

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

Databases

Create database

Librería

utf8mb4_spanish_ci

Create

☐ Check all

Drop

Search

	Database	Collation	Action
<input type="checkbox"/>	db_nueva	utf8mb4_general_ci	Check privileges
<input type="checkbox"/>	information_schema	utf8_general_ci	Check privileges
<input type="checkbox"/>	mysql	utf8mb4_general_ci	Check privileges
<input type="checkbox"/>	performance_schema	utf8_general_ci	Check privileges
<input type="checkbox"/>	phpmyadmin	utf8_bin	Check privileges
<input type="checkbox"/>	test	latin1_swedish_ci	Check privileges

Total: 6

Creo la nueva tabla:

Create new table

Table name

Number of columns

Articulos

3

Create

Configuro `CodigoArticulo` como PK.

Guardamos la tabla de detalle_factura.

Name	Type	Length/Values	Default	Null	Index
numero_factura	INT				
Pick from Central Columns					
item_factura	INT				
Pick from Central Columns					
codigo_articulo	VARCHAR				
Pick from Central Columns					
unidades	INT				
Pick from Central Columns					

Table comments: Collation:

PARTITION definition:

Partition by: (Expression or column list)

Partitions:

Preview SQL Save

Con las keys configuradas.

Agregamos la restricción.

Server: 127.0.0.1 » Database: librería » Table: articulos

Table structure Relation view

Foreign key constraints

Actions	Constraint properties	Column	Foreign key constraint (INNODB)		
			Database	Table	Column
	fk_articulo ON DELETE RESTRICT ON UPDATE RESTRICT	CodigoArticulo	librería	articulos	CodigoArticulo
+ Add constraint					

Internal relationships

Choose column to display: CodigoArticulo

Preview SQL Save

Creamos tabla de localidades.

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	codigo_postal	int(11)		No	None			Change Drop More
<input type="checkbox"/>	2	localidad	varchar(100) utf8mb4_spanish_ci		No	None			Change Drop More

☐ Check all With selected: [Browse](#) [Change](#) [Drop](#) [Primary](#) [Unique](#) [Index](#) [Spatial](#)

[Fulltext](#) [Add to central columns](#) [Remove from central columns](#)

[Print](#) [Propose table structure](#) [Track table](#) [Move columns](#) [Normalize](#)

[Add](#) 1 column(s) after localidad [Go](#)

Indexes

Creamos tabla de clientes.

Database management interface showing the structure of the 'clientes' table.

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	numero_cliente	int(11)			No	None			Change Drop More
2	apellido_cliente	varchar(60)	utf8mb4_spanish_ci		No	None			Change Drop More
3	nombre_cliente	varchar(120)	utf8mb4_spanish_ci		No	None			Change Drop More
4	calle	varchar(60)	utf8mb4_spanish_ci		No	None			Change Drop More
5	numero_calle	int(11)			No	None			Change Drop More
6	codigo_postal	int(11)			No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	numero_cliente	0	A	No	

Con los pk y fk

Luego creamos la tabla facturas.

Database management interface showing the structure of the 'facturas' table and the execution of a SQL query to add a foreign key constraint.

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	numero_factura	int(11)			No	None			Change Drop More
2	fecha	date			No	None			Change Drop More
3	numero_cliente	int(11)			No	None			Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	numero_factura	0	A	No	
Edit Rename Drop	fk_cliente	BTREE	No	No	numero_cliente	0	A	No	

SQL Query:

```
ALTER TABLE `facturas` ADD CONSTRAINT `fk_cliente` FOREIGN KEY (`numero_cliente`) REFERENCES `clientes` (`numero_cliente`) ON DELETE RESTRICT ON UPDATE RESTRICT;
```

Con los pk y fk

Foreign key constraints

Actions	Constraint properties	Column	Foreign key constraint (InnoDB)		
			Database	Table	Column
	<input type="text" value="articulo"/>	<input type="text" value="codigo_articul"/>	<input type="text" value="librería"/>	<input type="text" value="articulos"/>	<input type="text" value="CodigoArticu"/>
ON DELETE	<input type="text" value="RESTRICT"/>	ON UPDATE			
	<input type="text" value="factura"/>	<input type="text" value="numero_facti"/>	<input type="text" value="librería"/>	<input type="text" value="facturas"/>	<input type="text" value="numero_facti"/>
ON DELETE	<input type="text" value="CASCADE"/>	ON UPDATE			
+ Add constraint					

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”.

```
✓ 5 rows inserted. (Query took 0.0024 seconds.)

-- Corregido para coincidir con tus columnas: CodigoArticulo, Articulo, Precio 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);

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

✓ 3 rows inserted. (Query took 0.0026 seconds.)

-- 2. Cargamos clientes (tus nombres de columna usan guion bajo) 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);

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

✓ 3 rows inserted. (Query took 0.0028 seconds.)

-- 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);

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

✓ 6 rows inserted. (Query took 0.0030 seconds.)

-- 4. Cargamos detalle_factura (el nombre de tu tabla y columnas usan guion bajo) 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);

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

<div>←T→</div>					codigo_postal	localidad		
<input type="checkbox"/>		Edit		Copy		Delete	1870	Avellaneda
<input type="checkbox"/>		Edit		Copy		Delete	1900	La Plata

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...

Your SQL query has been executed successfully.

```
SHOW CREATE TABLE detalle_factura;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

Table	Create Table
detalle_factura	CREATE TABLE `detalle_factura` (`numero_factura`...

Bueno, ahi se ven creadas las tablas.

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

Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT c.apellido_cliente, c.nombre_cliente FROM clientes c JOIN facturas f ON c.numero_cliente = f.numero_cliente WHERE f.fecha = '2024-08-02';
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

apellido_cliente	nombre_cliente
Martinez	Natalia
Garcia	Carlos

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
<input type="checkbox"/> avion	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> clientes	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> destinos	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> horarios	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> reservas	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> 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.

```
1  -- 1. Insertamos en las tablas sin dependencias
2  INSERT INTO Avion (codigo, marca, estado) VALUES
3  (101, 'Boeing 737', 'Activo'),
4  (102, 'Airbus A320', 'Mantenimiento');
5
6  INSERT INTO Horarios (id_horario, hora_vuelo) VALUES
7  (1, '08:00 AM'),
8  (2, '02:30 PM'),
9  (3, '09:15 PM');
10
11 INSERT INTO Clientes (codigo, nombre, apellido, fecha_nacimiento, tipo, usuario) VALUES
12 (1, 'Juan', 'Perez', '1990-05-15', 'Frecuente', 'jperez'),
13 (2, 'Maria', 'Gomez', '1985-11-20', 'Ocasional', 'mgomez'),
14 (3, 'Carlos', 'Lopez', '2001-02-10', 'Frecuente', 'clopez'),
15 (4, 'Ana', 'Martinez', '1998-07-30', 'Nuevo', 'amartinez'),
16 (5, 'Patricio', 'Sussini', '2002-01-25', 'VIP-UTN', 'drsetup');
17
18 -- 2. Insertamos en Destinos
19 INSERT INTO Destinos (id_destino, id_horario, nombre) VALUES
20 (1, 1, 'Madrid'),
21 (2, 2, 'Miami'),
22 (3, 3, 'Tokio');
23
24 -- 3. Insertamos en Vuelos
25 INSERT INTO Vuelos (id_vuelo, disponibilidad, tipo_vuelo, codigo_avion, id_destino) VALUES
26 (501, 'Disponible', 'Internacional', 101, 1),
27 (502, 'Pocos asientos', 'Internacional', 102, 2),
28 (503, 'Disponible', 'Internacional', 101, 3);
29
30 -- 4. Insertamos las 8 Reservas
31 INSERT INTO Reservas (id_reserva, estado, id_vuelo, codigo_cliente) VALUES
32 (1001, 'Confirmada', 501, 1),
33 (1002, 'Confirmada', 501, 2),
34 (1003, 'Pendiente', 502, 3),
35 (1004, 'Confirmada', 502, 4),
```

```

36 (1005, 'Cancelada', 503, 1),
37 (1006, 'Confirmada', 503, 2),
38 (1007, 'Confirmada', 501, 5),
39 (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.