Curso 2020/2021 Pablo L. Navarro

**PROYECTO “FREE PELOS”**

Contenido:

|  |  |
| --- | --- |
| 1.Presentación | Página 2 |
| 2.Modelo Entidad-Relación | Página 3 |
| 3.Modelo Relacional (Lógico) | Páginas 3 - 4 |
| 4.DDL | Páginas 5-10 |
| 5.DML | Páginas11-18 |
| 6.SQL | Páginas 19-25 |
| 7.PLPG SQL | Páginas 26-32 |
| 8. CONCLUSIONES | Página 33 |
|  |  |
|  |  |

DAM 1-1 Bases de Datos 3º Trimestre

1. **PRESENTACIÓN**

El objetivo del proyecto es organizar el trabajo de una peluquera autónoma que ofrece gran cantidad de servicios en diferentes lugares.

Lo más importante que se necesita registrar es la información relacionada con el cliente al que asignaremos un código, nombre, sexo, preferencias de servicio y teléfono; luego guardaremos la información del servicio que va a recibir el cual identificaremos con un nombre, descripción y precio preestablecido.

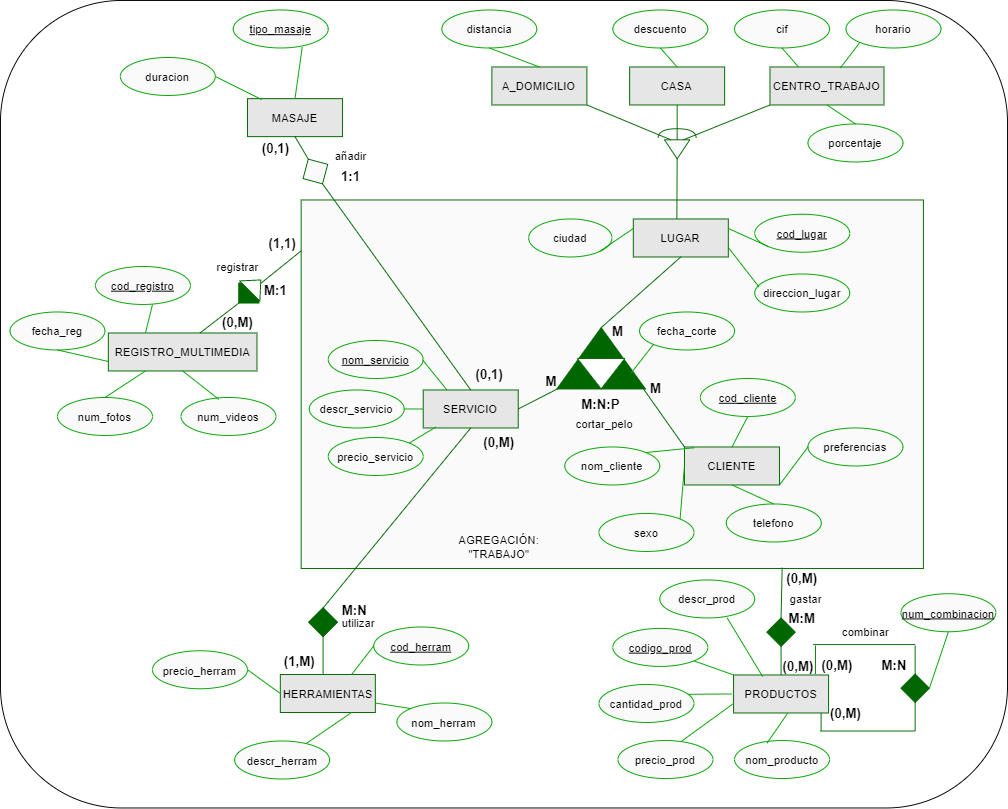
Lo siguiente a tener en cuenta es dónde se va dar el servicio con el cliente. Lo registraremos con un código, dirección, ciudad y si es a domicilio, en el hogar de la autónoma o en un centro de trabajo externo. En el caso de que el lugar del corte de pelo sea a domicilio registraremos la distancia recorrida desde el hogar, si el corte se produce en el hogar registraremos los descuentos que pueden conllevar y si es un centro de trabajo externo necesitaremos el cif de la empresa, el horario de apertura y el porcentaje con el que están negociados los servicios.

Un servicio concreto puede conllevar un masaje del que registraremos su tipo y duración, por tanto, un masaje concreto sólo puede estar vinculado a un servicio en especial probablemente más caro, dentro del servicio de corte va incluido el precio del masaje lo reciba o no el cliente puesto que el trabajador carece de titulación y necesita horas para practicar.

Casi siempre un trabajo o corte implica el gasto de uno o varios productos ya que a su vez pueden combinarse muchas veces entre si para conseguir las medidas deseadas. De cada producto nos interesa registrar su código, cantidad, descripción, precio y nombre genérico.

Por último, esporádicamente se hacen registros fotográficos de los trabajos para subir contenido a las redes sociales y publicitarse, cada registro llevará un código único ya que en un mismo corte se pueden hacer varios registros diferentes, guardaremos el número de fotos y vídeos que incluye cada uno.

1. **MODELO ENTIDAD – RELACIÓN**



1. **RELACIONAL**

|  |  |
| --- | --- |
| CLIENTE: | (cod\_cliente, preferencias, telefono, sexo, nom\_cliente) |
| PK: | (cod\_cliente) |
|  |  |
| LUGAR: | (cod\_lugar, direccion\_lugar, ciudad) |
| PK: | (cod\_lugar) |
|  |  |
| SERVICIO: | (nom\_servicio, descr\_servicio, precio\_servicio) |
| PK: | (nom\_servicio) |
|  |  |
|  |  |
| MASAJE: | (tipo\_masaje, duracion) |
| PK: | (tipo\_masaje) |
|  |  |
| AÑADIR\_MASAJE: | (tipo\_masaje, nom\_servicio) |
| PK: | (nom\_servicio,tipo\_masaje) |
| FK: | (nom\_servicio) → SERVICIO |
| FK: | (tipo\_masaje) → MASAJE |
|  |  |
|  |  |
| A\_DOMICILIO: | (cod\_lugar, distancia) |
| PK: | (cod\_lugar) |
| FK: | (cod\_lugar) → LUGAR |
|  |  |
| CASA: | (cod\_lugar, descuento) |
| PK: | (cod\_lugar) |
| FK: | (cod\_lugar) → LUGAR |
|  |  |
| CENTRO\_TRABAJO: | (cod\_lugar, cif, horario, porcentaje) |
| PK: | (cod\_lugar) |
| FK: | (cod\_lugar) → LUGAR |
|  |  |
|  |  |
| HERRAMIENTAS: | (cod\_herram, nom\_herram, descr\_herram, precio\_herram) |
| PK: | (cod\_herram) |
|  |  |
| UTILIZACION: | (nom\_servicio, cod\_herramienta) |
| PK: | (nom\_servicio, cod\_herramienta) |
| FK: | (nom\_servicio) → SERVICIO |
| FK: | (cod\_herramienta) → HERRAMIENTAS |
|  |  |
|  |  |
| PRODUCTOS: | (cod\_prod, descr\_prod, cantidad\_prod, precio\_prod, nom\_prod) |
| PK: | (cod\_prod) |
|  |  |
| COMBINAR: | (num\_combinacion, cod\_prod1, cod\_prod2) |
| PK: | (num\_combinacion, cod\_prod1, cod\_prod2) |
| FK: | (cod\_producto1, cod\_producto2) --> PRODUCTOS |
|  |  |
| CORTE\_PELO | (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio, cod\_producto, cod\_reg) |
| PK: | (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio) |
| FK: | (cod\_cliente) → CLIENTE |
| FK: | (cod\_lugar) → LUGAR |
| FK: | (nom\_servicio) → SERVICIO |
| FK: | (cod\_producto) → PRODUCTOS |
| FK: | (cod\_reg) → REGISTRO\_MULTIMEDIA |
|  |  |
| GASTAR: | (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio, cod\_producto) |
| PK: | (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio, cod\_producto) |
| FK: | (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio, cod\_producto) → CORTE\_PELO |
|  |  |
| REGISTRO\_MULTIMEDIA: | (cod\_registro, num\_fotos, num\_videos, fecha\_registro, cod\_cliente, cod\_lugar, nom\_servicio) |
| PK: | (cod\_registro, fecha\_registro) |
| FK: | (fecha\_registro, cod\_cliente, cod\_lugar, nom\_servicio) → CORTE\_PELO |
| VNN: | (fecha\_registro, cod\_cliente, cod\_lugar, nom\_servicio) |

1. **DDL**

CREATE TABLE cliente(

cod\_cliente SERIAL NOT NULL,

nombre\_cliente varchar (50),

telefono numeric (10),

preferencias varchar (100) DEFAULT 'Sin preferencias',

sexo varchar (10),

CONSTRAINT PK\_CLIENTE PRIMARY KEY (cod\_cliente)

);

CREATE TABLE lugar(

cod\_lugar SERIAL NOT NULL,

direccion\_lugar varchar (100),

ciudad varchar (50),

CONSTRAINT PK\_LUGAR PRIMARY KEY (cod\_lugar)

);

CREATE TABLE servicio(

nom\_servicio varchar (50) NOT NULL,

descr\_servicio varchar (100) DEFAULT 'Sin descripción',

precio\_servicio numeric (5) CHECK (precio\_servicio >= 0),

CONSTRAINT PK\_SERVICIO PRIMARY KEY (nom\_servicio)

);

CREATE TABLE masaje (

tipo\_masaje varchar (50) NOT NULL,

duracion int CHECK (duracion >= 0), -- MINUTOS

CONSTRAINT PK\_MASAJE PRIMARY KEY (tipo\_masaje)

);

CREATE TABLE anyadir\_masaje (

tipo\_masaje varchar (50) NOT NULL,

nom\_servicio varchar (50) NOT NULL,

CONSTRAINT PK\_ANYADIR\_MASAJE PRIMARY KEY (tipo\_masaje, nom\_servicio),

CONSTRAINT FK\_ANYADIR\_TIPO FOREIGN KEY (tipo\_masaje)

REFERENCES masaje (tipo\_masaje),

CONSTRAINT FK\_ANYADIR\_SERVICIO FOREIGN KEY (nom\_servicio)

REFERENCES servicio (nom\_servicio)

);

CREATE TABLE a\_domicilio(

cod\_lugar int NOT NULL,

distancia numeric(5) CHECK (distancia >= 0),

CONSTRAINT PK\_DOMICILIO PRIMARY KEY (cod\_lugar),

CONSTRAINT FK\_DOMICILIO FOREIGN KEY (cod\_lugar)

REFERENCES lugar (cod\_lugar)

);

CREATE TABLE casa(

cod\_lugar int NOT NULL,

descuento numeric (5) CHECK (descuento >= 0),

CONSTRAINT PK\_CASA PRIMARY KEY (cod\_lugar),

CONSTRAINT FK\_CASA FOREIGN KEY (cod\_lugar)

REFERENCES lugar(cod\_lugar)

);

CREATE TABLE centro\_trabajo(

cod\_lugar int NOT NULL,

cif varchar(20) NOT NULL,

horario varchar (50) DEFAULT '09:00 - 21:00',

porcentaje numeric (5) CHECK (porcentaje >= 0),

CONSTRAINT PK\_CENTRO PRIMARY KEY (cod\_lugar),

CONSTRAINT FK\_CENTRO FOREIGN KEY (cod\_lugar)

REFERENCES lugar(cod\_lugar)

);

CREATE TABLE herramientas(

cod\_herram SERIAL NOT NULL,

nom\_herram varchar (50),

descr\_herram varchar (100) DEFAULT 'Sin descripción',

precio\_herram numeric (10) CHECK (precio\_herram >= 0),

CONSTRAINT PK\_HERRAMIENTAS PRIMARY KEY (cod\_herram)

);

CREATE TABLE utilizar(

nom\_servicio varchar(50) NOT NULL,

cod\_herram int NOT NULL DEFAULT 000,

CONSTRAINT PK\_UTILIZAR PRIMARY KEY (nom\_servicio, cod\_herram),

CONSTRAINT FK\_UTILIZAR\_SERVICIO FOREIGN KEY (nom\_servicio)

REFERENCES servicio (nom\_servicio),

CONSTRAINT FK\_UTILIZAR\_HERRAMIENTA FOREIGN KEY (cod\_herram)

REFERENCES herramientas (cod\_herram)

ON UPDATE CASCADE

ON DELETE SET DEFAULT

);

CREATE TABLE productos (

cod\_prod SERIAL NOT NULL,

nom\_prod varchar (150),

descr\_prod varchar (100) DEFAULT 'Sin descripción',

cantidad\_prod numeric (5) CHECK (cantidad\_prod >= 0),

precio\_prod numeric (5) CHECK (precio\_prod >= 0),

CONSTRAINT PK\_PRODUCTOS PRIMARY KEY (cod\_prod)

);

CREATE TABLE combinar (

num\_combinacion SERIAL NOT NULL, --Tengo que crear un pk adicional

cod\_prod1 int NOT NULL DEFAULT 0, --para poder repetir las combinaciones

cod\_prod2 int NOT NULL DEFAULT 0,

CONSTRAINT PK\_COMBINAR PRIMARY KEY (num\_combinacion, cod\_prod1, cod\_prod2),

CONSTRAINT FK\_COMBINAR1 FOREIGN KEY (cod\_prod1)

REFERENCES productos (cod\_prod)

ON UPDATE CASCADE

ON DELETE SET DEFAULT,

CONSTRAINT FK\_COMBINAR2 FOREIGN KEY (cod\_prod2)

REFERENCES productos (cod\_prod)

ON UPDATE CASCADE

ON DELETE SET DEFAULT

);

CREATE TABLE corte\_pelo(

fecha\_corte date CHECK(to\_char(fecha\_corte, 'YYYY') > '2017'),

cod\_cliente int NOT NULL,

cod\_lugar int NOT NULL,

nom\_servicio varchar (50) NOT NULL,

cod\_producto int DEFAULT '000',

CONSTRAINT PK\_CORTE\_PELO PRIMARY KEY (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio),

CONSTRAINT FK\_CORTE\_CLIENTE FOREIGN KEY (cod\_cliente)

REFERENCES cliente (cod\_cliente),

CONSTRAINT FK\_CORTE\_LUGAR FOREIGN KEY (cod\_lugar)

REFERENCES lugar (cod\_lugar),

CONSTRAINT FK\_CORTE\_SERVICIO FOREIGN KEY (nom\_servicio)

REFERENCES servicio (nom\_servicio),

CONSTRAINT FK\_CORTE\_PRODUCTO FOREIGN KEY (cod\_producto)

REFERENCES productos (cod\_prod)

ON UPDATE CASCADE

ON DELETE SET DEFAULT

);

CREATE TABLE registro\_multimedia(

cod\_registro SERIAL NOT NULL,

num\_fotos numeric(5) CHECK(num\_fotos >= 0),

num\_videos numeric(5) CHECK(num\_videos >= 0),

fecha\_registro date CHECK(to\_char(fecha\_registro, 'YYYY') > '2017'),

cod\_cliente int,

cod\_lugar int,

nom\_servicio varchar(50),

CONSTRAINT PK\_REGISTRO\_MULTIMEDIA PRIMARY KEY (cod\_registro),

CONSTRAINT FK\_REGISTRO\_MULTIMEDIA FOREIGN KEY (fecha\_registro, cod\_cliente, cod\_lugar, nom\_servicio)

REFERENCES corte\_pelo (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio)

ON UPDATE CASCADE

ON DELETE SET NULL

);

CREATE TABLE gastar (

fecha\_corte date CHECK(to\_char(fecha\_corte, 'YYYY') > '2017'),

cod\_cliente int,

cod\_lugar int,

cod\_servicio varchar (50),

cod\_producto int DEFAULT '000',

CONSTRAINT PK\_GASTAR PRIMARY KEY (fecha\_corte, cod\_cliente, cod\_lugar, cod\_servicio, cod\_producto),

CONSTRAINT FK\_GASTAR\_CORTE FOREIGN KEY (fecha\_corte, cod\_cliente, cod\_lugar, cod\_servicio)

REFERENCES corte\_pelo (fecha\_corte, cod\_cliente, cod\_lugar, nom\_servicio)

ON DELETE SET NULL

ON UPDATE CASCADE,

CONSTRAINT FK\_GASTAR\_PRODUCTO FOREIGN KEY (cod\_producto)

REFERENCES productos (cod\_prod)

ON DELETE SET DEFAULT ON UPDATE CASCADE );

1. **DML**

--------TABLA CLIENTE--------

INSERT INTO cliente (nombre\_cliente, telefono, preferencias, sexo)

VALUES

('Misis One', 1234567, 'Corte y Tinte', 'M'),

('Don Two', 4567473, 'Rapado al 0', 'H'),

('Lady Three', 45645645, 'Corte a la antigua', 'M'),

('Mister Four', 45688942, 'Rapado Demoníaco', 'H'),

('Sir Five', 23476794, 'Corte Militar', 'H');

INSERT INTO cliente (nombre\_cliente, telefono)

VALUES

('Queen Six', 389432),

('Maese Seven', 795950),

('Duquesa Eight', 079067456),

('Nine-Chan', 067453623),

('Hokage Ten', 0674567);

--------TABLA SERVICIO--------

INSERT INTO servicio

VALUES

('Corte Simple', 'Lavar y cortar', 20),

('Corte Doble', 'Lavar y cortar 2 pax', 30),

('Corte Especial', 'Secado especial', 40),

('Corte y Tinte', 'Pr de gama alta', 50),

('Corte Eco', 'poco pelo',10);

INSERT INTO servicio (nom\_servicio, precio\_servicio)

VALUES

('Corte Samurai', 100),

('Permanente', 200),

('Corte Laser', 300),

('Japonés', 150),

('Ochentero', 60);

--------TABLA LUGAR--------

INSERT INTO lugar

VALUES (000, 'Direccion Casa', 'Benidorm');

INSERT INTO lugar (direccion\_lugar, ciudad)

VALUES

('Calle Stephen King 19', 'Misery City'),

('Calle Brandom Sanderson 14', 'Ciudad de las Brumas'),

('Calle Patrick Rothfuss 33', 'Ciudad del Silencio'),

('Calle Shinigami', 'Tokio'),

('Complejo Umbrella', 'Raccon City'),

('Mansion Hellsing', 'New York'),

('Marine Ford', 'Grand Line'),

('Calle Ichigo', 'Karakura Town'),

('Calle Songbird 3', 'Columbia');

--------TABLA MASAJE--------

INSERT INTO masaje

VALUES

('Completo', 60),

('Medio', 30),

('Piernas', 20),

('Brazos', 20),

('Cabeza', 15),

('Vietnamita', 90),

('Turco', 45),

('Sueco', 35),

('Shiatsu', 60),

('Lumbar', 15);

--------TABLA AGREGAR\_MASAJE--------

INSERT INTO anyadir\_masaje

VALUES

('Completo', 'Corte Especial'),

('Cabeza', 'Corte Simple'),

('Piernas', 'Corte Doble'),

('Brazos', 'Corte y Tinte'),

('Medio', 'Corte Eco'),

('Shiatsu', 'Japonés'),

('Cabeza', 'Corte Especial'),

('Sueco','Ochentero'),

('Lumbar','Permanente'),

('Vietnamita', 'Corte Laser');

--------TABLA CASA-------

INSERT INTO casa

VALUES (0, 15);

--Si el lugar es casa habra 15%

--------TABLA A\_DOMICILIO-------

INSERT INTO a\_domicilio

VALUES

(1, 50),

(2, 100),

(3, 300),

(4, 900),

(8, 500),

(9, 800);

--------TABLA CENTRO\_TRABAJO-------

INSERT INTO centro\_trabajo

VALUES

(5, 'TVIRUSZ00MB1', '00:00 - 08:00', 60),

(7, 'ECH1R00DA', '02:00 - 22:00', 40);

INSERT INTO centro\_trabajo (cod\_lugar, cif, porcentaje)

VALUES (6, '41UC4RD', 50);

--------TABLA HERRAMIENTAS--------

INSERT INTO herramientas

VALUES (000, 'Borrado u Obsoleto', 'Herramienta no disponible, borrada u obsoleta', '0');

INSERT INTO herramientas (nom\_herram, precio\_herram)

VALUES

('Tijeras Schvarosky', 400),

('Harusame', 900),

('Longclaw', 300),

('Masamune', 1000),

('X-Gun', 200),

('Secador Solar', 100),

('Andúril', 3000),

('Oathkeeper', 200),

('Rizador de Saúco', 900);

--------TABLA UTILIZAR--------

INSERT INTO utilizar

VALUES

('Corte Eco', 1),

('Corte Eco', 9),

('Corte Especial', 3),

('Corte Doble', 7),

('Permanente', 7),

('Japonés', 4),

('Japonés', 5),

('Ochentero', 2),

('Corte Samurai', 4),

('Corte y Tinte', 8);

--------TABLA PRODUCTOS--------

INSERT INTO productos(cod\_prod, nom\_prod, descr\_prod)

VALUES (000, 'Producto Obsoleto', 'El producto ya no existe o no se utiliza');

INSERT INTO productos (nom\_prod, cantidad\_prod, precio\_prod)

VALUES

('Tinte Revlon', 5, 20),

('Decolorador Natural', 2, 30),

('Tinte Schwarzkopzt', 10, 40),

('Tinte Loreal', 20, 15),

('Reparador Loreal', 50, 10),

('Mascarilla Natural', 10, 30),

('Eliminador de Residuios Schwarzkopzt', 5, 25),

('Champu Anticaida Revlon', 8, 8),

('Antigrasa Loreal', 5, 14);

--En la tabla combinar y gastar no hago inserts aun, hare un trigger

--para que cada insert disminuya la cantidad de los productos;

--------TABLA CORTE\_PELO--------

INSERT INTO corte\_pelo

VALUES

('05-04-2021', 1, 1, 'Corte Eco', 1),

('19-07-2019', 2, 2, 'Corte Simple', 2),

('11-09-2020', 3, 3, 'Corte Simple', 3),

('18-11-2019', 4, 4, 'Corte Especial', 4),

('01-02-2019', 5, 5, 'Permanente', 5),

('02-02-2021', 6, 6, 'Corte Doble', 6),

('08-12-2019', 7, 7, 'Ochentero', 7),

('10-07-2018', 8, 8, 'Corte Samurai', 8),

('26-01-2019', 9, 9, 'Japonés', 9),

('15-11-2020', 1, 2, 'Japonés', 3),

('22-09-2018', 4, 5, 'Corte y Tinte', 6),

('09-07-2020', 7, 8, 'Corte Especial', 9);

--------TABLA REGISTRO\_MULTIMEDIA--------

INSERT INTO registro\_multimedia (num\_fotos, num\_videos, fecha\_registro, cod\_cliente, cod\_lugar, nom\_servicio)

VALUES

(20,5,'19-07-2019', 2, 2, 'Corte Simple'),

(2,0,'11-09-2020', 3, 3, 'Corte Simple'),

(50,18,'18-11-2019', 4, 4, 'Corte Especial'),

(10,1,'01-02-2019', 5, 5, 'Permanente'),

(14,12,'02-02-2021', 6, 6, 'Corte Doble'),

(30,6,'10-07-2018', 8, 8, 'Corte Samurai'),

(20,1,'26-01-2019', 9, 9, 'Japonés'),

(0,10,'15-11-2020', 1, 2, 'Japonés'),

(0,2,'22-09-2018', 4, 5, 'Corte y Tinte'),

(15,3,'09-07-2020', 7, 8, 'Corte Especial');

1. **SQL**

------ (A) 5 CONSULTAS SIMPLES DE UNA SOLA TABLA ------

-- Muestra todo el contenido de la tabla mas importante sin ninguna modificacion,

-- quiero mostrarla asi de primeras para luego ir ensenyando lo que voy haciendo.

SELECT \* FROM corte\_pelo;

-- Muestra el codigo 0, esta fila no se podra eliminar (por triggers), de esta manera

-- podremos eliminar cualquier producto y los FK de otras tablas pasaran a referenciar al 0

-- evitando asi posibles errores o perdida de informacion.

SELECT cod\_prod, nom\_prod, descr\_prod

FROM productos

WHERE cod\_prod = 0;

-- Como en el caso anterior, ocurre lo mismo con las herramientas

SELECT cod\_herram, nom\_herram, descr\_herram

FROM herramientas

WHERE nom\_herram LIKE 'B%';

-- Acota los registros con un determinado numero de fotos o videos

SELECT num\_fotos, num\_videos, fecha\_registro, nom\_servicio

FROM registro\_multimedia

WHERE num\_fotos > 20

OR num\_videos > 10;

-- Muestra los diferentes tipos de sexo que se tienen en cuenta

SELECT DISTINCT sexo FROM cliente;

-----------------------------------------------------------------------------------------------------------------------------------------------------------

------ (B) 2 UPDATES Y 2 DELETES EN CUALQUIER TABLA ------

UPDATE cliente SET sexo = 'X' WHERE sexo IS NULL; -- La mitad de los inserts estan sin sexo

UPDATE herramientas SET cod\_herram = 333 WHERE cod\_herram = 7;

-- Desencadena UPDATE CASCADE en UTILIZAR

DELETE FROM productos WHERE cod\_prod = 1;

-- Desencadena DELETE SET DEFAULT en CORTE\_PELO y GASTAR

DELETE FROM corte\_pelo WHERE fecha\_corte BETWEEN '2019-06-19' AND '2019-08-19';

-- Desencadena DELETE SET NULL en REGISTRO\_MULTIMEDIA y GASTAR

----------------------------------------------------------------------------------------------------------------------------------------------------------- ------ (C) 3 CONSULTAS CON MAS DE UNA TABLA ------

-- Muestra el nombre de las herramientas, el precio y el servicio donde se usan ordenadas por precio

SELECT h.nom\_herram AS herramienta, h.precio\_herram AS precio\_euros, u.nom\_servicio AS servicio\_donde\_se\_usa

FROM herramientas h, utilizar u

WHERE h.cod\_herram = u.cod\_herram

AND h.precio\_herram > 800

ORDER BY h.precio\_herram DESC;

-- Muestra la distancia desde casa a las direcciones y ciudades mas lejanas, ordenado por distancia

SELECT d.distancia AS km, l.ciudad, l.direccion\_lugar AS direccion

FROM a\_domicilio d, lugar l

WHERE d.cod\_lugar = l.cod\_lugar

AND d.distancia > 50

ORDER BY km DESC;

-- Muestra los cortes a partir del 2020 ordenados por fecha, aparece nombre del cliente,

-- su direccion y ciudad, el producto utilizado y el servicio que consumio.

SELECT cp.fecha\_corte AS fecha, cl.nombre\_cliente AS cliente, lu.ciudad,

lu.direccion\_lugar AS direccion, pr.nom\_prod AS producto\_usado, cp.nom\_servicio AS servicio

FROM corte\_pelo cp, cliente cl, lugar lu, productos pr

WHERE cp.cod\_cliente = cl.cod\_cliente

AND cp.cod\_lugar = lu.cod\_lugar

AND cp.cod\_producto = pr.cod\_prod

AND TO\_CHAR(cp.fecha\_corte, 'YYYY') >= '2020'

ORDER BY fecha;

----------------------------------------------------------------------------------------------------------------------------------------------------------- ------ (D) 3 CONSULTAS USANDO FUNCIONES ------

-- Media de todos los masajes

SELECT AVG (duracion) AS media\_minutos\_masajes

FROM masaje;

-- Total de clientes registrados

SELECT COUNT(\*) as clientes\_registrados

FROM cliente;

-- Muestra el coste del servicio y el producto de los anyos seleccionados

SELECT SUM(se.precio\_servicio + pr.precio\_prod) AS ganancia\_bruto

FROM corte\_pelo cp, servicio se, productos pr

WHERE cp.cod\_producto = pr.cod\_prod

AND cp.nom\_servicio = se.nom\_servicio

AND TO\_CHAR(cp.fecha\_corte, 'YYYY') IN ('2018','2020');

----------------------------------------------------------------------------------------------------------------------------------------------------------- ------ (E) 2 CONSULTAS USANDO GROUP BY ------

-- Muestra el total obtenido por servicio y el numero de veces que se ha realizado

SELECT cp.nom\_servicio, SUM(se.precio\_servicio) AS total\_por\_servicio, COUNT(cp.nom\_servicio) AS veces

FROM corte\_pelo cp, servicio se

WHERE cp.nom\_servicio = se.nom\_servicio

GROUP BY cp.nom\_servicio

ORDER BY total\_por\_servicio DESC;

-- Muestra el total obtenido por cada cliente por servicio contar el gasto de productos

SELECT c.nombre\_cliente, s.precio\_servicio, s.nom\_servicio

FROM cliente c, servicio s, corte\_pelo p

WHERE s.precio\_servicio > 10

AND c.cod\_cliente = p.cod\_cliente

AND s.nom\_servicio = p.nom\_servicio

GROUP BY c.nombre\_cliente, s.precio\_servicio, s.nom\_servicio

ORDER BY s.precio\_servicio DESC;

----------------------------------------------------------------------------------------------------------------------------------------------------------- ------ (F) 2 CONSULTAS UTILIZANDO SUBCONSULTAS ------

-- Muestra qué tipo de masaje incluyó cada servicio realizado

SELECT \*

FROM anyadir\_masaje

WHERE nom\_servicio IN (SELECT nom\_servicio FROM corte\_pelo);

-- Muestra el nombre del cliente que tuvo el servicio más barato (3 subconsultas anidadas)

SELECT nombre\_cliente AS cliente\_mas\_tacanyo

FROM cliente

WHERE cod\_cliente =

(SELECT cod\_cliente FROM corte\_pelo WHERE nom\_servicio =

(SELECT nom\_servicio FROM servicio WHERE precio\_servicio =

(SELECT MIN(precio\_servicio) FROM servicio)));

----------------------------------------------------------------------------------------------------------------------------------------------------------- ------ (G) 2 CONSULTAS UTILIZANDO GROUP BY y HAVING ------

-- Muestra el total por servicio de los servicios que al menos se hayan realizado 2 veces

SELECT cp.nom\_servicio, SUM(se.precio\_servicio) AS total\_por\_servicio, COUNT(cp.nom\_servicio) AS veces

FROM corte\_pelo cp, servicio se

WHERE cp.nom\_servicio = se.nom\_servicio

GROUP BY cp.nom\_servicio

HAVING COUNT(cp.nom\_servicio) >= 2;

-- Muestra el total de gasto por productos de los productos que al menos se hayan gastado 2 veces

SELECT pr.nom\_prod AS producto, SUM(pr.precio\_prod) AS total\_gasto, COUNT(cp.cod\_producto) AS veces\_usado

FROM productos pr, corte\_pelo cp

WHERE pr.cod\_prod = cp.cod\_producto

GROUP BY pr.nom\_prod

HAVING COUNT(cp.cod\_producto) >= 2;

-----------------------------------------------------------------------------------------------------------------------------------------------------------

------ (H) 3 UPDATES USANDO SUBCONSULTAS EN WHERE Y SET ------

-- Copia las preferencias del cliente con pk 5 en el 9

UPDATE cliente

SET preferencias = (

SELECT preferencias

FROM cliente

WHERE cod\_cliente = 5

)

WHERE nombre\_cliente = (

SELECT nombre\_cliente

FROM cliente

WHERE cod\_cliente = 9

);

-- Cambia el pk de herramienta 4 por 60; Desencadena UPDATE CASCADE en UTILIZAR

UPDATE herramientas

SET cod\_herram = (

SELECT (cod\_prod \* 10)

FROM productos

WHERE nom\_prod = 'Mascarilla Natural'

)

WHERE cod\_herram = (

SELECT cod\_cliente

FROM corte\_pelo

WHERE fecha\_corte = '2019-11-18');

-- Cambia el cod\_prod de 5 a 80; Desencadena UPDATE CASCADE en CORTE\_PELO

UPDATE productos

SET cod\_prod = (

SELECT (cod\_cliente \* 20)

FROM corte\_pelo

WHERE fecha\_corte = '2019-11-18'

)

WHERE cod\_prod = (

SELECT cod\_lugar

FROM corte\_pelo

WHERE nom\_servicio = 'Permanente'

AND cod\_cliente = 5

);

1. PLPG SQL

------------------------------------------------ 2 VISTAS ---------------------------------------------------------------

-- Esta primera vista es una mejora de la tabla corte\_pelo donde sustitituimos los pk's que protegen los datos por

-- otros de interes directo, referenciamos los vinculos con el resto de tablas y anyadimos el calculo del total

-- bruto por servicio de manera que para obtener el neto de cada uno de los servicios completos solo faltaria

-- aplicar descuentos o incrementos dependiendo de si el lugar es a domicilio, en un centro de trabajo o en

-- la casa del trabajador, pero para ello necesitaremos utilizar funciones;

CREATE VIEW precio\_corte\_bruto (fecha, codigo, nombre\_cliente, direccion, ciudad, servicio, precio\_servicio,

producto, precio\_producto, total\_bruto ) AS

SELECT cp.fecha\_corte, cl.cod\_cliente, cl.nombre\_cliente, lu.direccion\_lugar, lu.ciudad, se.nom\_servicio,

se.precio\_servicio, pr.nom\_prod, pr.precio\_prod, (pr.precio\_prod + se.precio\_servicio) AS total\_bruto

FROM corte\_pelo cp, cliente cl, lugar lu, productos pr, servicio se

WHERE cp.cod\_cliente = cl.cod\_cliente

AND cp.cod\_lugar = lu.cod\_lugar

AND cp.nom\_servicio = se.nom\_servicio

AND cp.cod\_producto = pr.cod\_prod;

SELECT \* FROM precio\_corte\_bruto;

--En esta segunda vista mostramos el masaje que puede incluir gratuitamente cada servicio con su duracion;

CREATE VIEW masaje\_precio (servicio, masaje, duracion, precio) AS

SELECT se.nom\_servicio, ma.tipo\_masaje, ma.duracion, se.precio\_servicio

FROM masaje ma, servicio se, anyadir\_masaje am

WHERE se.nom\_servicio = am.nom\_servicio

AND ma.tipo\_masaje = am.tipo\_masaje;

SELECT \* FROM masaje\_precio;

------------------------------------------------ 2 FUNCIONES ---------------------------------------------------------------

-- A la primera funcion le pasaremos como parametro el id de un cliente y la fecha donde se realizo el corte y nos devolvera

-- el total del precio con los descuentos o incrementos aplicados dependiendo del lugar donde se realizo; Codigo comentado;

CREATE OR REPLACE FUNCTION precio\_neto (cliente\_id int, fecha\_servicio date)

RETURNS numeric AS $$

-- GUARDAMOS EL BRUTO Y COD\_LUGAR DE LA VISTA 1 Y CORTE\_PELO PARA EL CLIENTE Y FECHA ELEGIDOS;

DECLARE bruto numeric := (SELECT total\_bruto FROM precio\_corte\_bruto

WHERE cliente\_id = codigo AND fecha\_servicio = fecha);

DECLARE id\_lugar int := (SELECT cod\_lugar FROM corte\_pelo

WHERE cliente\_id = cod\_cliente AND fecha\_servicio = fecha\_corte);

-- PREPARAMOS VARIABLES PARA GUARDAR LOS RESULTADOS;

DECLARE tasas numeric;

DECLARE neto numeric;

BEGIN

-- IF 1º COMPRUEBA SI EL SERVICIO FUE EN CASA Y APLICA DESCUENTO;

IF id\_lugar = 0 THEN

tasas := (bruto \* 0.15);

neto := (bruto - tasas);

RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE EN CASA',cliente\_id, fecha\_servicio;

-- IF 2º COMPRUEBA SI EL SERVICIO FUE A DOMICILIO Y APLICA INCREMENTO POR KM;

ELSIF EXISTS ( SELECT cod\_lugar FROM a\_domicilio

WHERE cod\_lugar = id\_lugar ) THEN

tasas := ((SELECT distancia FROM a\_domicilio WHERE cod\_lugar = id\_lugar) \* 0.20);

neto := (bruto + tasas);

RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE A DOMICILIO',cliente\_id, fecha\_servicio;

-- IF 3º COMPRUEBA SI EL SERVICIO FUE EN CENTRO DE TRABAJO Y APLICA PORCENTAJES;

ELSIF EXISTS ( SELECT cod\_lugar FROM centro\_trabajo

WHERE cod\_lugar = id\_lugar ) THEN

tasas := (bruto \* ((100 - (SELECT porcentaje FROM centro\_trabajo WHERE cod\_lugar = id\_lugar))/100));

neto := TRUNC((bruto - tasas),2);

RAISE INFO 'EL SERVICIO DE CLIENTE % Y FECHA % FUE EN UN CENTRO DE TRABAJO',cliente\_id, fecha\_servicio;

-- SI NO EXISTE EN NINGUN LUGAR LOS PARAMETROS NO SON VALIDOS, INFORMAMOS Y DEVOLVEMOS 0.0

ELSE

RAISE INFO 'NO EXISTE REGISTRO CON CODIGO DE CLIENTE % Y FECHA %',cliente\_id, fecha\_servicio;

RETURN 0.0;

END IF;

-- SI EXISTE PERO EL CALCULO ES NULL SE ELIMINO EL PRODUCTO O SERVICIO DEL REGISTRO, INFORMAMOS Y DEVOLVEMOS 0.0

IF neto IS NULL THEN

RAISE INFO 'EL SERVICIO O PRODUCTO FUERON ELIMINADOS PARA EL CLIENTE % Y FECHA %',cliente\_id, fecha\_servicio;

RETURN 0.0;

END IF;

RETURN neto;

END;

$$ LANGUAGE PLPGSQL;

SELECT precio\_neto (8, '2018-07-10'); -- DEVUELVE DATO (A\_DOMICILIO);

SELECT precio\_neto (5, '2019-02-01'); -- DEVUELVE DATO (CENTRO\_TRABAJO);

SELECT precio\_neto (1, '2018-04-05'); -- DEVUELVE 0.0 Y MENSAJE DE ELIMINADO;

SELECT precio\_neto (4, '2010-01-01'); -- DEVUELVE 0.0 INDICANDO QUE NO EXISTE REGISTRO;

-- Ahora aprovechamos la funcion para crear otra y recorrer con un cursor la vista entre los anyos deseados obteniendo

-- todos los beneficios netos y la suma total de estos los acumulamos en otra variable que agregamos tras el bucle;

CREATE OR REPLACE FUNCTION get\_registros (anyo\_inicio varchar, anyo\_fin varchar)

RETURNS text AS $$

DECLARE registros\_salida text DEFAULT '';

DECLARE total\_beneficio numeric DEFAULT 0;

DECLARE record\_registros RECORD;

DECLARE cursor\_registros CURSOR

FOR SELECT fecha, nombre\_cliente, codigo, servicio, total\_bruto

FROM precio\_corte\_bruto

WHERE TO\_CHAR(fecha, 'YYYY') BETWEEN anyo\_inicio AND anyo\_fin;

BEGIN

OPEN cursor\_registros;

RAISE WARNING 'Como invocamos dos veces la funcion precio\_neto, todas las INFO estan duplicadas';

LOOP

FETCH cursor\_registros INTO record\_registros;

EXIT WHEN NOT FOUND;

registros\_salida := registros\_salida || ' ---- ' || record\_registros.fecha || '; '

|| record\_registros.nombre\_cliente || '; ' || record\_registros.servicio || '; TOTAL NETO => '

|| (SELECT precio\_neto ((SELECT cod\_cliente FROM corte\_pelo

WHERE record\_registros.codigo = cod\_cliente

AND record\_registros.fecha = fecha\_corte),

record\_registros.fecha)) || '€';

total\_beneficio := total\_beneficio + (SELECT precio\_neto ((SELECT cod\_cliente FROM corte\_pelo

WHERE record\_registros.codigo = cod\_cliente

AND record\_registros.fecha = fecha\_corte),

record\_registros.fecha));

END LOOP;

CLOSE cursor\_registros;

registros\_salida := registros\_salida || ' >>>>>>>>>>>> TOTAL DE GANANCIAS => ' || total\_beneficio || '€';

RETURN registros\_salida;

END;

$$ LANGUAGE PLPGSQL;

SELECT get\_registros ('2018', '2021');

SELECT get\_registros ('2020', '2020');

SELECT get\_registros ('2021', '2021');

------------------------------------------------ 2 TRIGGERS ---------------------------------------------------------------

--Trigger para asegurar que los deletes en productos no puedan afectar al codigo 0;

CREATE OR REPLACE FUNCTION eliminar\_producto()

RETURNS TRIGGER AS $$

DECLARE codigo int := OLD.cod\_prod;

BEGIN

IF codigo = 0 THEN

RAISE EXCEPTION 'TRIGGER: La entrada con codigo 0 no se puede borrar';

ELSE

DELETE FROM productos WHERE cod\_prod = NEW.cod\_prod;

END IF;

RETURN OLD;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER protege\_productos\_delete BEFORE DELETE

ON productos FOR EACH ROW

EXECUTE PROCEDURE eliminar\_producto();

DELETE FROM productos WHERE cod\_prod = 0;

DELETE FROM productos WHERE cod\_prod = 7;

DELETE FROM productos WHERE nom\_prod = 'Producto Obsoleto';

SELECT \* FROM productos;

-- Esta funcion y trigger se disparan al introducir dos codigos de producto dentro de la tabla combinar,

-- se comprueba si el producto existe, en caso afirmativo se suma cantidad al resultado y se resta de

-- los sumandos, en caso contrario se crea el nuevo producto y se anyade a la tabla productos;

CREATE OR REPLACE FUNCTION combinar\_productos()

RETURNS TRIGGER AS $$

DECLARE cod1 int := NEW.cod\_prod1;

DECLARE cod2 int := NEW.cod\_prod2;

DECLARE nom1 varchar := (SELECT nom\_prod FROM productos WHERE cod\_prod = cod1);

DECLARE nom2 varchar := (SELECT nom\_prod FROM productos WHERE cod\_prod = cod2);

DECLARE cant1 int := (SELECT cantidad\_prod FROM productos WHERE cod\_prod = cod1);

DECLARE cant2 int := (SELECT cantidad\_prod FROM productos WHERE cod\_prod = cod2);

DECLARE prec1 numeric := (SELECT precio\_prod FROM productos WHERE cod\_prod = cod1);

DECLARE prec2 numeric := (SELECT precio\_prod FROM productos WHERE cod\_prod = cod2);

DECLARE nom\_combinado varchar := nom1 || ' - ' || nom2;

DECLARE precio\_combinado numeric := prec1 + prec2;

BEGIN

IF cant1 > 0 AND cant2 > 0 THEN

IF EXISTS (SELECT nom\_prod FROM productos WHERE nom\_prod = nom\_combinado) THEN

RAISE INFO 'Trigger de combinacion que actualiza las cantidades en productos';

UPDATE productos SET cantidad\_prod = cantidad\_prod -1 WHERE cod\_prod = cod1;

UPDATE productos SET cantidad\_prod = cantidad\_prod -1 WHERE cod\_prod = cod2;

UPDATE productos SET cantidad\_prod = cantidad\_prod +1 WHERE nom\_prod = nom\_combinado;

ELSE

RAISE INFO 'Trigger de combinacion que anyade un nuevo producto y resta cantidades';

UPDATE productos SET cantidad\_prod = cantidad\_prod -1 WHERE cod\_prod = cod1;

UPDATE productos SET cantidad\_prod = cantidad\_prod -1 WHERE cod\_prod = cod2;

INSERT INTO productos (nom\_prod, descr\_prod, cantidad\_prod, precio\_prod)

VALUES (nom\_combinado, 'Combinacion', 1, precio\_combinado);

END IF;

ELSE

RAISE INFO 'Combinacion fallida, no hay suficientes productos para % ', nom\_combinado;

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER actualiza\_cantidades BEFORE INSERT

ON combinar FOR EACH ROW

EXECUTE PROCEDURE combinar\_productos();

INSERT INTO combinar (cod\_prod1, cod\_prod2) VALUES (0,5);

INSERT INTO combinar (cod\_prod1, cod\_prod2) VALUES (6,5);

INSERT INTO combinar (cod\_prod1, cod\_prod2) VALUES (5,4);

SELECT \* FROM productos;

SELECT \* FROM combinar;

1. **CONCLUSIONES**

Para empezar diré que he tenido muchos problemas para desarrollar el proyecto ya que el tema no es mi especialidad, cuando empecé no tenía la confianza suficiente como para desarrollar algo que controlara demasiado, pensé en hacer algo simple sin saber que la complejidad exigía que supiera mucho del tema. Como justificación diré que la base de datos era para otra persona muy cercana a mi al inicio del curso pero que en la tercera evaluación ya no forma parte de mi vida, pensé en empezar de cero pero acabé descartando la idea ya que tenía el modelo entidad relación y relacional ya formado, por tanto, esta base de datos ha terminado siendo algo útil solo a nivel técnico, que me ha enseñado practicando y probando a enrevesar las tablas cada vez más, y dejándome una gran sensación de querer empezar mi propio sistema sin depender de nadie.

En cuanto al apartado técnico, veo que falta mucho para que la base sea verdaderamente funcional, habría que añadir más triggers que sirvieran de restricciones para que no se actualicen los “códigos 0” que menciono a menudo, más funciones que permitieran extraer los datos de una manera más óptima y por supuesto utilizar el resto de tablas como gastar y herramientas que no he tenido espacio para hacer pero me he negado a eliminar, aunque a cada trimestre he tenido que revistar todo el contenido casi desde el principio, modificando relaciones, añadiendo nuevas pk´s para que no obtener problemas e incluso borrando atributos que en un principio consideraba indispensables.

Finalmente, aunque la base no sea lo más útil del mundo, funciona perfectamente y creo que he conseguido aplicar todos los conocimientos que he ido aprendiendo a lo largo del curso lo que me ha hecho asentar cada uno de los conceptos para empezar ahora mi propio proyecto.