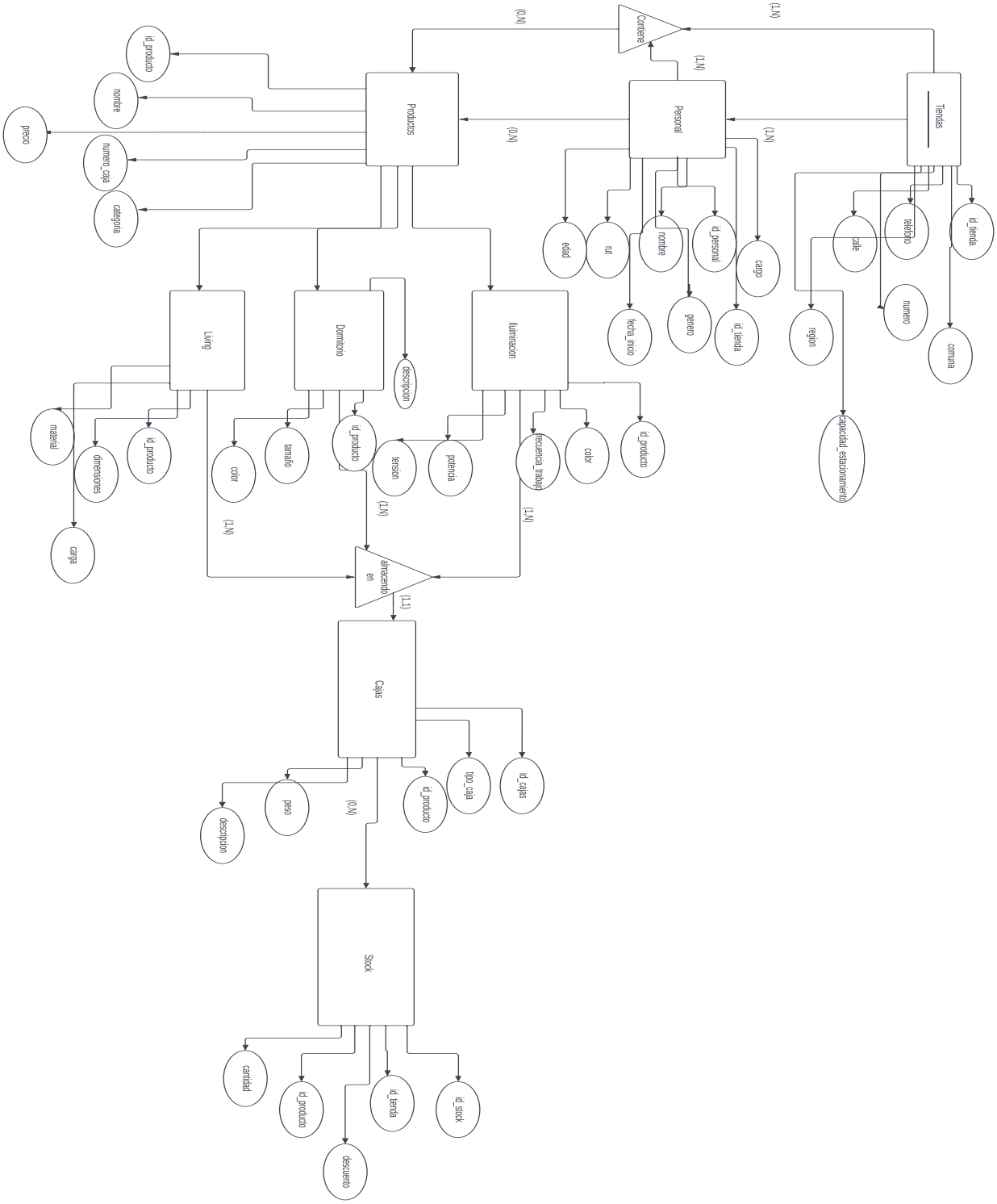
Entrega 2 Base de datos: GRUPO 89

2.1.1



2.1.2

Tabla tiendas

```
CREATE TABLE tiendas (
   id_tienda INT PRIMARY KEY,
   telefono VARCHAR(12),
   calle VARCHAR(50),
   numero INT,
   comuna VARCHAR(50),
   region VARCHAR(50),
   capacidad_estacionamiento INT,
);
```

Tabla personal

```
CREATE TABLE personal (
   id_personal INT PRIMARY KEY,
   nombre VARCHAR(50),
   rut VARCHAR(10),
   edad INT,
   genero INT,
   fecha_inicio DATE,
   cargo VARCHAR(50)
   id_tienda INT,
   FOREIGN KEY (id_tienda) REFERENCES tiendas(id_tienda)
);
```

Tabla productos

```
CREATE TABLE productos (
   id_producto INT PRIMARY KEY,
   nombre VARCHAR(50),
   precio FLOAT,
   numero_caja INT,
   categoria VARCHAR(50),
);
```

Tabla dormitorio

```
CREATE TABLE dormitorio(
   id_producto INT PRIMARY KEY,
   tamano VARCHAR(50),
   color VARCHAR(50),
   descripcion INT,
   FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```

Tabla iluminacion

```
CREATE TABLE iluminacion (
   id_producto INT PRIMARY KEY,
   color VARCHAR(50),
   frecuencia_trabajo INT,
   potencia INT,
   tension INT,
   FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```

Tabla living

```
CREATE TABLE living (
   id_producto INT PRIMARY KEY,
   dimensiones VARCHAR(50),
   material VARCHAR(50),
   carga INT,
   FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```

Tabla cajas

```
CREATE TABLE cajas (
   id_caja INT PRIMARY KEY,
   tipo_caja INT,
   id_producto INT,
   peso FLOAT,
   descripcion INT,
   FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```

Tabla stock

```
CREATE TABLE stock (
   id_stock INT PRIMARY KEY,
   id_tienda INT,
   id_producto INT,
   cantidad INT,
   descuento INT,
   FOREIGN KEY (id_tienda) REFERENCES tiendas(id_tienda),
   FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);
```

Este es el esquema relacional que se deriva del modelo de Entidad/Relación. Se han agregado las restricciones de integridad correspondientes, como las llaves primarias y llaves foráneas, para mantener la coherencia y consistencia de los datos en la base de datos.

2.2.1

Para verificar si el modelo eta en BCNF, necesitamos analizar las dependencias funcionales de cada tabla, entonces:

Tabla tiendas:

- no hay dependencias funcionales adicionales en esta tabla aparte de la llave primaria.

Tabla personal:

- id_personal → nombre, rut, edad, genero, fecha_inicio
- id_tienda → id_personal

Tabla productos:

- id_producto → nombre, precio, numero_caja

Tabla dormitorio:

- id_producto → tamano, color, descripción

Tabla iluminación:

- id_producto → frecuencia_trabajo, color, potencia, tensión

Tabla living:

- id_producto → dimensiones, materia, carga, tensión

Tabla cajas:

- id_producto, id_caja → peso, descripción

Tabla stock:

- id_stock → id_tienda, id_producto, cantidad, descuento

Al analizar estas dependencias funcionales, podemos ver que todas las tablas están en BCNF, ya que todas las dependencias funcionales están completamente determinadas por las llaves primarias de cada tabla. Es decir, no hay dependencias parciales o dependencias transitivas que violen BCNF.

Consultas SQL Implementadas en la APP WEB

Consulta 1.

```
SELECT id_producto, telefono, calle, numero
FROM productos
WHERE categoria = '$categoria'
ORDER BY precio DESC
LIMIT 5;
```

- Donde $categoria corresponde a la selección del usuario

Consulta 2.

```
SELECT id_tienda, teléfono, calle, numero,
FROM tiendas
WHERE comuna
LIKE '%$comuna%';
```

- Donde $comuna corresponde al input de texto del usuario

Consulta 3.

```
SELECT p.id_producto, p.nombre, c.numero_caja, SUM(c.peso)
FROM productos AS p
JOIN cajas AS c  ON p.id_producto = c.id_producto
WHERE p.id_producto = $id
GROUP BY p.id_producto;
```

- Donde $id corresponde al input de texto del usuario

Consulta 4.

```
SELECT t.id_tienda, p.nombre AS nombre_producto, p.precio, s.descuento
FROM tiendas AS t
JOIN Stock AS s ON t.id_tienda = s.id_ienda
JOIN Productos AS p ON s.id_producto = p.id_producto
WHERE p.nombre LIKE '%$nombre%'
AND s.cantidad > 0
AND t.region = $region';
```

- Donde $nombre corresponde al input de texto y $region corresponde a la selección del usuario

Consulta 5.

```
SELECT COUNT(p.id_personal) AS total_personal
FROM personal AS p
JOIN tiendas AS t ON p.id_tienda = t.id_tienda
WHERE t.region = ‘$region’;
```

- Donde $region corresponde a la selección del usuario

Consulta 6.

```
SELECT t.id_tienda, COUNT(CASE WHEN p.genero = 1 THEN 1 END) AS total_femenino,
COUNT(CASE WHEN p.genero = 0 THEN 1 END) AS total_masculino,
AVG(CASE WHEN p.genero = 1 THEN p.edad END) AS edad_promedio_femenino,
AVG(CASE WHEN p.genero = 0 THEN p.edad END) AS edad_promedio_masculino
FROM tiendas AS t
JOIN personal AS p ON t.id_tienda = p.id_tienda
WHERE t.region = '$region'
GROUP BY t.id_tienda;
```

- Donde $region corresponde a la selección del usuario

Consulta 7.

```
SELECT t.id_tienda, t.calle, t.numero, t.comuna,
SUM(p.precio * s.cantidad) AS valor_total_stock
FROM tiendas AS t
JOIN stock AS s ON t.id_tienda = s.id_tienda
JOIN productos AS p ON s.id_producto = p.id_producto
WHERE t.region = '$region'
GROUP BY t.id_tienda, t.calle, t.numero, t.comuna;
```

- Donde $region corresponde a la selección del usuario

Supuestos y Consideraciones:

GRUPO 89

(Se nombran cambios y consideraciones más generales, algunos cambios propios de la filtración de los datos y arreglo de las tablas no se mencionan)

Tablas:

Personal:

- Se cambia el género: “Femenino” = 1 y “Masculino” = 0.
- Se agrega columna “cargo” para especificar el cargo del empleado.
- Se asigna un id_tienda a cada empleado para saber en qué tienda trabaja. Por ejemplo, si el cargo es “Jefe”, el id_tienda corresponde a la tienda de la cual es jefe.

Productos:

- Se crean las tablas iluminacion, dormitorio y living, cada una con los id_producto y sus características.
- Se agrega a la tabla productos el atributo “categoría” para especificar de qué tipo es cada producto.

Iluminacion, dormitorio y living:

- Se crean a partir de los datos que estaban en el archivo productos.csv
- Se cambia el atributo “descripción” por: “Uno de los productos más vendidos” = 1, “Sin descripción” = 0.

Cajas:

- Se cambia el atributo “descripción” por: “Frágil” = 1, “Sin descripción” = 0.
- Se agrega atributo “id_caja” como un id, y se cambia el “id_caja” por “tipo_caja”, para identificar distintos tipos de caja de cada caja en particular. Aquí asumimos que lo que un principio era “id_caja” correspondía a un tipo en específico de caja, ya que se repetían y tenían distinto peso y id_producto.

Stock:

- Agregamos un id_stock y los ordenamos.

Archivos :

Index.php:

- Para tener un diseño de la página más agradable, buscamos en internet y utilizamos una función llamada “myFunction” para lograr el efecto que se aprecia al bajar por la página.

Styles:

- Utilizamos una imagen de internet a la que nombramos “fondo.png” para utilizar de fondo de la página.
- En el archivo mystyles.css utilizamos el código de las ayudantías y lo modificamos para incluir el fondo, cambiar los colores y el tipo de letra.

Datos_filtrados:

- En este archivo se encuentran los archivos “filtrar_datos.py”, “arreglar_personal_productos.py” y “sacar_regiones.py” además de los archivos csv, que fueron utilizados para filtrar los datos y crear los archivos csv nuevos, corregir errores y listar las regiones para utilizar en el archivo index.php.