Universidad de La Laguna

Proyecto final: Supuesto Inditex

Administración y diseño de bases de datos

Karina Kalwani Israni
alu0101109046@ull.edu.es

Noelia Ibáñez Silvestre
alu0101225555@ull.edu.es

Índice

Objetivo

El objetivo de este proyecto es el desarrollo de una base de datos, suponiendo que el cliente que nos pide desarrollarla es la multinacional española de confección textil conocida como Inditex. Aclarar que este trabajo se basa en un supuesto imaginario y no en un caso real. Se ha intentado centralizar la información relacionada, desde los clientes que realizan las compras, qué compran, dónde lo compran; hasta la fábrica donde se confeccionan esos productos.

Contexto de la base de datos

Supongamos que queremos realizar una base de datos para una gran empresa de fabricación y distribución textil como es Inditex. Esta empresa nos pide un seguimiento del stock de sus tiendas y empleados.

Inditex tiene en nómina a casi 152.000 empleados y opera más de 7.000 tiendas en los cinco continentes bajo las marcas principales de Zara, Zara Home, Massimo Dutti, Pull & Bear, Bershka, Oysho, Lefties y Stradivarius.

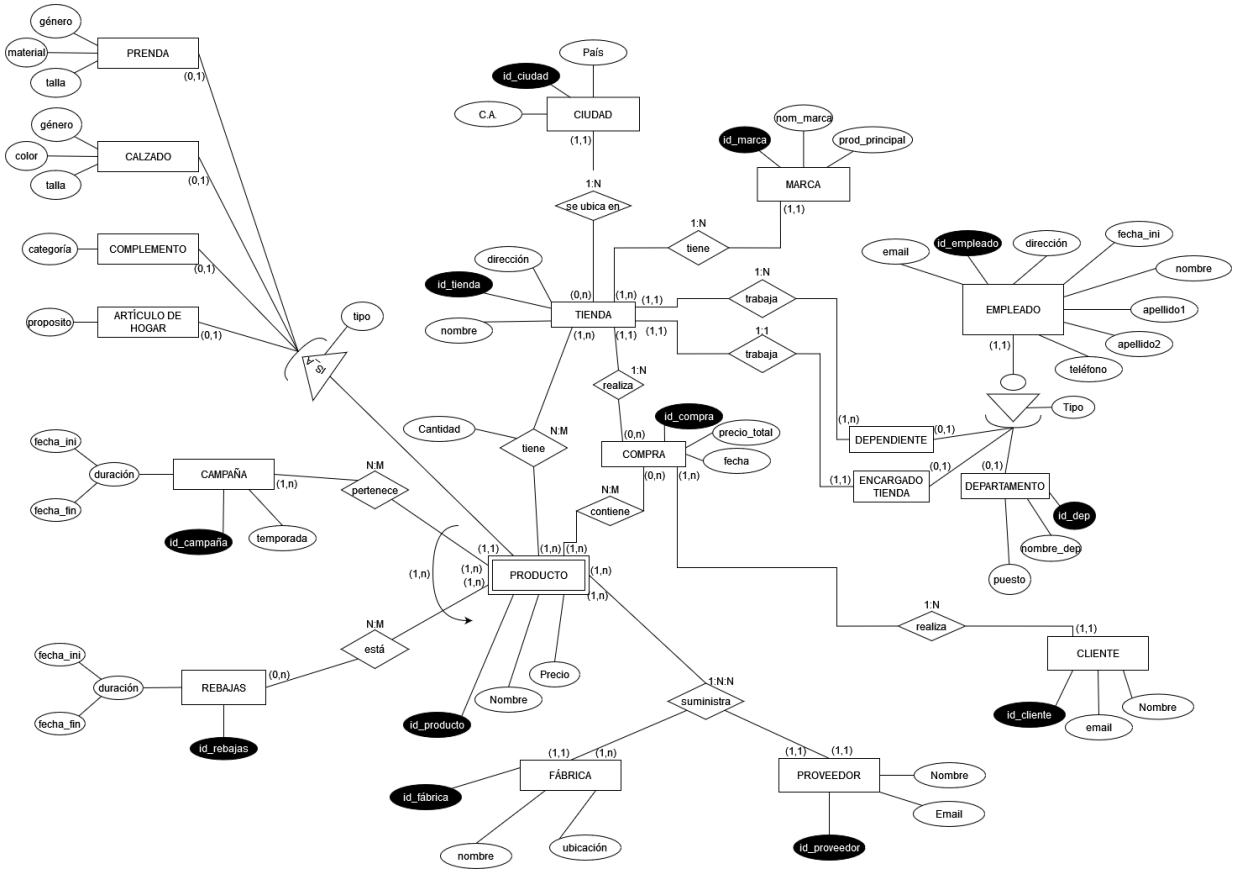
La propia empresa nos indica algunos requisitos a tener en cuenta:

- Un dependiente sólo puede trabajar para una tienda pero ésta tendrá varios.
- Una tienda sólo tendrá un encargado.
- El empleado podrá ser únicamente encargado, dependiente o estar trabajando para algún departamento.
- Una tienda podrá estar ubicada en una ciudad y, ésta, podrá tener N tiendas.
- Un producto se podrá clasificar en prenda, calzado, complemento o artículo de hogar. Además, cada categoría tendrá sus atributos específicos, pero también existirán algunos productos que no pertenezcan a ninguna de las anteriores categorías.
- Los productos formarán parte de una campaña, en la que hará referencia al tiempo en el que el producto está en disponible para compra en tienda.
- Además, podrán formar parte de las rebajas, pero para ello, previamente debe de haber estado disponible en tienda siendo parte de una campaña.
- Una fábrica pertenece a un proveedor y fabrica diferentes productos.
- Los proveedores suministrarán varios productos a las tiendas.
- Cada marca de la empresa tendrá tiendas distribuidas en ciudades.

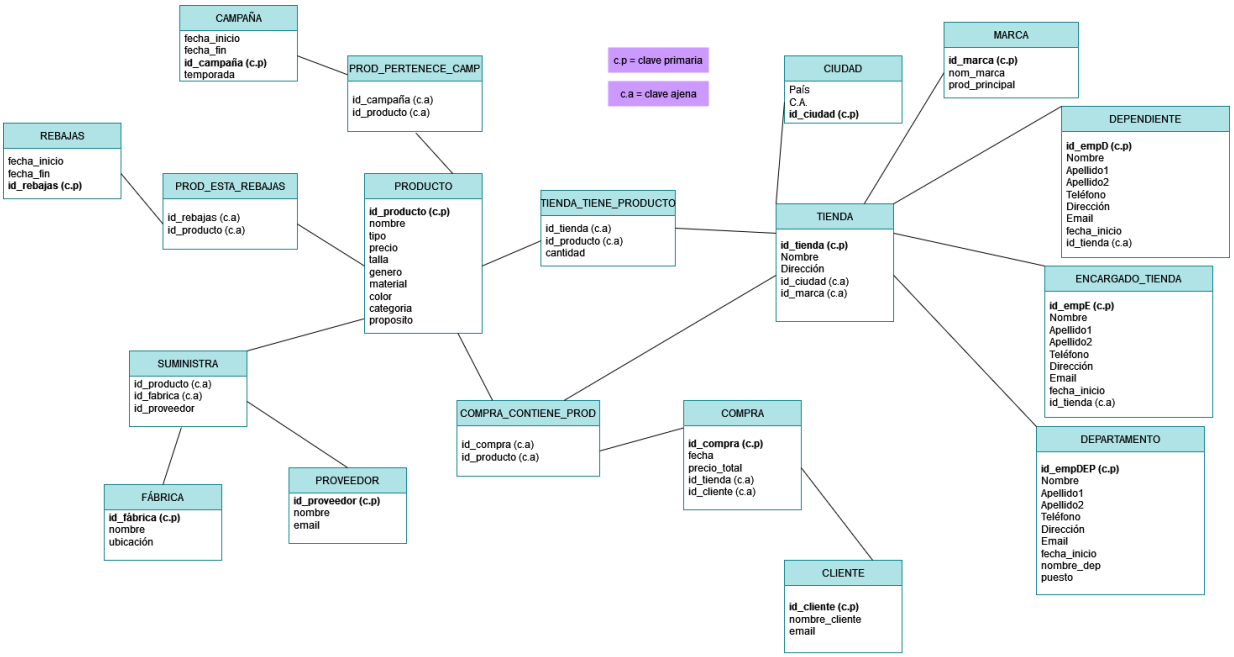
Como solución se deciden crear las siguientes entidades:

- Tienda (Zara, Zara Home, Oysho...): Son las tiendas registradas con su id, nombre y ubicación.
- Ciudad: Lugar donde está localizada la tienda.
- Empleado: Corresponde a los datos personales del empleado. Además, depende del puesto de este.
- Clientes: Datos de los clientes que realizaron una compra.
- Productos: Todos los productos de cada tienda. Un producto podrá ser una prenda, calzado, complemento o artículo de hogar (por ejemplo, Zara Home o cualquier otra tienda que tenga una pequeña sección dedicada a este tipo de productos).
- Prendas: prendas de ropa que se venden (camisetas, sudaderas, pantalones, camisas, vestidos...). Si una camiseta es de mujer, esta es exclusiva de esta. De igual manera si se trata de una prenda para hombre (es exclusiva de este).
- Calzado: la mayoría de las tiendas tienen una sección de calzados. Para cada calzado se tendrán los colores disponibles, la talla, etc
- Complementos: en esta categoría entrarán productos de bisutería, bufandas, gafas de sol, etc.
- Artículos de Hogar: todos los artículos de la tienda Zara Home, así como los de aquellas tiendas que tengan una pequeña sección dedicada a dichos productos.
- Proveedores: Los productos vendrán de un proveedor, y dicho proveedor tendrá una fábrica donde fabrica el producto
- Fábrica: Cada proveedor tendrá N fábricas donde fabricará productos para Inditex.
- Compras: Compras realizadas por los clientes.
- Campañas: esta tabla hará referencia a las campañas otoño, invierno, primavera y verano, incluso campaña navidad, es decir, los cambios de estaciones en las que la ropa de las tiendas cambia para adecuarse al tiempo, ó la mercancía de la tienda se cambia para adecuarse a la época. Los complementos (algunos de ellos) serán comunes a todas las campañas (exceptuando, por ejemplo, guantes de invierno, paraguas...).
- Rebajas: rebajas invierno y rebajas de verano; para que sea un producto esté disponible en una rebaja de invierno, previamente debería de haber sido un artículo sin rebajar de invierno (u otoño) y si es una rebaja de verano, previamente debe de haber sido un artículo de verano (o primavera).
- Dependiente: Empleado dependiente de la tienda.
- Encargado de tienda: Empleado encargado de la tienda.
- Departamento: Empleado que tiene un puesto en un determinado departamento.

Modelo Entidad-Relación



Modelo Relacional



Base de Datos en PostgreSQL

Para la implementación de la base de datos en el Sistema Gestor PostgreSQL, en primer lugar hemos creado una base de datos nueva. Para ejecutar el script SQL hay que usar el siguiente comando:

psql --dbname proyecto -f script.sql

donde *proyecto* es el nombre que se le da a la bbdd creada anteriormente.

Para la Clave Primaria de todas las tablas hemos optado por usar SEQUENCE, de tal manera que, cada vez que se inserte un elemento nuevo en la tabla, se autoincremente el valor del ID (clave primaria) de la fila nueva.

```
CREATE SEQUENCE public.ciudad_id_ciudad_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
CREATE TABLE ciudad (
  id_ciudad integer PRIMARY KEY DEFAULT nextval('public.ciudad_id_ciudad_seq'::regclass) NOT NULL,
  c_a VARCHAR(200) NOT NULL,
  pais VARCHAR(100) NOT NULL,
  CONSTRAINT unique_vals
  UNIQUE (c_a, pais)
);
```

En el script SQL hemos usado funciones específicas, como por ejemplo las restricciones CHECK, UNIQUE y TRIGGER.

UNIQUE

La restricción UNIQUE impide la duplicación de claves no primarias, es decir, se especifica que dos registros no puedan tener el mismo valor en un campo. Esta restricción se usa cuando ya se estableció una clave primaria pero se necesita asegurar que otros campos también sean únicos.

En el ejemplo anterior se puede observar que se ha usado la restricción UNIQUE, por lo que cuando se intente insertar una Comunidad Autónoma y País ya existente nos saltará un mensaje de error.

```
INSERT INTO ciudad
VALUES(DEFAULT, 'canarias', 'españa');

INSERT INTO ciudad
VALUES(DEFAULT, 'madrid', 'españa');

-- No dejara añadir la siguiente fila ya que hemos puesto la restriccion de UNIQUE
INSERT INTO ciudad
VALUES(DEFAULT, 'madrid', 'españa');
```

```
psql:script.sql:57: ERROR:  duplicate key value violates unique constraint "unique_vals"
DETAIL:  Key (c_a, pais)=(madrid, españa) already exists.
 id_ciudad |   c_a    |  pais
-----------+----------+--------
         1 | canarias | españa
         2 | madrid   | españa
(2 rows)
```

En este caso se puede ver como no nos dejó insertar (Madrid, España) porque ya lo habíamos hecho anteriormente.

CHECK

La restricción CHECK especifica los valores que acepta un campo, evitando que se inserten valores inapropiados ó verificando, como en este caso, los datos que se desean insertar (también se podría aplicar para las actualizaciones).

Un ejemplo de esto se produce en la tabla producto en la que hacemos uso de ellos para poder insertar productos en función del tipo que es este, con sus correspondientes atributos.

```
CHECK((tipo = 'prenda' AND
        talla IS NOT NULL AND
        genero IS NOT NULL AND
        material IS NOT NULL AND
        color IS NULL AND
        categoria IS NULL AND
        proposito IS NULL) OR
      (tipo = 'calzado' AND
        genero IS NOT NULL AND
        color IS NOT NULL AND
        talla IS NOT NULL AND
        material IS NULL AND
        categoria IS NULL AND
        proposito IS NULL) OR
```

```
(tipo = 'complemento' AND
  talla IS NULL AND
  genero IS NULL AND
  material IS NULL AND
  color IS NULL AND
  categoria IS NOT NULL AND
  proposito IS NULL) OR
(tipo = 'articulo de hogar' AND
  talla IS NULL AND
  genero IS NULL AND
  material IS NULL AND
  color IS NULL AND
  categoria IS NULL AND
  proposito IS NOT NULL) OR
(tipo IS NULL))
```

```
INSERT INTO producto
VALUES(DEFAULT, 'Camisa hombre', 6.99, 'prenda', 'M', 'H', 'algodon', NULL, NULL, NULL);

INSERT INTO producto
VALUES(DEFAULT, 'Tacones rojo', 12, 'calzado', '40', 'M', NULL, 'rojo');

-- Deja añadir sin poner tipo ya que la relacion es de tipo exclusiva parcial
INSERT INTO producto
VALUES(DEFAULT, 'Auriculares Mickey Mouse', 9.99);

-- Va a fallar porque se ha puesto un atributo (color) que no corresponde a este tipo de producto
INSERT INTO producto
VALUES(DEFAULT, 'Collar', 6.90, 'complemento', NULL, NULL, NULL, 'rojo', 'bisuteria', NULL);
```

A la hora de insertar un producto, si es de tipo prenda, se debe especificar una talla, el género (Hombre o Mujer) y el material. Aunque si no es de ningún tipo de los especificados (relación exclusiva parcial) nos dejará añadir también, como en el caso de *Auriculares Mickey Mouse* que no son de ningún tipo, pero lo hemos podido añadir. Esto se permite para poder añadir artículos que están pendientes de ser clasificados en una sección.

```
psql:script.sql:253: ERROR:  new row for relation "producto" violates check constraint "producto_check"
DETAIL:  Failing row contains (4, Collar, 6.90, complemento, null, null, null, rojo, bisuteria, null).
 id_producto |       nombre_prod        | precio_prod |  tipo   | talla | genero | material | color | categoria | proposito
-------------+--------------------------+-------------+---------+-------+--------+----------+-------+-----------+-----------
           1 | Camisa hombre            |        6.99 | prenda  | M     | H      | algodon  |       |           |
           2 | Tacones rojo             |       12.00 | calzado | 40    | M      |          | rojo  |           |
           3 | Auriculares Mickey Mouse |        9.99 |         |       |        |          |       |           |
(3 rows)
```

TRIGGER

Un trigger es un objeto que se almacena en la base de datos y se asocia a una determinada tabla. Este trigger (o disparador) se ejecutará automáticamente cuando ocurra un evento sobre la tabla a la que está asociada.

En el siguiente ejemplo, el trigger ha sido creado para comprobar que un artículo añadido a las rebajas, previamente, formó parte de alguna campaña.

```sql
CREATE OR REPLACE FUNCTION public.prod_in_campana()
  RETURNS trigger AS
$BODY$
BEGIN
    IF NEW.id_producto NOT IN (SELECT id_producto FROM prod_pertenece_camp) THEN
      RAISE EXCEPTION 'El producto debe pertenecer a una campaña';
    END IF;
    RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql;


CREATE TRIGGER prod_incl
  BEFORE INSERT
  ON prod_esta_rebajas
  FOR EACH ROW
  EXECUTE PROCEDURE public.prod_in_campana();
```

A continuación, insertamos tres artículos en rebajas. Sin embargo, solo deja insertar los dos primeros ya que el tercer artículo no ha pertenecido a ninguna campaña con anterioridad. Con esto resolveremos la inclusividad.

```
INSERT INTO prod_esta_rebajas
VALUES(1, 1);

INSERT INTO prod_esta_rebajas
VALUES(2, 2);

INSERT INTO prod_esta_rebajas
VALUES(2, 3);
```

```
psql:script.sql:368: ERROR:  El producto debe pertenecer a una campaña
CONTEXT:  PL/pgSQL function prod_in_campana() line 4 at RAISE
 id_producto | id_rebajas |  nombre_prod  | fecha_ini  | fecha_fin
-------------+------------+---------------+------------+------------
           1 |          1 | Camisa hombre | 2022-01-01 | 2022-03-01
           2 |          2 | Tacones rojo  | 2022-07-01 | 2022-08-01
(2 rows)
```

NOTA: se trata de algunos ejemplos en los que se emplean las restricciones pedidas.

API REST

Para el desarrollo de la API REST, elegimos hacer uso de Flask el cual nos permite realizar operaciones CRUD sobre la base de datos implementada. Para este caso decidimos implementar la API para clientes ya que la empresa desea mantener una base de datos para los clientes que han realizado una compra y, en casos futuros, ésta será ampliada de manera que pueda ofrecerles ventajas, llevar un registro más exhaustivo para análisis de mercado...

Como podemos observar, se permiten las tareas de visualización de clientes de la empresa, la creación de nuevos clientes, la modificación de clientes ya existentes e incluso el borrado de ellos.

P.F: Clientes Create Delete Update About

Clientes

#1 - Shakira - pique_casio@gmail.com

#2 - Clara - clara-mentegmail.com

#3 - Juan - Juanito2@gmail.com

P.F: Clientes Create Delete Update About

Añadir nuevo cliente

Nombre cliente Nombre cliente

Email Email

Submit

P.F: Clientes Create Delete Update About

Eliminar un cliente

ID id

Delete

P.F: Clientes Create Delete Update About

Actualizar un cliente

Introduzca el ID del cliente a actualizar id

Nuevos valores:

Nombre cliente Nombre

Email Email

Update

About

Realizado por:

Karina Kalwani Israni - alu0101109046

Noelia Ibañez Silvestre - alu0101225555

Repositorio

https://github.com/kk-2503/proyecto-AyDBD.git