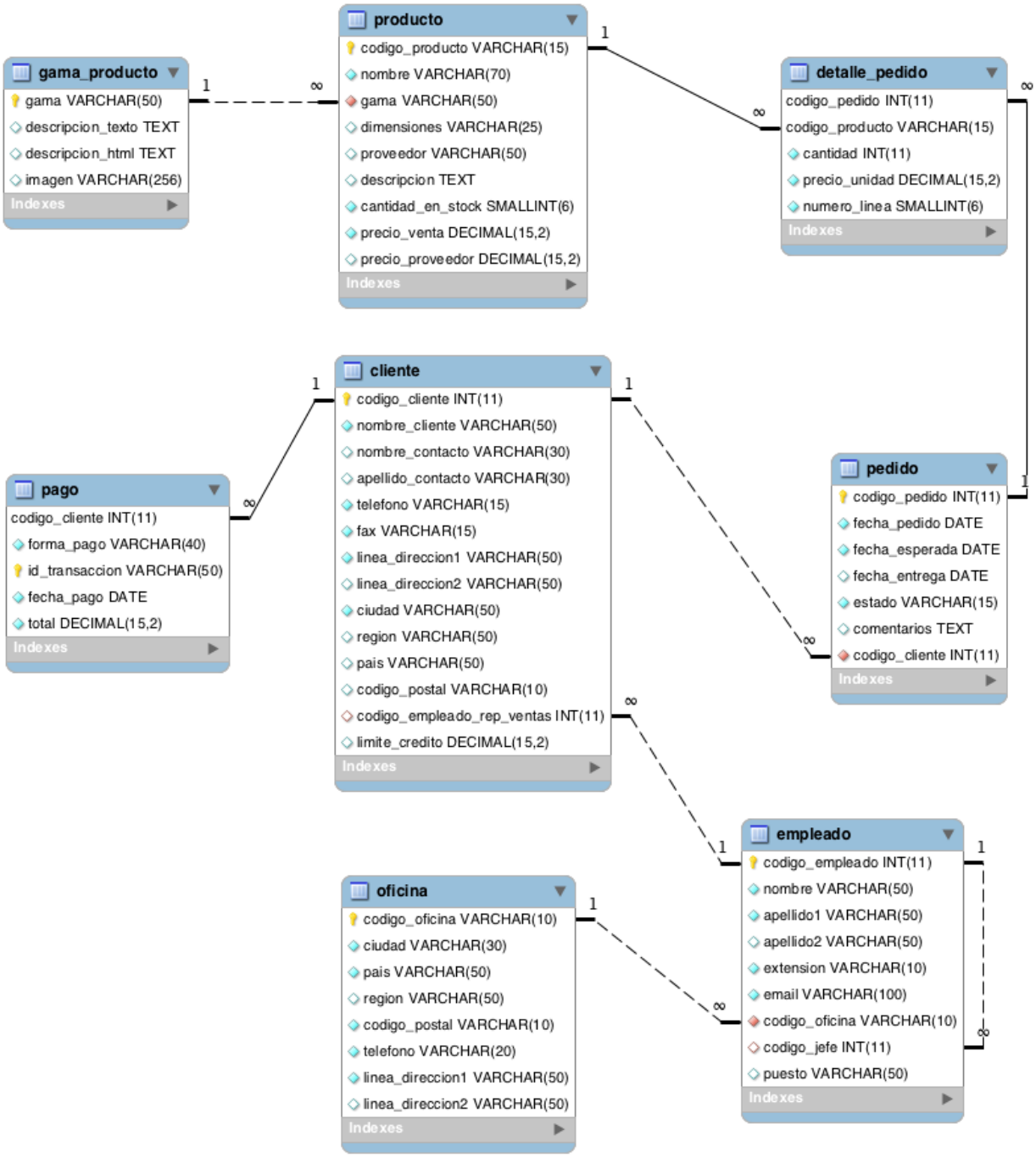
Objetivos:

- Funciones
- Procedimientos

Jardinería



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

```sql
-- EJ1 Mostrar el nombre de un cliente dado su código.--

DROP PROCEDURE IF EXISTS mostrar_nombre_cliente;

DELIMITER //

CREATE PROCEDURE mostrar_nombre_cliente(idCliente INT)

BEGIN

    SELECT nombre_cliente
    FROM cliente
    WHERE codigo_cliente = idCliente;

END//

DELIMITER ;

CALL mostrar_nombre_cliente(10);
```

-- EJ1 Mostrar el nombre de un cliente dado su código.--

```
DROP PROCEDURE IF EXISTS mostrar_nombre_cliente;
DELIMITER //
CREATE PROCEDURE mostrar_nombre_cliente(idCliente INT)
BEGIN
	SELECT nombre_cliente
	FROM cliente
	WHERE codigo_cliente = idCliente;
END//
DELIMITER ;
CALL mostrar_nombre_cliente(10);
```

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

```
-- EJ2 Mostrar el precioVenta y la gama de un producto dado su código. --

DROP PROCEDURE IF EXISTS mostrar_pv_gama;

DELIMITER //

CREATE PROCEDURE mostrar_pv_gama(idProducto VARCHAR(15))
BEGIN


    SELECT precio_Venta, gama
    FROM producto
    WHERE codigo_producto = idProducto
    GROUP BY precio_venta, gama;

END//

DELIMITER ;

CALL mostrar_pv_gama(22225);
```


```
-- EJ2 Mostrar el precioVenta y la gama de un producto dado su código. --

DROP PROCEDURE IF EXISTS mostrar_pv_gama;

DELIMITER //

CREATE PROCEDURE mostrar_pv_gama(idProducto VARCHAR(15))
BEGIN


   SELECT precio_Venta, gama
   FROM producto
   WHERE codigo_producto = idProducto
   GROUP BY precio_venta, gama;

END//

DELIMITER ;

CALL mostrar_pv_gama(22225);
```

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

```
-- EJ3 Mostrar toda la informacion de un pedido dado su código (fechaEsperada, fechaEntrega, fechapedido, estado,comentarios). --

DROP PROCEDURE IF EXISTS mostrar_informacion_pedido;

DELIMITER $$

CREATE PROCEDURE mostrar_informacion_pedido(codPedido INT)
BEGIN

    SELECT fecha_esperada, fecha_entrega, fecha_pedido, estado, comentarios
    FROM pedido
    WHERE codigo_pedido = codPedido
    ORDER BY fecha_esperada, fecha_entrega, fecha_pedido, estado, comentarios;

END$$

DELIMITER ;

CALL mostrar_informacion_pedido(30);
```


```
-- EJ3 Mostrar toda la informacion de un pedido dado su código
(fechaEsperada, fechaEntrega, fechapedido, estado,comentarios). --

DROP PROCEDURE IF EXISTS mostrar_informacion_pedido;

DELIMITER $$

CREATE PROCEDURE mostrar_informacion_pedido(codPedido INT)
BEGIN

      SELECT fecha_esperada, fecha_entrega, fecha_pedido, estado, comentarios
      FROM pedido
      WHERE codigo_pedido = codPedido
      ORDER BY fecha_esperada, fecha_entrega, fecha_pedido, estado,
comentarios;

END$$

DELIMITER ;


CALL mostrar_informacion_pedido(30);
```

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

```
-- EJ4 Realizar una función que me devuelva la suma de pagos que ha realizado. Pasa el código por parámetro. --

DROP FUNCTION IF EXISTS suma_pagos_realizado;

DELIMITER //
CREATE FUNCTION suma_pagos_realizado(idPago INT) RETURNS INT DETERMINISTIC
BEGIN

    DECLARE sumaPago INT;

        SELECT SUM(total)
        INTO sumaPago
        FROM pago
        WHERE codigo_cliente = idPago;

    RETURN sumaPago;

END//

DELIMITER ;

SELECT suma_pagos_realizado(4);
```


```
-- EJ4 Realizar una función que me devuelva la suma de pagos que ha realizado. Pasa el código por parámetro. --

DROP FUNCTION IF EXISTS suma_pagos_realizado;

DELIMITER //
CREATE FUNCTION suma_pagos_realizado(idPago INT) RETURNS INT DETERMINISTIC
BEGIN

      DECLARE sumaPago INT;

            SELECT SUM(total)
            INTO sumaPago
            FROM pago
            WHERE codigo_cliente = idPago;

      RETURN sumaPago;

END//

DELIMITER ;

SELECT suma_pagos_realizado(4);
```

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

```
-- EJ5 Realizar un método o procedimiento que muestre el total en euros de un pedido, pásale el código por parámetro. --

DROP PROCEDURE IF EXISTS total_euros_pedido;

DELIMITER $$

CREATE PROCEDURE total_euros_pedido(codPedido INT)
BEGIN

    SELECT SUM(cantidad * precio_unidad)
    FROM detalle_pedido
    WHERE codigo_pedido = codPedido;

END$$

DELIMITER ;

CALL total_euros_pedido(40);
```


```
-- EJ5 Realizar un método o procedimiento que muestre el total en euros de un pedido, pásale el código por parámetro. --

DROP PROCEDURE IF EXISTS total_euros_pedido;

DELIMITER $$

CREATE PROCEDURE total_euros_pedido(codPedido INT)
BEGIN

      SELECT SUM(cantidad * precio_unidad)
      FROM detalle_pedido
      WHERE codigo_pedido = codPedido;

END$$

DELIMITER ;

CALL total_euros_pedido(40);
```

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

```
-- EJ6 Mostrar el nombre de un cliente dado su código. Controla en caso de que no se encuentre, mostrando unmensaje por ejemplo. --

DROP FUNCTION IF EXISTS mostrar_cliente_codigo;

DELIMITER $$

CREATE FUNCTION mostrar_cliente_codigo(codCliente INT) RETURNS VARCHAR(50) DETERMINISTIC
BEGIN

    DECLARE nombreCliente VARCHAR(50);

    SELECT nombre_cliente
    INTO nombreCliente
    FROM cliente
    WHERE codigo_cliente = codCliente;

    IF (nombreCliente IS NOT NULL) THEN
        RETURN nombreCliente;
    ELSE
        RETURN CONCAT('NO EXISTE EL CLIENTE CON ID ',codCliente);
    END IF;

END$$

DELIMITER ;

SELECT mostrar_cliente_codigo(8);
-- Mensaje cuando el cliente no se encuentra --
SELECT mostrar_cliente_codigo(60);
```


```
DROP FUNCTION IF EXISTS mostrar_cliente_codigo;

DELIMITER $$

CREATE FUNCTION mostrar_cliente_codigo(codCliente INT) RETURNS VARCHAR(50)
DETERMINISTIC

BEGIN

	DECLARE nombreCliente VARCHAR(50);

	SELECT nombre_cliente

	INTO nombreCliente

	FROM cliente

	WHERE codigo_cliente = codCliente;

	IF (nombreCliente IS NOT NULL) THEN

		RETURN nombreCliente;

	ELSE

		RETURN CONCAT('NO EXISTE EL CLIENTE CON ID ',codCliente);

	END IF;

END$$

DELIMITER ;
```

```
SELECT mostrar_cliente_codigo(8);

-- Mensaje cuando el cliente no se encuentra --

SELECT mostrar_cliente_codigo(60);
```

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

```
-- EJ7 Realizar una función que me devuelva la suma de pagos que ha realizado.
-- Pasa el código por parámetro. Controla en caso de que no se encuentre,
-- en ese caso devuelve un -1.--

DROP FUNCTION IF EXISTS suma_pagos_realizado2;

DELIMITER //
CREATE FUNCTION suma_pagos_realizado2(idPago INT) RETURNS INT DETERMINISTIC
BEGIN

    DECLARE sumaPago INT;

    IF EXISTS(SELECT *
        FROM pago
        WHERE codigo_cliente = idPago) THEN

        SELECT SUM(total)
        INTO sumaPago
        FROM pago
        WHERE codigo_cliente = idPago;

    RETURN sumaPago;

    ELSE
        RETURN -1;
    END IF;
END//

DELIMITER ;

SELECT suma_pagos_realizado2(3);
-- Devuelve -1 si no se encuentra --
SELECT suma_pagos_realizado2(70);
```

```
-- EJ7 Realizar una función que me devuelva la suma de pagos que ha
realizado.
-- Pasa el código por parámetro. Controla en caso de que no se encuentre,
-- en ese caso devuelve un -1.--

DROP FUNCTION IF EXISTS suma_pagos_realizado2;

DELIMITER //
CREATE FUNCTION suma_pagos_realizado2(idPago INT) RETURNS INT DETERMINISTIC
BEGIN

	DECLARE sumaPago INT;

	IF EXISTS(SELECT *
		FROM pago
		WHERE codigo_cliente = idPago) THEN

		SELECT SUM(total)
		INTO sumaPago
		FROM pago
		WHERE codigo_cliente = idPago;

	RETURN sumaPago;

	ELSE
		RETURN -1;
	END IF;
END//
DELIMITER ;

SELECT suma_pagos_realizado2(3);
-- Devuelve -1 si no se encuentra --
SELECT suma_pagos_realizado2(70);
```

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. Pásale otro parámetro, si supera ese limite, lanzaremos una excepcion propia y devolveremos un 0

```
-- EJ8 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. Pásale otro parámetro, si supera ese limite,
-- lanzaremos una excepcion propia y devolveremos un 0. --

DROP FUNCTION IF EXISTS total_euros_pedido;

DELIMITER $$

CREATE FUNCTION total_euros_pedido(codPedido INT, limite NUMERIC(7,2)) RETURNS NUMERIC(7,2) DETERMINISTIC
BEGIN

    DECLARE totalEuros NUMERIC(7,2);


        SELECT SUM(cantidad*precio_unidad)
        INTO totalEuros
        FROM detalle_pedido dp
        WHERE codigo_pedido = codPedido;

    IF(totalEuros > limite) THEN
        SIGNAL SQLSTATE '45000' SET message_text='EL TOTAL DE EUROS HA SUPERADO EL LÍMITE';
        RETURN 0;
    ELSEIF(totalEuros IS NULL) THEN
        SIGNAL SQLSTATE '45000' SET message_text='NO EXISTEN PAGOS ASOCIADOS';
    ELSE
        RETURN totalEuros;
    END IF;
END$$

DELIMITER ;

SELECT total_euros_pedido(1, 10000);
-- Devuelve la excepción --
SELECT total_euros_pedido(4, 1000);
```

```
DROP FUNCTION IF EXISTS total_euros_pedido;


DELIMITER $$


CREATE FUNCTION total_euros_pedido(codPedido INT, limite NUMERIC(7,2))
RETURNS NUMERIC(7,2) DETERMINISTIC

BEGIN


      DECLARE totalEuros NUMERIC(7,2);
```

```
            SELECT SUM(cantidad*precio_unidad)
            INTO totalEuros
            FROM detalle_pedido dp
            WHERE codigo_pedido = codPedido;

      IF(totalEuros > limite) THEN
            SIGNAL SQLSTATE '45000' SET message_text='EL TOTAL DE EUROS HA
SUPERADO EL LÍMITE';
            RETURN 0;
      ELSEIF(totalEuros IS NULL) THEN
            SIGNAL SQLSTATE '45000' SET message_text='NO EXISTEN PAGOS
ASOCIADOS';
      ELSE
            RETURN totalEuros;
      END IF;
END$$

DELIMITER ;

SELECT total_euros_pedido(1, 10000);
-- Devuelve la excepción --
SELECT total_euros_pedido(4, 1000);
```

9. Realiza un resumen con informe de las estadísticas de los pedidos realizados por meses y por años.
(NIVEL S