliente = v_codigocliente;

  return v_sumapagos;

end;
/

declare
  v_codigocliente clientes.codigocliente%type := &codigo;
  v_suma pagos.cantidad%type;
```

```
begin
  v_suma := Pagos_cliente(v_codigocliente);
  DBMS_OUTPUT.PUT_LINE('La suma de pagos es ' || v_suma);

end;
```

5. Realizar un método o procedimiento que muestre el total en euros de un pedido, pásale el código por parámetro.

```
create or replace procedure total_pedido(v_codigopedido pedidos.codigopedido%type)
as
  v_total number(8) := 0;
begin

  select sum(dp.cantidad * dp.PRECIOUNIDAD) into v_total
  from pedidos p, detallepedidos dp
  where p.codigopedido = dp.codigopedido and p.codigopedido = v_codigopedido;

  DBMS_OUTPUT.PUT_LINE('El pedido total es ' || v_total);

end;
/

declare
  v_codigopedido pedidos.codigopedido%type := &codigo;
begin
  total_pedido(v_codigopedido);

end;
```

6. Mostrar el nombre de un cliente dado su código. Controla en caso de que no se encuentre, mostrando un mensaje.

```
declare
  v_codigocliente clientes.codigocliente%type := &codigo;
  v_nombrecliente clientes.nombrecliente%type;
begin

  select nombrecliente into v_nombrecliente
  from clientes
  where codigocliente = v_codigocliente;

  DBMS_OUTPUT.PUT_LINE('El nombre del cliente es ' || v_nombrecliente);

exception
  when no_data_found then
```

```
    DBMS_OUTPUT.PUT_LINE('No existe el cliente');

end;
```

7. Realizar una función que me devuelva la suma de pagos que ha realizado. Pasa el código por parámetro. En caso de que no se encuentre, devuelve un -1.

```
create or replace function Pagos_cliente(v_codigocliente clientes.codigocliente%type)
return Number
as
  v_sumapagos pagos.cantidad%type := 0;
begin

  select sum(cantidad) into v_sumapagos
  from pagos
  where codigocliente = v_codigocliente;

  if v_sumapagos is null then
    raise no_data_found;
  else
    return v_sumapagos;
  end if;
exception
  when no_data_found then
    return -1;

end;
/

declare
  v_codigocliente clientes.codigocliente%type := &codigo;
  v_suma pagos.cantidad%type;
begin
  v_suma := Pagos_cliente(v_codigocliente);
  if v_suma = -1 then
    DBMS_OUTPUT.PUT_LINE('El cliente no existe');
  else
    DBMS_OUTPUT.PUT_LINE('La suma de pagos es ' || v_suma);
  end if;

end;
```

8. Realizar un método o procedimiento que muestre el total en euros de un pedido, pásale el código por parámetro. Controla en caso de que no se encuentre, en ese caso devuelve un 0. Después pásale otro parámetro, si supera ese limite dado, lanzaremos una excepción propia y devolveremos un 0.

```
create or replace function total_pedido_func
            (v_codigopedido pedidos.codigopedido%type, v_limite number)
return number
as
  v_total number(8) := 0;
  limite_superado exception;
begin

  select sum(dp.cantidad * dp.PRECIOUNIDAD) into v_total
  from pedidos p, detallepedidos dp
  where p.codigopedido = dp.codigopedido and p.codigopedido = v_codigopedido;

  if v_total is null then
    raise no_data_found;
  else
    if v_limite < v_total then
      raise limite_superado;
    else
      return v_total;
    end if;
  end if;

exception
  when no_data_found then
    return -1;
  when limite_superado then
    DBMS_OUTPUT.PUT_LINE('Limite superado');
    return 0;

end;
/

declare
  v_codigopedido pedidos.codigopedido%type := &codigo;
  v_total number(8);
  v_limite number(8) := &limite;
begin
  v_total := total_pedido_func(v_codigopedido , v_limite);

  if v_total = -1 then
    DBMS_OUTPUT.PUT_LINE('no existe el pedido');
  else
    DBMS_OUTPUT.PUT_LINE('El pedido total es ' || v_total);
  end if;

end;
```

9. Crea una función a la que le pasaremos como parámetros de entrada: MATRICULA, NUEVO_PRECIO_COMPRA. La función modificara los datos del coche que tenga la matricula introducida actualizando el precio_compra de la siguiente forma:

-Si precio_compra es nulo hacer un update en el campo precio_compra asignándole el valor de nuevo_precio_compra

-Si no hacer un update en el campo precio_compra asignándole el valor de precio_compra+(precio_compra-nuevo_precio_compra).

La función devolverá el numero de filas actualizadas. Crea un bloque anónimo que ejecute la función anterior y muestre el resultado devuelto por la función.

```
create or replace function actualizaPrecioCoche (
      v_matricula COCHE.MATRICULA%type,
      v_nuevo_precio_compra COCHE.PRECIO_COMPRA%type)
return number
as
  v_precio_compra COCHE.PRECIO_COMPRA%type;
begin

  select precio_compra into v_precio_compra
  from coche where matricula = v_matricula;

  if v_precio_compra is null then
    update coche
    set precio_compra = v_nuevo_precio_compra
    where matricula = v_matricula;
  else
    update coche
    set precio_compra = precio_compra+(precio_compra-v_nuevo_precio_compra)
    where matricula = v_matricula;
  end if;

  return SQL%ROWCOUNT;

end;
/

select * from coche

DECLARE
    v_matricula COCHE.MATRICULA%type := &matricula;
    v_nuevo_precio_compra COCHE.PRECIO_COMPRA%type := &nuevo_precio;
    v_total_filas number(8);
BEGIN
  v_total_filas := actualizaPrecioCoche(v_matricula, v_nuevo_precio_compra);
```

```
  DBMS_OUTPUT.put_line('Se han modificado '  || v_total_filas || ' filas');

END;
```

10. Crea un procedimiento que reciba como parámetros de entrada: P_ID_MARCA, P_NUMERO_COCHES. Utiliza un bucle para insertar N registros nuevos en la tabla COCHE. El numero de registros a insertar viene indicado por el parámetro P_NUMEROS_COCHES(CONTADOR) y el bucle empezará en 1, los datos a insertar serán:

-matricula=’A00’||CONTADOR
-DESCRIPCION=p_id_marca
-id_marca=p_id_marca
-precio_compra=nulo

Controlar excepción para cuando exista algún coche en la base de datos y se viole la clave primaria.

```
create or replace procedure creaCoches(
  p_id_marca coche.id_marca%type,
  p_numero_coches number)
as

begin

    for contador IN 1..p_numero_coches LOOP
      insert into coche values('A00'||contador, p_id_marca, p_id_marca, null);
    END LOOP;

exception
  when dup_val_on_index then
    DBMS_OUTPUT.put_line('Registro duplicado');

end;
/

declare
  p_id_marca coche.id_marca%type := &id;
  p_numero_coches number(8) := &num;
begin

  creaCoches(p_id_marca, p_numero_coches);

end;
```

11. Crea un procedimiento al que le pasaremos el dni_cliente y la matricula. El procedimiento deberá controlar en las ventas de los coches(tabla vende) los siguientes supuestos:

-.Si no existe un registro con ese dni_cliente y esa matricula saltara a la zona de excepciones y mostrara un mensaje “no existe la venta introducida”.

– Si existe la venta introducida:
a) Mostrará el precio antiguo.
b) Actualizará el precio subiendo 1000 euros.
c) Devolverá en un parámetro de salida del procedimiento (ps_nuevo_precio) el precio nuevo tras la actualización. Crea un bloque anónimo que llame al procedimiento anterior y muestre el precio nuevo devuelto por el procedimiento.

```
create or replace procedure actualizaVenta(
  p_dni_cliente vende.dni_cliente%type,
  p_matricula vende.matricula%type,
  ps_nuevo_precio out vende.precio%type
)
as
  venta vende%rowtype;
begin

    select * into venta
    from vende
    where dni_cliente = p_dni_cliente
    and matricula = p_matricula;

    DBMS_OUTPUT.PUT_line('el precio antiguo es ' || venta.precio);

    ps_nuevo_precio :=  venta.precio + 1000;

    update vende
    set precio = ps_nuevo_precio
    where dni_cliente = p_dni_cliente
    and matricula = p_matricula;

EXCEPTION
  WHEN no_data_found then
    DBMS_OUTPUT.PUT_line('No existe la venta introducida');

end;
/

declare
  v_dni_cliente vende.dni_cliente%type := &dni;
  v_matricula vende.matricula%type := &matricula;
  v_nuevo_precio vende.precio%type;
```

```
begin

  actualizaVenta(v_dni_cliente, v_matricula, v_nuevo_precio);
  if v_nuevo_precio is not null then
    DBMS_OUTPUT.PUT_line('el nuevo precio es ' || v_nuevo_precio);
  end if;

end;
```

12. Crear un cursor para ver todos los clientes que no hayan hecho pagos. Hazlo con un loop.

```
declare
  v_nombrecliente clientes.nombrecliente%type;
  cursor clientes_sin_pagos_cursor is
    select nombrecliente
    from clientes c
    where not exists(select codigocliente from pagos where codigocliente = c.codigocliente);
begin

  open clientes_sin_pagos_cursor;

  loop
    fetch clientes_sin_pagos_cursor into v_nombrecliente;
    exit when clientes_sin_pagos_cursor%notfound;

    dbms_output.put_line(v_nombrecliente);

  end loop;

  close clientes_sin_pagos_cursor;

end;
```

13. Crear un cursor para ver todos los clientes que no hayan hecho pagos. Hazlo con un for.

```
declare
  cursor clientes_sin_pagos_cursor is
    select nombrecliente
    from clientes c
    where not exists(select codigocliente from pagos where codigocliente = c.codigocliente);
begin

  for registro in clientes_sin_pagos_cursor loop

    dbms_output.put_line(registro.nombrecliente);
```

```
  end loop;
end;
```