
---

## Situación

Disponemos de la siguiente tabla en MySQL:

```
CREATE TABLE libros (
   codigo INT UNSIGNED AUTO_INCREMENT,
   titulo VARCHAR(40) NOT NULL,
   autor VARCHAR(30),
   editorial VARCHAR(15),
   PRIMARY KEY (codigo),
   INDEX i_editorial (editorial),
   UNIQUE i_tituloeditorial (titulo, editorial)
);
```

---

1️⃣ ¿Cómo consultar todos los índices de la tabla "libros"?

Consulta los índices existentes en la tabla `libros` para conocer qué optimizaciones de búsqueda ya están implementadas.

2️⃣ ¿Cómo crear un índice sobre el campo "autor" para acelerar las búsquedas por autor?

Crea un índice que permita mejorar la eficiencia de las consultas basadas en el campo `autor`.

3️⃣ ¿Cómo eliminar el índice "i_editorial" de la tabla "libros"?

Elimina el índice `i_editorial` si ya no es necesario.

4️⃣ ¿Qué pasará si intentamos insertar un libro con el mismo título y editorial ya existente?

Considera que hay una restricción `UNIQUE (titulo, editorial)`, lo que impide duplicados.

5️⃣ ¿Cómo optimizar una consulta que busca libros por editorial?

Asegura que las búsquedas por `editorial` sean más rápidas utilizando índices.

6️⃣ ¿Cómo crear un índice compuesto por "titulo" y "autor"?

Genera un índice que optimice consultas donde se busquen libros por ambos campos.

7️⃣ ¿Cómo verificar si una consulta está utilizando un índice?

Consulta si MySQL está usando un índice al ejecutar una consulta específica.

8️⃣ ¿Qué sucede si intentamos definir dos claves primarias en la tabla?

Analiza qué error genera MySQL al intentar establecer más de una `PRIMARY KEY`.

9️⃣ ¿Cómo eliminar la clave primaria de la tabla "libros"?

Elimina la clave primaria de la tabla sin afectar otros índices.

🔟 ¿Cómo hacer que una búsqueda de libros ordenada por editorial sea más eficiente?

Optimiza una consulta que usa `ORDER BY editorial` para mejorar su rendimiento.

---

## 📌 Respuestas

1️⃣ Consultar todos los índices de la tabla "libros"

```
SHOW INDEX FROM libros;
```

2️⃣ Crear un índice sobre el campo "autor"

```
CREATE INDEX i_autor ON libros(autor);
```

3️⃣ Eliminar el índice "i_editorial"

```
DROP INDEX i_editorial ON libros;
```

4️⃣ Insertar un libro con el mismo título y editorial ya existente

```
INSERT INTO libros (titulo, autor, editorial)
VALUES ('El Quijote', 'Cervantes', 'Alfaguara');
INSERT INTO libros (titulo, autor, editorial)
VALUES ('El Quijote', 'Otro Autor', 'Alfaguara'); -- ❌ ERROR
```

📌 Explicación:
El segundo `INSERT` genera un error porque hay un índice `UNIQUE (titulo, editorial)`, lo que impide duplicados en esa combinación de valores.

5️⃣ Optimizar una consulta que busca libros por editorial

Si no existe un índice:

CREATE INDEX i_editorial ON libros(editorial);

Consulta optimizada:

SELECT * FROM libros WHERE editorial = 'Alfaguara';

6️⃣ Crear un índice compuesto por "titulo" y "autor"
CREATE INDEX i_tituloautor ON libros(titulo, autor);

📌 Explicación:
Este índice mejora consultas como:

SELECT * FROM libros WHERE titulo = '1984' AND autor = 'George Orwell';

7️⃣ Verificar si una consulta usa un índice
EXPLAIN SELECT * FROM libros WHERE titulo = 'Cien años de soledad';

📌 Explicación:
`EXPLAIN` muestra información sobre cómo MySQL ejecutará la consulta y si está usando un índice.

8️⃣ Intentar definir dos claves primarias
ALTER TABLE libros ADD PRIMARY KEY (editorial); -- ❌ ERROR

📌 Explicación:
MySQL no permite más de una `PRIMARY KEY` en una tabla.

9️⃣ Eliminar la clave primaria de la tabla "libros"
ALTER TABLE libros DROP PRIMARY KEY;

📌 Importante:
Si `codigo` es `AUTO_INCREMENT`, perderá esa propiedad.

🔟 Optimizar la búsqueda ordenada por editorial

Si la consulta es:

SELECT * FROM libros ORDER BY editorial;

Optimización con índice:

CREATE INDEX i_editorial_orden ON libros(editorial);

📌 Explicación:
Este índice permite que MySQL ordene los datos sin escanear toda la tabla.

---