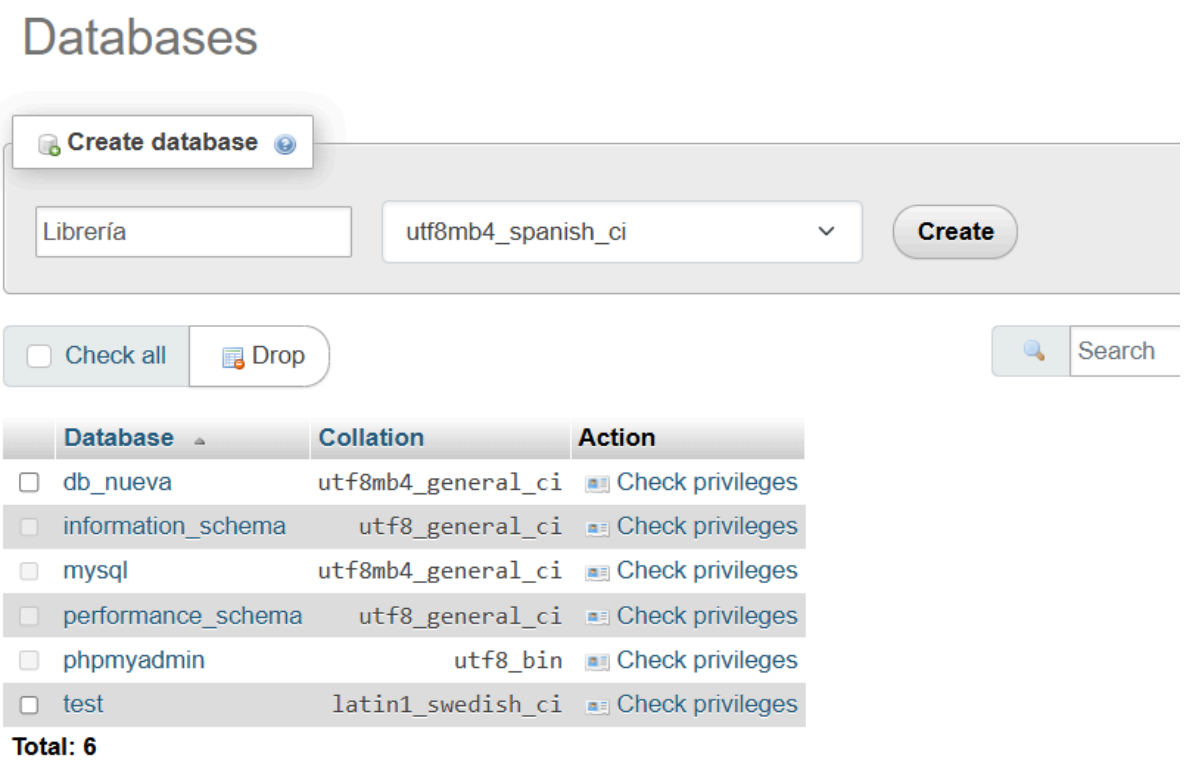
# Trabajo Práctico 6

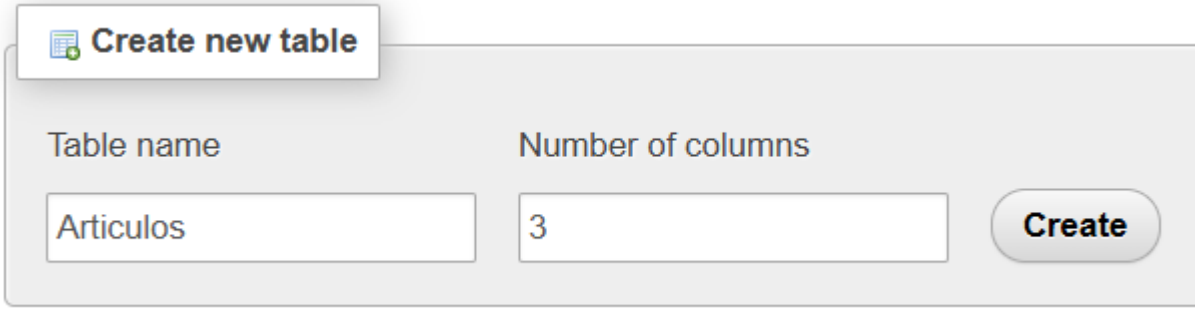
## Sistemas de gestión de bases de datos


- **Alumno**: Sussini Guanziroli, Patricio
- **Materia**: Bases de datos I
- **Tutora**: Constanza Uño

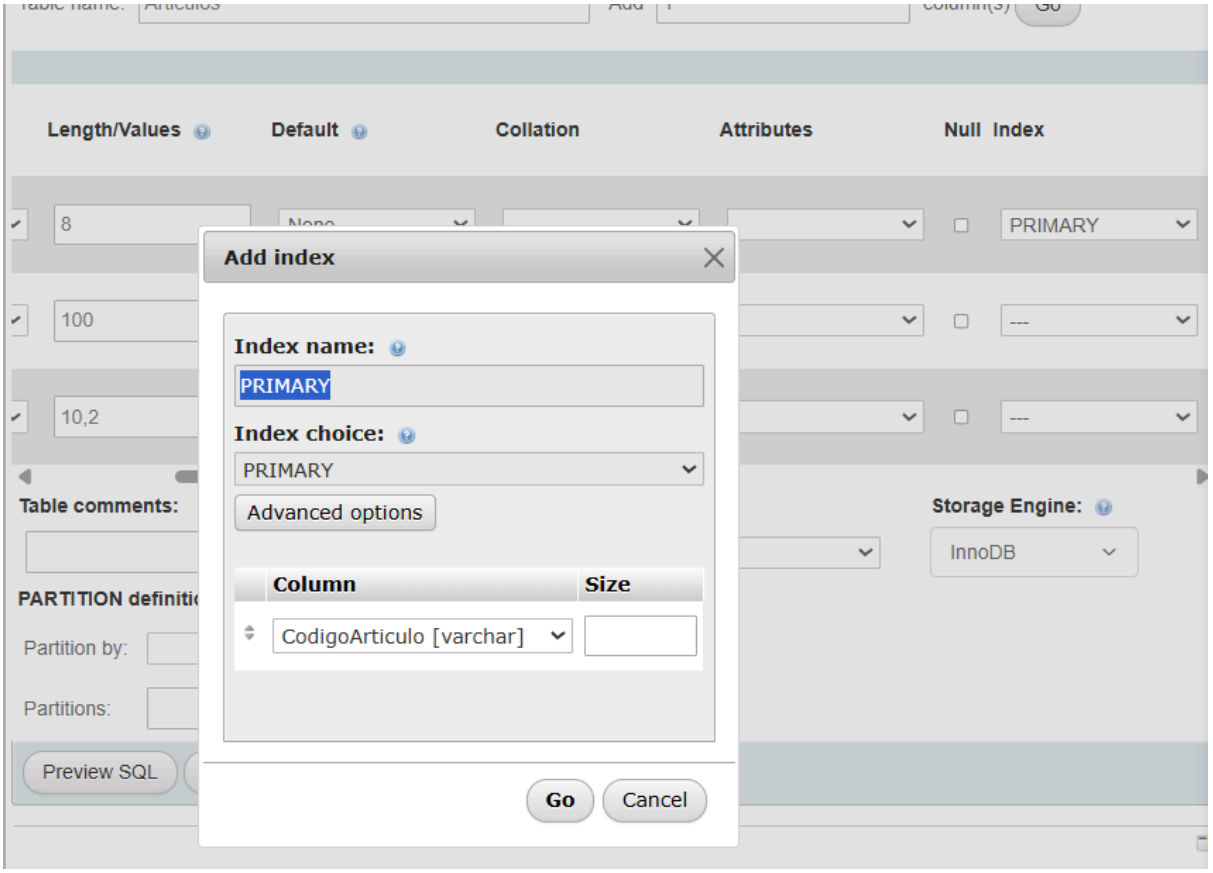
# A. Creación de la Database



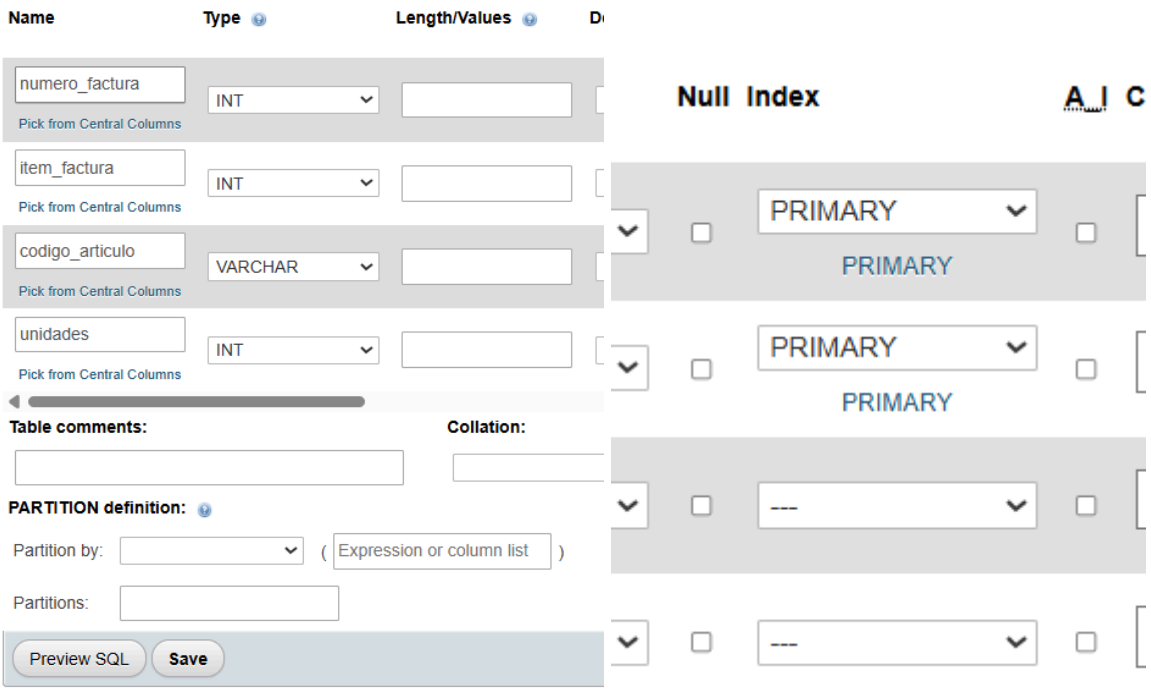

Creo la nueva tabla:



Configuro CodigoArticulo como PK.

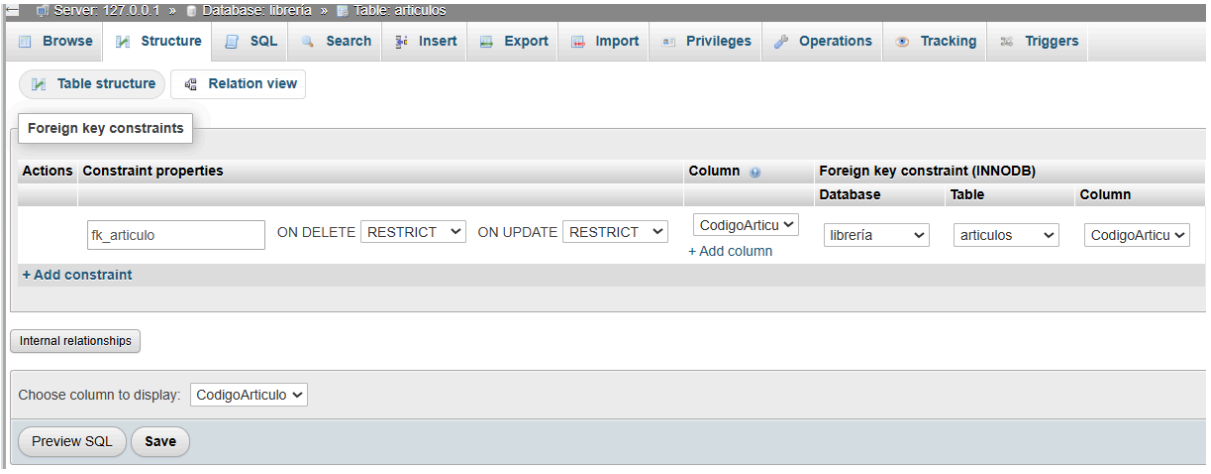



Guardamos la tabla de detalle_factura.

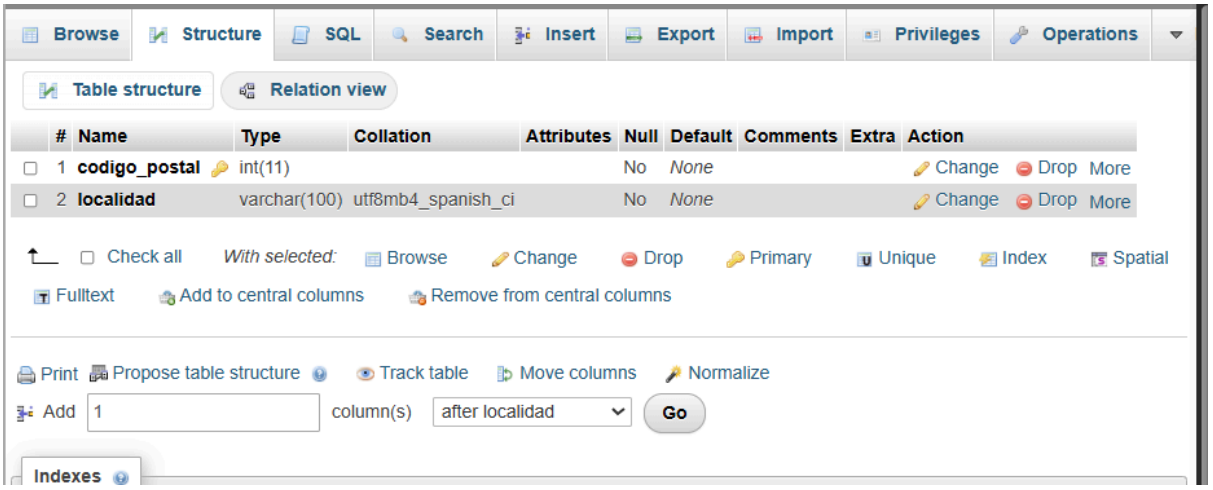



Con las keys configuradas.

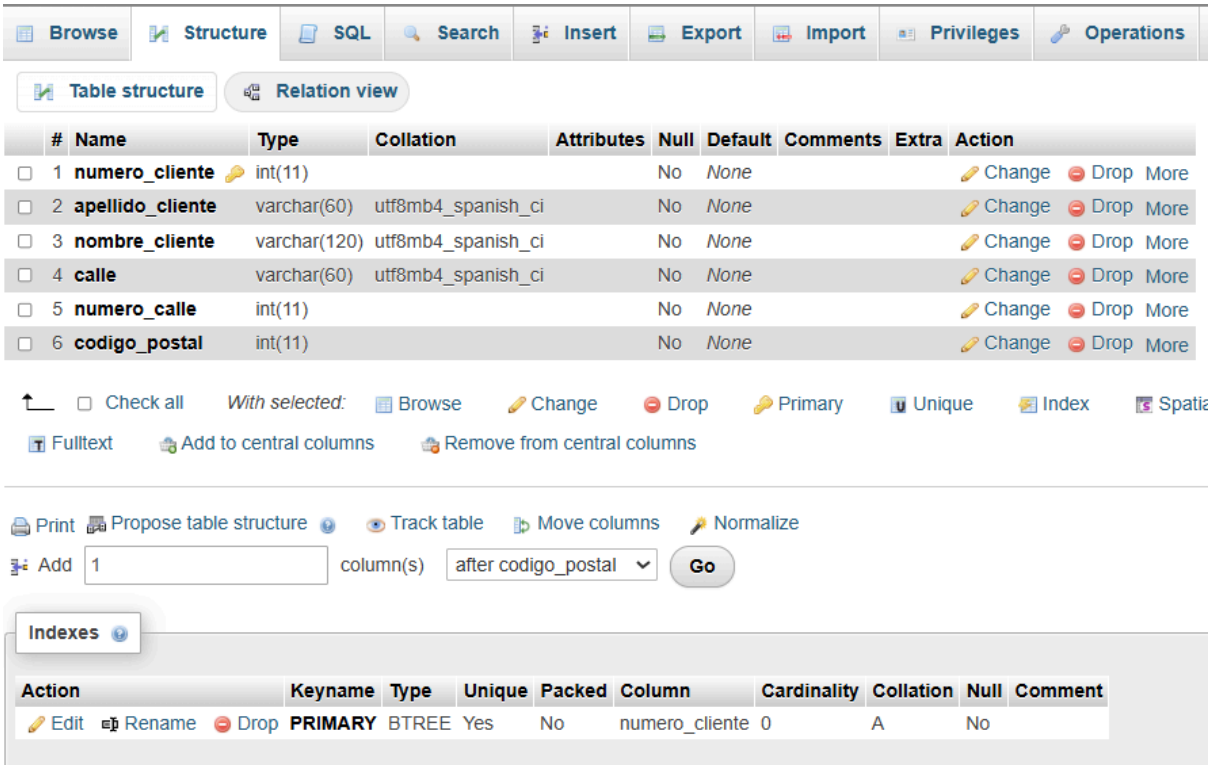
Agregamos la restricción.




Creamos tabla de localidades.

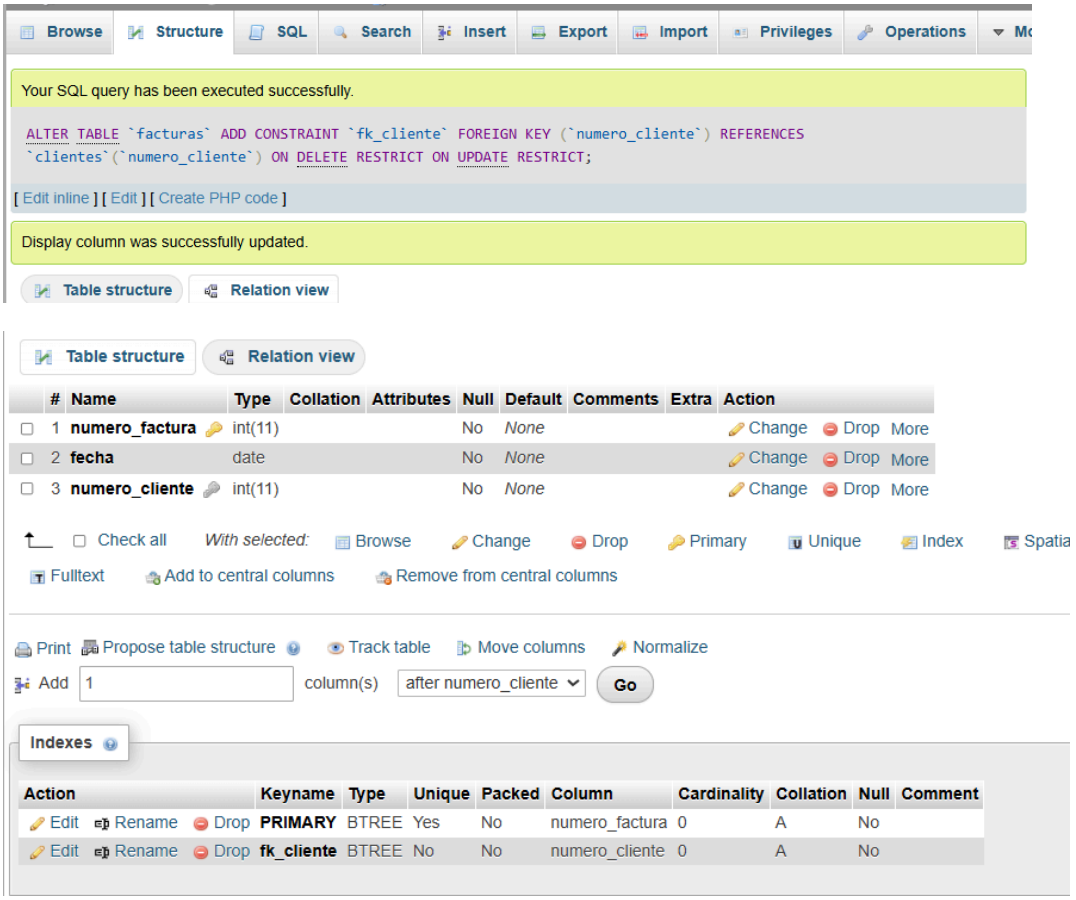


Creamos tabla de clientes.

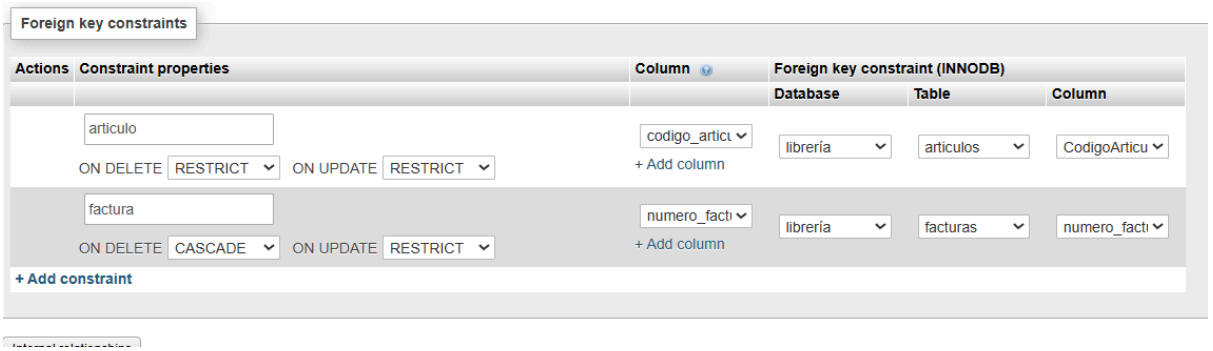



Con los pk y fk

Luego creamos la tabla facturas.



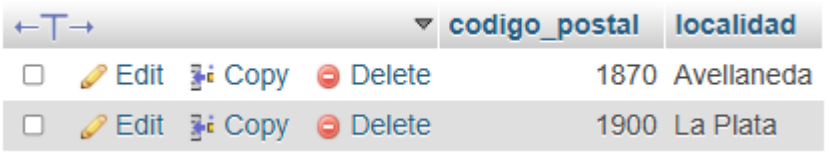
Con los pk y fk




Agregamos por último las relaciones.

**B. Carga de datos.**

Vamos a cargar los datos usando INSERT INTO como sentencia de SQL sobre la base de datos “librería”.


Quedan cargadas las tablas.

Sentencia SQL de carga completa:

**– Cargamos Localidades**

INSERT INTO localidades (codigo_postal, localidad) VALUES

(1870, 'Avellaneda'),

(1900, 'La Plata');

**-- Cargamos artículos**

INSERT INTO articulos (CodigoArticulo, Articulo, Precio) VALUES

('1-0023-D', 'Auriculares inalámbricos', 175),

('2-0023-D', 'Teclado inalámbrico', 245),

('3-0023-D', 'Mini Adaptador USB-C USB 2.0', 75),

('4-0023-D', 'Cable red UTP RJ45', 90),

('5-0023-D', 'Cámara web USB', 165);

**-- 2. Cargamos clientes**

INSERT INTO clientes (numero_cliente, apellido_cliente, nombre_cliente, calle, numero_calle, codigo_postal) VALUES

(21, 'Rodriguez', 'Facundo', 'Calle Rivero', 41, 1870),

(22, 'Martinez', 'Natalia', 'Calle San Eloy', 82, 1870),

(23, 'Garcia', 'Carlos', 'Avenida de Italia', 245, 1900);

**-- 3. Cargamos facturas**

INSERT INTO facturas (numero_factura, fecha, numero_cliente) VALUES

(335, '2024-08-01', 21),

(336, '2024-08-02', 22),

(337, '2024-08-02', 23);

**-- 4. Cargamos detalle_factura**

INSERT INTO detalle_factura (numero_factura, item_factura, codigo_articulo, unidades) VALUES

(335, 1, '2-0023-D', 10),

(335, 2, '4-0023-D', 12),

(335, 3, '5-0023-D', 1),

(336, 1, '1-0023-D', 2),

(336, 2, '3-0023-D', 2),

(337, 1, '2-0023-D', 25);

**C. Visualizar la creación de tablas y el armado de índices.**

Mediante la sentencia SHOW CREATE TABLE {nombre_tabla}

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE localidades;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| localidades | CREATE TABLE `localidades` ( `codigo_postal` int... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE clientes;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| clientes | CREATE TABLE `clientes` ( `numero_cliente` int(1... |

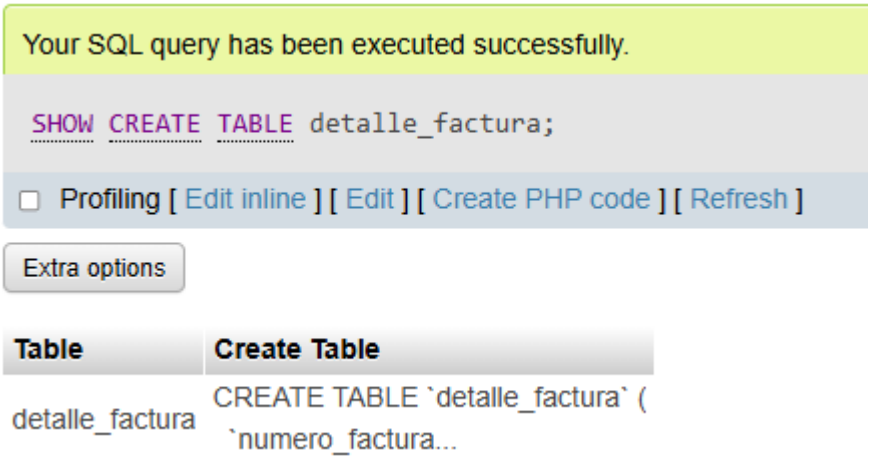
Your SQL query has been executed successfully.

```
SHOW CREATE TABLE facturas;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| facturas | CREATE TABLE `facturas` ( `numero_factura` int(1... |




Bueno, ahi se ven creadas las tablas.

**D. Sentencia para consultar sobre qué clientes realizaron compras el 02/08/2024**


Usando la sentencia podemos saber que Natalia y Carlos realizaron compras ese día.

# Actividad 2:

## a. Análisis de la db.-

- Las tablas: **clientes** y **vuelos**, son centrales, y un vuelo también puede tener múltiples reservas. La tabla **reservas** es una tabla intermedia que conecta clientes específicos con un vuelo específico.
- Las tablas de “soporte” que hacen de catálogo.
    - **Avión:** Contiene la info de los aviones, un avión puede ser usado en muchos vuelos.
    - **Horarios:** Define las horas de vuelo. Un horario puede coincidir con muchos destinos.
    - **Destinos:** Define los lugares a donde se puede volar y está relacionado con los horarios. Un mismo destino puede tener múltiples vuelos.
- Las **relaciones clave:**
    - **Reservas** depende de **Clientes** y **Vuelos.**
    - **Vuelos** depende de **Avión y Destinos.**
    - **Destinos** depende de **Horarios.**

## b. Creación de la db en phpMyAdmin.

Creo la base de datos **Vuelos** usando sentencias de SQL con la información de las tablas correspondientes.

```
CREATE DATABASE Vuelos;
USE Vuelos;

CREATE TABLE Clientes (
        codigo INT PRIMARY KEY,
        nombre VARCHAR(50),
        apellido VARCHAR(50),
        fecha_nacimiento DATE,
        tipo VARCHAR(20),
        usuario VARCHAR(20)
);

CREATE TABLE Avion (
        codigo INT PRIMARY KEY,
        marca VARCHAR(50),
        estado VARCHAR(20)
);

CREATE TABLE Horarios (
        id_horario INT PRIMARY KEY,
        hora_vuelo VARCHAR(20)
);

CREATE TABLE Destinos (
        id_destino INT PRIMARY KEY,
         id_horario INT,
        nombre VARCHAR(50),
        FOREIGN KEY (id_horario) REFERENCES Horarios(id_horario)
);
```

```
CREATE TABLE Vuelos (
        id_vuelo INT PRIMARY KEY,
        disponibilidad VARCHAR(20),
        tipo_vuelo VARCHAR(20),
        codigo_avion INT,
        id_destino INT,
        FOREIGN KEY (codigo_avion) REFERENCES Avion(codigo),
        FOREIGN KEY (id_destino) REFERENCES Destinos(id_destino)
 );

CREATE TABLE Reservas (
        id_reserva INT PRIMARY KEY,
        estado VARCHAR(20),
        id_vuelo INT,
        codigo_cliente INT,
        FOREIGN KEY (id_vuelo) REFERENCES Vuelos(id_vuelo),
        FOREIGN KEY (codigo_cliente) REFERENCES Clientes(codigo)
 );
```

Solo creamos las tablas y todavía no se cargarán datos en ellas.

| | Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|---|
| ☐ | avion | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | clientes | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | destinos | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 32.0 KiB | - |
| ☐ | horarios | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 16.0 KiB | - |
| ☐ | reservas | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 48.0 KiB | - |
| ☐ | vuelos | Browse Structure Search Insert Empty Drop | 0 | InnoDB | utf8mb4_general_ci | 48.0 KiB | - |
| | **6 tables** | **Sum** | **0** | **InnoDB** | **utf8mb4_general_ci** | **176.0 KiB** | **0 B** |

## c. Carga de datos en la DB respetando el orden establecido por sus dependencias.

El orden necesario es:

1. **Avion, Horarios y Clientes**, sin claves foráneas.
2. **Destinos:** Esta tabla necesita que los horarios existan.
3. **Vuelos**: Necesita que los aviones y destinos existan.
4. **Reservas:** Depende de absolutamente todo.

## d. Sentencias SQL para insertar los datos realizando DML.

Usando la sentencia INSERT, ingresamos las tablas con 5 pasajeros y 8 reservas. Siendo el último pasajero VIP totalmente, Patricio Sussini.

```sql
-- 1. Insertamos en las tablas sin dependencias
INSERT INTO Avion (codigo, marca, estado) VALUES
(101, 'Boeing 737', 'Activo'),
(102, 'Airbus A320', 'Mantenimiento');

INSERT INTO Horarios (id_horario, hora_vuelo) VALUES
(1, '08:00 AM'),
(2, '02:30 PM'),
(3, '09:15 PM');

INSERT INTO Clientes (codigo, nombre, apellido, fecha_nacimiento, tipo, usuario) VALUES
(1, 'Juan', 'Perez', '1990-05-15', 'Frecuente', 'jperez'),
(2, 'Maria', 'Gomez', '1985-11-20', 'Ocasional', 'mgomez'),
(3, 'Carlos', 'Lopez', '2001-02-10', 'Frecuente', 'clopez'),
(4, 'Ana', 'Martinez', '1998-07-30', 'Nuevo', 'amartinez'),
(5, 'Patricio', 'Sussini', '2002-01-25', 'VIP-UTN', 'drsetup');

-- 2. Insertamos en Destinos
INSERT INTO Destinos (id_destino, id_horario, nombre) VALUES
(1, 1, 'Madrid'),
(2, 2, 'Miami'),
(3, 3, 'Tokio');

-- 3. Insertamos en Vuelos
INSERT INTO Vuelos (id_vuelo, disponibilidad, tipo_vuelo, codigo_avion, id_destino) VALUES
(501, 'Disponible', 'Internacional', 101, 1),
(502, 'Pocos asientos', 'Internacional', 102, 2),
(503, 'Disponible', 'Internacional', 101, 3);

-- 4. Insertamos las 8 Reservas
INSERT INTO Reservas (id_reserva, estado, id_vuelo, codigo_cliente) VALUES
(1001, 'Confirmada', 501, 1),
(1002, 'Confirmada', 501, 2),
(1003, 'Pendiente', 502, 3),
(1004, 'Confirmada', 502, 4),
```

```
(1005, 'Cancelada', 503, 1),
(1006, 'Confirmada', 503, 2),
(1007, 'Confirmada', 501, 5),
(1008, 'Pendiente', 502, 5);
```

## e. Visualizamos las queries de CREATE.

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE Clientes;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| Clientes | CREATE TABLE `clientes` ( `codigo` int(11) NOT N... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE Vuelos;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| Vuelos | CREATE TABLE `vuelos` ( `id_vuelo` int(11) NOT N... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE avion;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| avion | CREATE TABLE `avion` ( `codigo` int(11) NOT NULL... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE destinos;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| destinos | CREATE TABLE `destinos` ( `id_destino` int(11) N... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE reservas;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| reservas | CREATE TABLE `reservas` ( `id_reserva` int(11) N... |

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE horarios;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Create Table |
|---|---|
| horarios | CREATE TABLE `horarios` ( `id_horario` int(11) N... |

## f. Visualizar planes de ejecución con sentencias. (JOIN + EXPLAIN)

Your SQL query has been executed successfully.

```
EXPLAIN SELECT v.id_vuelo, c.nombre, c.apellido FROM Reservas r JOIN Clientes c ON r.codigo_cliente = c.codigo JOIN Vuelos v ON r.id_vuelo
= v.id_vuelo JOIN Destinos d ON v.id_destino = d.id_destino WHERE d.nombre = 'Madrid';
```

[ Edit inline ] [ Edit ] [ Skip Explain SQL] [ Create PHP code ]

Extra options

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | d | ALL | PRIMARY | *NULL* | *NULL* | *NULL* | 3 | Using where |
| 1 | SIMPLE | v | ref | PRIMARY,id_destino | id_destino | 5 | vuelos.d.id_destino | 1 | Using index |
| 1 | SIMPLE | r | ALL | id_vuelo,codigo_cliente | *NULL* | *NULL* | *NULL* | 8 | Using where; Using join buffer (flat, BNL join) |
| 1 | SIMPLE | c | eq_ref | PRIMARY | PRIMARY | 4 | vuelos.r.codigo_cliente | 1 | |

Se visualiza el plan de SQL para la ejecución.

## g. Análisis y Optimización con índice.

- Primero ejecutamos el plan sin índice.

Your SQL query has been executed successfully.

```
EXPLAIN SELECT * FROM Clientes ORDER BY apellido, nombre;
```

[ Edit inline ] [ Edit ] [ Skip Explain SQL] [ Create PHP code ]

Extra options

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | Clientes | ALL | *NULL* | *NULL* | *NULL* | *NULL* | 5 | Using filesort |

Como usa filesort, la db tuvo que recorrer la totalidad de las entradas para ordenarlas, lo cual es lento.

- Ahora agregamos el índice.

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0009 seconds.)

```
CREATE INDEX idx_apellido_nombre ON Clientes (apellido, nombre);
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Your SQL query has been executed successfully.

```
EXPLAIN SELECT * FROM Clientes ORDER BY apellido, nombre;
```

[ Edit inline ] [ Edit ] [ Skip Explain SQL] [ Create PHP code ]

Extra options

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | Clientes | ALL | *NULL* | *NULL* | *NULL* | *NULL* | 5 | Using filesort |

Sigue usando filesort porque es muy pequeña la tabla, pero está funcionando y creado el índice.

Your SQL query has been executed successfully.

```
SHOW INDEX FROM Clientes;
```

Profiling [ Edit inline ] [ Edit ] [ Create PHP code ] [ Refresh ]

Extra options

| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
|---|---|---|---|---|---|---|---|---|---|---|
| clientes | 0 | PRIMARY | 1 | codigo | A | 5 | NULL | NULL | | BTREE |
| clientes | 1 | idx_apellido_nombre | 1 | apellido | A | 5 | NULL | NULL | YES | BTREE |
| clientes | 1 | idx_apellido_nombre | 2 | nombre | A | 5 | NULL | NULL | YES | BTREE |

## h. Generar y hacer análisis de una Vista.

- Creamos primero la vista usando una sentencia de SQL CREATE VIEW.

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```
CREATE VIEW vista_clientes_ordenados AS SELECT * FROM Clientes ORDER BY apellido, nombre;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

- Accedemos a la vista con otro orden.

Your SQL query has been executed successfully.

```
EXPLAIN SELECT * FROM vista_clientes_ordenados ORDER BY nombre;
```

[ Edit inline ] [ Edit ] [ Skip Explain SQL] [ Create PHP code ]

Extra options

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | clientes | ALL | NULL | NULL | NULL | NULL | 5 | Using filesort |

**Conclusión:**

La vista tiene un índice compuesto por apellido, nombre y le pedimos que los ordene solo por nombre, por lo tanto el índice falla y la consulta se hace manualmente. La efectividad de un índice compuesto depende directamente de que se utilicen sus columnas en el orden en que fueron definidas. Al pedir un orden que salta la primera columna del índice y pasa directamente a la segunda la base de datos no puede sacar provecho de la estructura pre-ordenada del índice para la operación y debe recurrir si o si al filesort.

Esto demuestra que los índices deben ser estrictamente diseñados de forma específica para las consultas que se busca optimizar.