

## Actividad 2 - Módulo 4

**Nombre:** Carlos Saldivia Susperreguy

### 1. Esquema de la base de datos

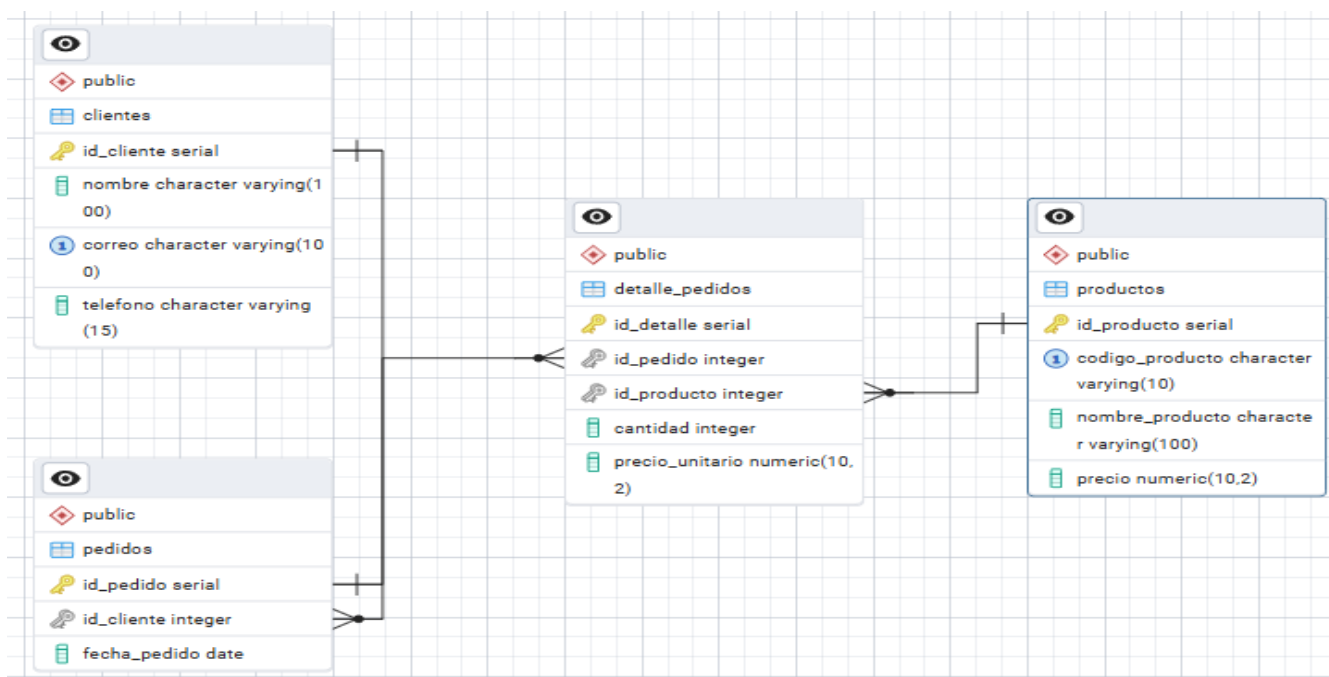
```
-- 1. ESQUEMA DE LA BASE DE DATOS
-- Cuatro tablas relacionadas con claves primarias y foráneas

CREATE TABLE clientes (
  id_cliente SERIAL PRIMARY KEY,
  nombre VARCHAR(100),
  correo VARCHAR(100) UNIQUE,
  telefono VARCHAR(15)
);

CREATE TABLE productos (
  id_producto SERIAL PRIMARY KEY,
  codigo_producto VARCHAR(10) UNIQUE NOT NULL,
  nombre_producto VARCHAR(100),
  precio DECIMAL(10, 2)
);

CREATE TABLE pedidos (
  id_pedido SERIAL PRIMARY KEY,
  id_cliente INT REFERENCES clientes(id_cliente),
  fecha_pedido DATE DEFAULT CURRENT_DATE
);

CREATE TABLE detalle_pedidos (
  id_detalle SERIAL PRIMARY KEY,
  id_pedido INT REFERENCES pedidos(id_pedido),
  id_producto INT REFERENCES productos(id_producto),
  cantidad INT CHECK (cantidad > 0),
  precio_unitario DECIMAL(10, 2),
  UNIQUE(id_pedido, id_producto)
);
```



## 2. Inserción de datos

```
-- 2. INSERCIÓN DE DATOS

-- Insertar 3 clientes
INSERT INTO clientes (nombre, correo, telefono) VALUES
('Carlos Saldivia', 'carlos.saldivia@kibernum.cl', '+56912345678'),
('Eduardo Villegas', 'eduardo.villegas@kibernum.cl', '+56987654321'),
('Mario Inostroza', 'mario.inostroza@kibernum.cl', '+56955555555');

-- Insertar 3 productos (con códigos únicos)
INSERT INTO productos (codigo_producto, nombre_producto, precio) VALUES
('ELEC001', 'Smartphone POCO F7', 599000.00),
('ELEC002', 'Audífonos Bluetooth', 89000.00),
('ROPA001', 'Polera Algodón Básica', 25000.00);

-- Insertar 2 pedidos
INSERT INTO pedidos (id_cliente, fecha_pedido) VALUES
(1, '2025-07-15'),
(2, '2025-07-16');

-- Insertar detalles de pedidos
INSERT INTO detalle_pedidos (id_pedido, id_producto, cantidad, precio_unitario) VALUES
-- Pedido 1: Smartphone + Polera
(1, 1, 1, 599000.00),
(1, 3, 2, 25000.00),
-- Pedido 2: Audífonos + Polera
(2, 2, 1, 89000.00),
(2, 3, 1, 25000.00);
```

### 3. Creación de índices

```
-- 3. CREACIÓN DE ÍNDICES

-- Índice 1: En correo de clientes (búsquedas frecuentes de login)
CREATE INDEX idx_clientes_correo ON clientes(correo);

-- Índice 2: En código de producto (búsquedas por código único)
CREATE INDEX idx_productos_codigo ON productos(codigo_producto);
```

Justificación:

- idx\_clientes\_correo: Mejora búsquedas de clientes por email en procesos de login/authenticación
- idx\_productos\_codigo: Acelera búsquedas de productos por código en gestión de inventario

#### 4. Creación de vistas

```
-- 4. CREACIÓN DE VISTA

CREATE VIEW vista_pedidos_completa AS
SELECT
    p.id_pedido,
    p.fecha_pedido,
    c.nombre AS cliente_nombre,
    c.correo,
    pr.codigo_producto,
    pr.nombre_producto,
    dp.cantidad,
    dp.precio_unitario,
    (dp.cantidad * dp.precio_unitario) AS subtotal_producto
FROM pedidos p
JOIN clientes c ON p.id_cliente = c.id_cliente
JOIN detalle_pedidos dp ON p.id_pedido = dp.id_pedido
JOIN productos pr ON dp.id_producto = pr.id_producto
ORDER BY p.fecha_pedido DESC;
```

Query

Query History

1

SELECT \* FROM public.vista\_pedidos\_completa

Data Output

Messages

Notifications

Showing rows: 1 to 4

Page No: 1

of 1


	id_pedido integer	fecha_pedido date	cliente_nombre character varying (100)	correo character varying (100)	codigo_producto character varying (10)	nombre_producto character varying (100)	cantidad integer	precio_unitario numeric (10,2)	subtotal numeric
1	2	2025-07-16	Eduardo Villegas	eduardo.villegas@kibernum.cl	ELEC002	Audífonos Bluetooth	1	89000.00	
2	2	2025-07-16	Eduardo Villegas	eduardo.villegas@kibernum.cl	ROPA001	Polera Algodón Básica	1	25000.00	
3	1	2025-07-15	Carlos Saldivia	carlos.saldivia@kibernum.cl	ELEC001	Smartphone POCO F7	1	599000.00	
4	1	2025-07-15	Carlos Saldivia	carlos.saldivia@kibernum.cl	ROPA001	Polera Algodón Básica	2	25000.00	

## 5. Optimización y análisis de consulta


```
-- Consulta relevante para área de ventas: Total de ventas por cliente
EXPLAIN ANALYZE
SELECT
    cliente_nombre,
    correo,
    COUNT(*) AS productos_comprados,
    SUM(cantidad) AS total_unidades,
    SUM(subtotal_producto) AS total_gastado
FROM vista_pedidos_completa
GROUP BY cliente_nombre, correo
ORDER BY total_gastado DESC;
```

	QUERY PLAN	
	text	
1	Sort (cost=214.24..214.74 rows=200 width=484) (actual time=0.116..0.118 rows=2 loops=1)	
2	Sort Key: (sum((((dp.cantidad)::numeric * dp.precio_unitario)))) DESC	
3	Sort Method: quicksort Memory: 25kB	
4	-> HashAggregate (cost=204.10..206.60 rows=200 width=484) (actual time=0.108..0.110 rows=2 loops=1)	
5	Group Key: c.nombre, c.correo	
6	Batches: 1 Memory Usage: 40kB	
7	-> Sort (cost=170.10..173.50 rows=1360 width=752) (actual time=0.097..0.099 rows=4 loops=1)	
8	Sort Key: p.fecha_pedido DESC	
9	Sort Method: quicksort Memory: 25kB	
10	-> Hash Join (cost=58.04..99.31 rows=1360 width=752) (actual time=0.081..0.087 rows=4 loops=1)	
11	Hash Cond: (dp.id_producto = pr.id_producto)	
12	-> Hash Join (cost=56.97..87.80 rows=1360 width=464) (actual time=0.042..0.046 rows=4 loops=1)	
13	Hash Cond: (p.id_cliente = c.id_cliente)	
14	-> Hash Join (cost=55.90..83.08 rows=1360 width=32) (actual time=0.023..0.026 rows=4 loops=1)	
15	Hash Cond: (dp.id_pedido = p.id_pedido)	
16	-> Seq Scan on detalle_pedidos dp (cost=0.00..23.60 rows=1360 width=28) (actual time=0.006..0.007 rows=4 loops=1)	
17	-> Hash (cost=30.40..30.40 rows=2040 width=12) (actual time=0.009..0.010 rows=2 loops=1)	
18	Buckets: 2048 Batches: 1 Memory Usage: 17kB	
19	-> Seq Scan on pedidos p (cost=0.00..30.40 rows=2040 width=12) (actual time=0.007..0.007 rows=2 loops=1)	
20	-> Hash (cost=1.03..1.03 rows=3 width=440) (actual time=0.011..0.012 rows=3 loops=1)	
21	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
22	-> Seq Scan on clientes c (cost=0.00..1.03 rows=3 width=440) (actual time=0.008..0.008 rows=3 loops=1)	
23	-> Hash (cost=1.03..1.03 rows=3 width=4) (actual time=0.029..0.030 rows=3 loops=1)	
24	Buckets: 1024 Batches: 1 Memory Usage: 9kB	
25	-> Seq Scan on productos pr (cost=0.00..1.03 rows=3 width=4) (actual time=0.024..0.025 rows=3 loops=1)	
26	Planning Time: 0.387 ms	
27	Execution Time: 0.231 ms	

```
-- Consulta adicional para verificar uso de índice por correo
EXPLAIN ANALYZE
SELECT * FROM clientes WHERE correo = 'carlos.saldivia@kibernum.cl';
```

	QUERY PLAN	
	text	
1	Seq Scan on clientes (cost=0.00..1.04 rows=1 width=488) (actual time=0.015..0.016 rows=1 loops=...	
2	Filter: ((correo)::text = 'carlos.saldivia@kibernum.cl')::text)	
3	Rows Removed by Filter: 2	
4	Planning Time: 0.095 ms	
5	Execution Time: 0.028 ms	

```
-- Consulta adicional para verificar uso de índice por código de producto
EXPLAIN ANALYZE
SELECT * FROM productos WHERE codigo_producto = 'ELEC001';
```

	QUERY PLAN	
	text	
1	Seq Scan on productos (cost=0.00..1.04 rows=1 width=276) (actual time=0.013..0.014 rows=1 loops=...	
2	Filter: ((codigo_producto)::text = 'ELEC001')::text)	
3	Rows Removed by Filter: 2	
4	Planning Time: 0.096 ms	
5	Execution Time: 0.025 ms	

## Justificación

Con solo 3 registros por tabla, es normal que PostgreSQL use Seq Scan. Los índices son más útiles con miles o millones de registros.