

# MySQL

## 05\_ Índices



# MySQL\_ Índices.

## Contenido

1. Indices de tabla	4
2. Indice primery	6
3. Indice comun	8
4. Indice unico	10
5. Borrar indices	12
6. Crear indice	14

# MySQL\_ Índices.

## Introducción

Los índices en MySQL son estructuras que se utilizan para acelerar la búsqueda y recuperación de datos en una tabla. Los índices funcionan como una especie de índice en un libro, donde las palabras clave se ordenan alfabéticamente y se asocian con las páginas donde se pueden encontrar en el texto.

En MySQL, los índices se crean en una o más columnas de una tabla. Cada vez que se realiza una búsqueda en la tabla utilizando la columna indexada, el motor de la base de datos utiliza el índice para encontrar los registros más rápidamente. Los índices son especialmente útiles para tablas grandes y para consultas que involucran búsquedas en campos específicos.

Es importante tener en cuenta que la creación de índices puede tener un impacto negativo en el rendimiento de la base de datos si se utilizan en exceso o de manera incorrecta. La creación de índices adicionales puede aumentar el tiempo de escritura de la tabla y el tamaño de la base de datos, lo que puede afectar la velocidad de las operaciones de inserción y actualización.

En resumen, los índices son una herramienta importante para mejorar el rendimiento de una base de datos en MySQL al acelerar la búsqueda y recuperación de datos en una tabla. Sin embargo, es importante utilizarlos de manera cuidadosa y eficiente para minimizar el impacto en el rendimiento general de la base de datos.



# MySQL\_ Índices.

## 1. MySQL Indices

### Los índices y su finalidad

1. Para facilitar la obtención de información de una tabla se utilizan índices.
2. El índice de una tabla desempeña la misma función que el índice de un libro: permite encontrar datos rápidamente; en el caso de las tablas, localiza registros.
3. Una tabla se indexa por un campo (o varios).
4. El índice es un tipo de archivo con 2 entradas: un dato (un valor de algún campo de la tabla) y un puntero.
5. Un índice posibilita el acceso directo y rápido haciendo más eficiente las búsquedas. Sin índice, se debe recorrer secuencialmente toda la tabla para encontrar un registro.
6. El objetivo de un índice es acelerar la recuperación de información.
7. La desventaja es que consume espacio en el disco.
8. La indexación es una técnica que optimiza el acceso a los datos, mejora el rendimiento acelerando las consultas y otras operaciones. Es útil cuando la tabla contiene miles de registros.

### Los índices se usan para varias operaciones:

- - para buscar registros rápidamente.
- - para recuperar registros de otras tablas empleando "join".

Es importante identificar el o los campos por los que sería útil crear un índice, aquellos campos por los cuales se realizan operaciones de búsqueda con frecuencia.

### Hay distintos tipos de índices, a saber:

1. **"primary key"**: es el que definimos como clave primaria. Los valores indexados deben ser únicos y además no pueden ser nulos. MySQL le da el nombre "PRIMARY". Una tabla solamente puede tener una clave primaria.
2. **"index"**: crea un índice común, los valores no necesariamente son únicos y aceptan valores "null". Podemos darle un nombre, si no se lo damos, se coloca uno por defecto. "key" es sinónimo de "index". Puede haber varios por tabla.
3. **"unique"**: crea un índice para los cuales los valores deben ser únicos y diferentes, aparece un mensaje de error si intentamos agregar un registro con un valor ya existente. Permite valores nulos y pueden definirse varios por tabla. Podemos darle un nombre, si no se lo damos, se coloca uno por defecto.

**Todos los índices pueden ser multicolumna, es decir, pueden estar formados por más de 1 campo.**

## MySQL\_ Índices.

En las siguientes lecciones aprenderemos sobre cada uno de ellos.

Una tabla puede tener hasta 64 índices. Los nombres de índices aceptan todos los caracteres y pueden tener una longitud máxima de 64 caracteres. Pueden comenzar con un dígito, pero no pueden tener sólo dígitos.

Una tabla puede ser indexada por campos de tipo numérico o de tipo carácter. También se puede indexar por un campo que contenga valores NULL, excepto los PRIMARY.

"show index" muestra información sobre los índices de una tabla. Por ejemplo:

```
show index from libros;
```

# MySQL\_ Índices.

## 2. Índice de tipo primary

El índice llamado primary se crea automáticamente cuando establecemos un campo como clave primaria, no podemos crearlo directamente. El campo por el cual se indexa puede ser de tipo numérico o de tipo carácter.

Los valores indexados deben ser únicos y además no pueden ser nulos. Una tabla solamente puede tener una clave primaria por lo tanto, solamente tiene un índice PRIMARY.

Puede ser multicolumna, es decir, pueden estar formados por más de 1 campo.

Veamos un ejemplo definiendo la tabla "libros" con una clave primaria:

```
create table libros(  
  codigo int unsigned auto_increment,  
  titulo varchar(40) not null,  
  autor varchar(30),  
  editorial varchar(15),  
  primary key(codigo)  
);
```

Podemos ver la estructura de los índices de una tabla con "show index". Por ejemplo:

```
show index from libros;
```

Aparece el índice PRIMARY creado automáticamente al definir el campo "codigo" como clave primaria.

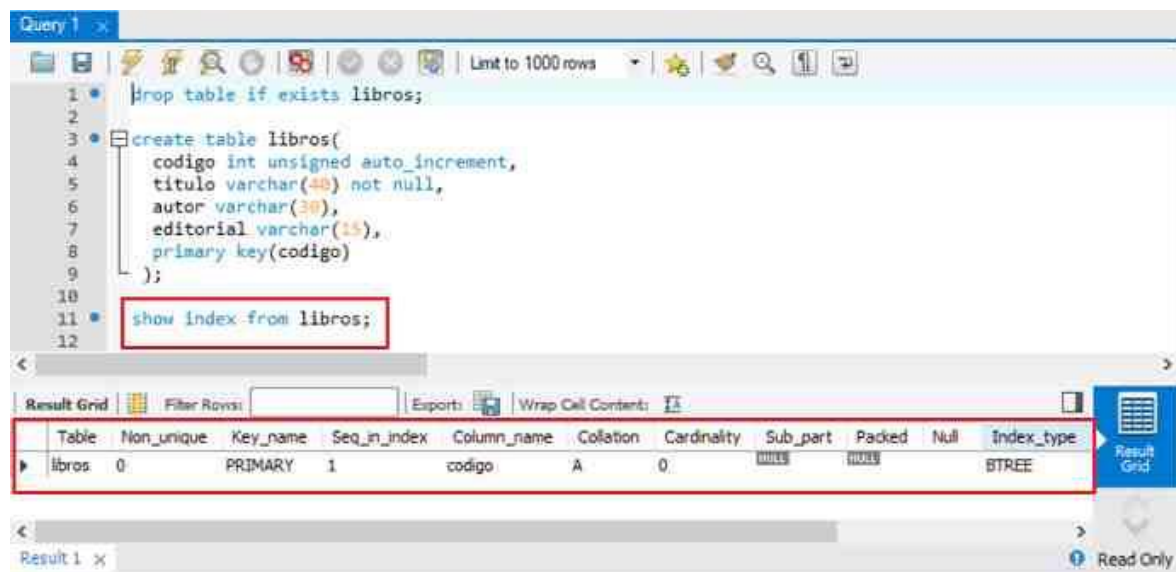
### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;  
  
create table libros(  
  codigo int unsigned auto_increment,  
  titulo varchar(40) not null,  
  autor varchar(30),  
  editorial varchar(15),  
  primary key(codigo)  
);  
  
show index from libros;
```

# MySQL\_ Índices.

Genera una salida similar a esta:



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
1 drop table if exists libros;
2
3 create table libros(
4     codigo int unsigned auto_increment,
5     titulo varchar(40) not null,
6     autor varchar(30),
7     editorial varchar(15),
8     primary key(codigo)
9 );
10
11 show index from libros;
```

The result grid displays the output of the 'show index from libros;' command:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
libros	0	PRIMARY	1	codigo	A	0				BTREE

# MySQL\_ Índices.

## 3. Índice común (index)

Dijimos que hay 3 tipos de índices. Hasta ahora solamente conocemos la clave primaria que definimos al momento de crear una tabla.

Vamos a ver el otro tipo de índice, común. Un índice común se crea con "index", los valores no necesariamente son únicos y aceptan valores "null". Puede haber varios por tabla.

Vamos a trabajar con nuestra tabla "libros".

Un campo por el cual realizamos consultas frecuentemente es "editorial", indexar la tabla por ese campo sería útil.

Creemos un índice al momento de crear la tabla:

```
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)  
);
```

Luego de la definición de los campos colocamos "index" seguido del nombre que le damos y entre paréntesis el o los campos por los cuales se indexará dicho índice.

"show index" muestra la estructura de los índices:

```
show index from libros;
```

Si no le asignamos un nombre a un índice, por defecto tomará el nombre del primer campo que forma parte del índice, con un sufijo opcional (\_2,\_3,...) para que sea único.

Ciertas tablas (MyISAM, InnoDB y BDB) soportan índices en campos que permiten valores nulos, otras no, debiendo definirse el campo como "not null".

Se pueden crear índices por varios campos:

```
create table libros(  
  codigo int unsigned auto_increment,  
  titulo varchar(40) not null,  
  autor varchar(30),  
  editorial varchar(15),  
  index i_tituloeditorial (titulo,editorial)  
);
```

Para crear índices por múltiple campos se listan los campos dentro de los paréntesis separados con comas. Los valores de los índices se crean concatenando los valores de los campos mencionados.

Recuerde que "index" es sinónimo de "key".



# MySQL\_ Índices.

## Servidor de MySQL instalado en forma local.

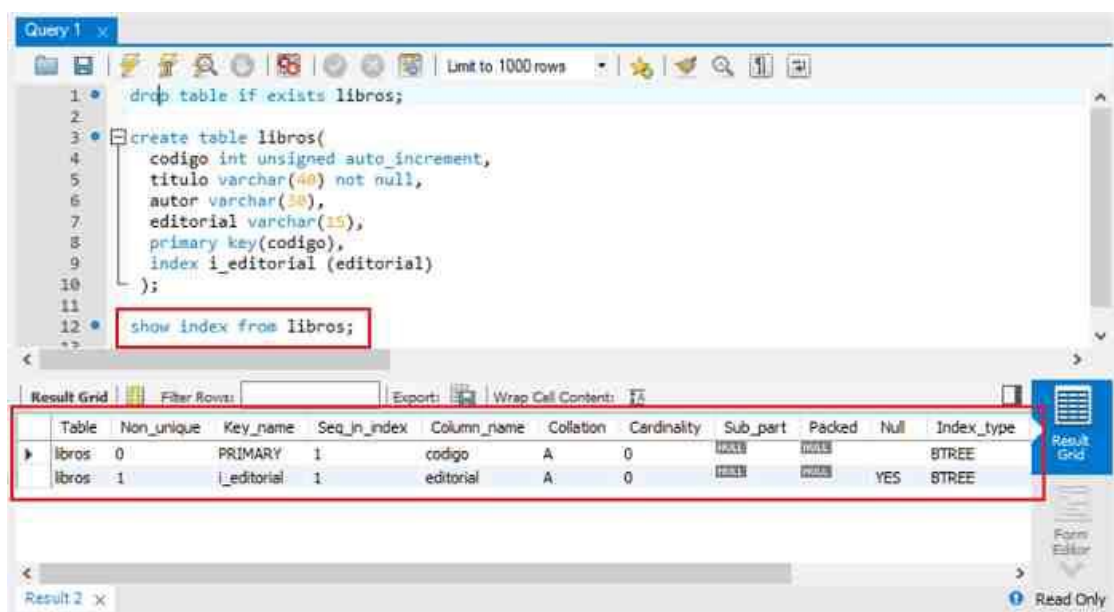
Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar un índice común:

```
drop table if exists libros;
```

```
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)  
);
```

```
show index from libros;
```

Genera una salida similar a esta:



The screenshot shows the MySQL Workbench interface. The top pane displays the following SQL queries:

```
1 • drop table if exists libros;  
2  
3 • create table libros(  
4   codigo int unsigned auto_increment,  
5   titulo varchar(40) not null,  
6   autor varchar(30),  
7   editorial varchar(15),  
8   primary key(codigo),  
9   index i_editorial (editorial)  
10  );  
11  
12 • show index from libros;
```

The bottom pane shows the result grid for the 'show index from libros;' query. The grid has columns: Table, Non\_unique, Key\_name, Seq\_in\_index, Column\_name, Collation, Cardinality, Sub\_part, Packed, Null, and Index\_type. The results are as follows:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
libros	0	PRIMARY	1	codigo	A	0				BTREE
libros	1	i_editorial	1	editorial	A	0			YES	BTREE

# MySQL\_ Índices.

## 4. Índice único (unique)

Veamos el otro tipo de índice, único. Un índice único se crea con "unique", los valores deben ser únicos y diferentes, aparece un mensaje de error si intentamos agregar un registro con un valor ya existente. Permite valores nulos y pueden definirse varios por tabla. Podemos darle un nombre, si no se lo damos, se coloca uno por defecto.

Vamos a trabajar con nuestra tabla "libros".

Crearemos dos índices únicos, uno por un solo campo y otro multicolumna:

```
create table libros(  
  codigo int unsigned auto_increment,  
  titulo varchar(40) not null,  
  autor varchar(30),  
  editorial varchar(15),  
  unique i_codigo(codigo),  
  unique i_tituloeditorial (titulo,editorial)  
);
```

Luego de la definición de los campos colocamos "unique" seguido del nombre que le damos y entre paréntesis el o los campos por los cuales se indexará dicho índice.

"show index" muestra la estructura de los índices:

```
show index from libros;
```

RESUMEN: Hay 3 tipos de índices con las siguientes características:

Tipo	Nombre	Palabra clave	Valores únicos	Acepta null	Cantida
clave primaria	PRIMARY	no	Si	No	1
común	darlo o por defecto	"index" o "key"	No	Si	varios
único	darlo o por defecto	"unique" Si	Si	varios	

# MySQL\_ Índices.

## Servidor de MySQL instalado en forma local.

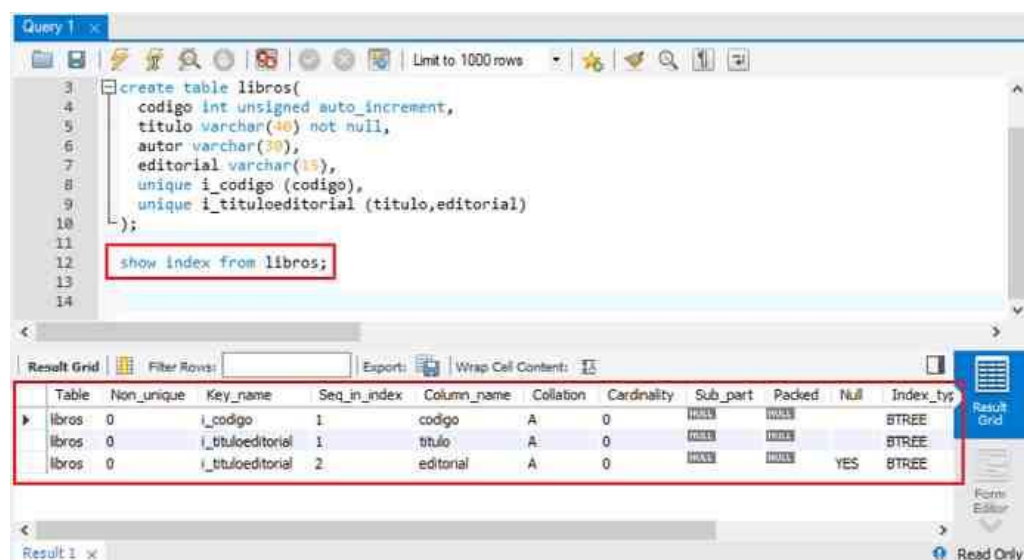
Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL donde creamos dos índices únicos:

```
drop table if exists libros;
```

```
create table libros(  
  codigo int unsigned auto_increment,  
  titulo varchar(40) not null,  
  autor varchar(30),  
  editorial varchar(15),  
  unique i_codigo (codigo),  
  unique i_tituloeditorial (titulo,editorial)  
);
```

```
show index from libros;
```

Genera una salida similar a esta:



The screenshot shows the MySQL Workbench interface. The top pane displays the SQL query: `create table libros(`, `codigo int unsigned auto_increment,`, `titulo varchar(40) not null,`, `autor varchar(30),`, `editorial varchar(15),`, `unique i_codigo (codigo),`, `unique i_tituloeditorial (titulo,editorial)`, `);`, and `show index from libros;`. The bottom pane shows the result grid for the `show index from libros;` query. The result grid has columns: Table, Non\_unique, Key\_name, Seq\_in\_index, Column\_name, Collation, Cardinality, Sub\_part, Packed, Null, and Index\_type. The data is as follows:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
libros	0	i_codigo	1	codigo	A	0				BTREE
libros	0	i_tituloeditorial	1	titulo	A	0				BTREE
libros	0	i_tituloeditorial	2	editorial	A	0			YES	BTREE

# MySQL\_ Índices.

## 5. Borrar indice (drop index)

Para eliminar un índice usamos "drop index". Ejemplo:

```
drop index i_editorial on libros;  
drop index i_tituloeditorial on libros;
```

Se elimina el índice con "drop index" seguido de su nombre y "on" seguido del nombre de la tabla a la cual pertenece.

Podemos eliminar los índices creados con "index" y con "unique" pero no el que se crea al definir una clave primaria. Un índice PRIMARY se elimina automáticamente al eliminar la clave primaria (tema que veremos más adelante).

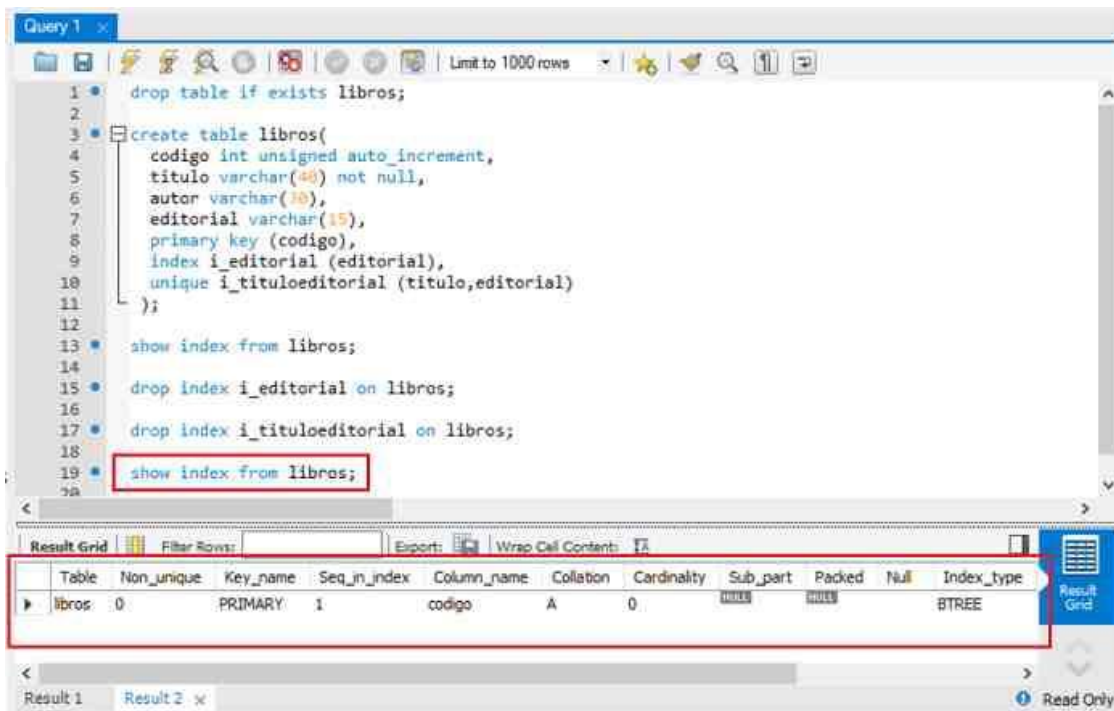
### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL que nos permiten eliminar índices definidos en una tabla:

```
drop table if exists libros;  
  
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)  
);  
  
show index from libros;  
  
drop index i_editorial on libros;  
  
drop index i_tituloeditorial on libros;  
  
show index from libros;
```

# MySQL\_ Índices.

Genera una salida similar a esta:



The screenshot shows a MySQL query editor with the following SQL code:

```
1 drop table if exists libros;
2
3 create table libros(
4     codigo int unsigned auto_increment,
5     titulo varchar(40) not null,
6     autor varchar(30),
7     editorial varchar(15),
8     primary key (codigo),
9     index i_editorial (editorial),
10    unique i_tituloeditorial (titulo,editorial)
11 );
12
13 show index from libros;
14
15 drop index i_editorial on libros;
16
17 drop index i_tituloeditorial on libros;
18
19 show index from libros;
```

The result grid shows the output of the last query:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
libros	0	PRIMARY	1	codigo	A	0				BTREE

# MySQL\_ Índices.

## 6. Agregar índices a tablas existentes (create index)

Podemos agregar un índice a una tabla existente.

Para agregar un índice común a la tabla "libros" tipeamos:

```
create index i_editorial on libros (editorial);
```

Entonces, para agregar un índice común a una tabla existente usamos "create index", indicamos el nombre, sobre qué tabla y el o los campos por los cuales se indexará, entre paréntesis.

Para agregar un índice único a la tabla "libros" tipeamos:

```
create unique index i_tituloeditorial on libros (titulo,editorial);
```

Para agregar un índice único a una tabla existente usamos "create unique index", indicamos el nombre, sobre qué tabla y entre paréntesis, el o los campos por los cuales se indexará.

Un índice PRIMARY no puede agregarse, se crea automáticamente al definir una clave primaria.

### Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para crear índices a una tabla existente:

```
drop table if exists libros;

create table libros(
  codigo int unsigned auto_increment,
  titulo varchar(40) not null,
  autor varchar(30),
  editorial varchar(15),
  primary key (codigo)
);

show index from libros;

create index i_editorial on libros (editorial);

show index from libros;

create unique index i_tituloeditorial on libros (titulo,editorial);

show index from libros;
```

# MySQL\_ Índices.

Genera una salida similar a esta:

The screenshot shows a MySQL Query Editor window with a query titled "Query 1". The query contains the following SQL commands:

```
1 drop table if exists libros;
2
3 create table libros(
4     codigo int unsigned auto_increment,
5     titulo varchar(40) not null,
6     autor varchar(30),
7     editorial varchar(15),
8     primary key (codigo)
9 );
10
11 show index from libros;
12
13 create index i_editorial on libros (editorial);
14
15 show index from libros;
16
17 create unique index i_tituloeditorial on libros (titulo,editorial);
18
19 show index from libros;
```

The result of the final query is displayed in a table with the following columns: Table, Non unique, Key name, Seq in index, Column name, Collation, Cardinality, Sub part, Packed, Null, and Index type. The table contains four rows of index information for the 'libros' table.

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type
libros	0	PRIMARY	1	codigo	A	0				BTREE
libros	0	i_tituloeditorial	1	titulo	A	0				BTREE
libros	0	i_tituloeditorial	2	editorial	A	0			YES	BTREE
libros	1	i_editorial	1	editorial	A	0			YES	BTREE

# MySQL\_ Índices.

## 7. Otros tipos de índices

**FULLTEXT:** estos índices se emplean para realizar búsquedas sobre texto (CHAR, VARCHAR y TEXT). Estos índices se componen por todas las palabras que están contenidas en la columna (o columnas) que contienen el índice. No aplica ninguna restricción especial a los datos de la columna (o columnas) que componen el índice sino que se emplea simplemente para mejorar el tiempo de ejecución de las consultas. Este tipo de índices sólo están soportados por [InnoDB y MyISAM en MySQL 5.7.](#)

Generalmente cuando queremos buscar una palabra o expresión mediante coincidencias en MySQL utilizamos LIKE ('%palabra%'), en caso que estamos buscando más de una palabra usamos LIKE '%palabra1%palabra2%', pero lo que muchos no saben es que esta forma de buscar le toma mucho tiempo al motor de base de datos encontrar los resultados y precisamente para eso es que se utiliza FullText Index que en síntesis es un índice para búsquedas sobre campos de texto solamente.. la sintaxis es fácil, MATCH() ... AGAINST(), donde MATCH se utiliza para especificar el nombre de la columna que estamos buscando y AGAINST para describir la palabra que se desea encontrar..

Pasemos al ejemplo para comprender mejor como funciona, entonces vamos a crear una tabla llamada.. no sé .. (libros) que tiene un índice de búsqueda de texto para los campos titulo y descripción.

```
CREATE TABLE IF NOT EXISTS libros (  
  id INT NOT NULL AUTO_INCREMENT ,  
  titulo VARCHAR(50) NOT NULL ,  
  descripcion TEXT NOT NULL ,  
  PRIMARY KEY (id) ,  
  FULLTEXT INDEX Buscar (titulo ASC, descripcion ASC) )  
ENGINE = MyISAM;
```

Y ahora insertemos unos cuantos registros para hacer las pruebas.

```
INSERT INTO libros (titulo, descripcion) VALUES  
( 'NodeJS para principiantes', 'Aprende como construir una aplicación web completa con  
Javascript.' ),  
( 'Descubriendo NodeJS', 'Está pensado para personas que tengan algo de conocimiento de  
JavaScript.' ),  
( 'Mastering NodeJS', 'Node is an exciting new platform developed by Ryan Dahl.' ),  
( 'Node para frontenders', 'Si tu sabes como usar JavaScript en el navegador...' ),  
( 'Descubriendo Node y Express', 'La mayoría de la gente quería ver algo de Node.js y Express..' );
```

Y ahora vamos a probarlo y ver que tal funciona.

```
mysql> SELECT * FROM libros WHERE MATCH(titulo, descripcion) AGAINST ('node');  
Empty set (0.00 sec)
```

Podríamos pensar que algo anda mal aquí, pero no es así.. por defecto, MySQL utiliza el modo de lenguaje natural (IN NATURAL LANGUAGE MODE), lo que indica que si la palabra buscada existe en más del 50% de los resultados, la salida de la consulta sería básicamente que no existen coincidencias (digamos que evita consultas genéricas).



## MySQL\_ Índices.

Entonces haciendo nuevamente la búsqueda, ahora cambiando el modificador hacia “IN BOOLEAN MODE” podremos evitar esa restricción.

```
mysql> SELECT * FROM libros WHERE MATCH(titulo, descripcion) AGAINST ('node' IN BOOLEAN MODE);
```

id	titulo	descripcion
3	Mastering NodeJS	Node is an exciting new platform developed by Ryan Dahl.
4	Node para frontenders	Si tu sabes como usar JavaScript en el navegador...
5	Descubriendo Node y Express	La mayoría de la gente quería ver algo de Node.js y Express..

```
3 rows in set (0.00 sec)
```

El uso del modificador booleano (IN BOOLEAN MODE) también nos permite utilizar algunos operadores que nos ayudan a hacer nuestras búsquedas más efectivas, como por ejemplo la siguiente consulta, que es similar a la anterior, pero con la mínima diferencia de que ahora estoy buscando libros que contengan node, pero no javascript.

```
mysql> SELECT * FROM libros WHERE MATCH(titulo, descripcion) AGAINST ('+node -javascript' IN BOOLEAN MODE);
```

id	titulo	descripcion
3	Mastering NodeJS	Node is an exciting new platform developed by Ryan Dahl.
5	Descubriendo Node y Express	La mayoría de la gente quería ver algo de Node.js y Express..

```
2 rows in set (0.00 sec)
```

**SPATIAL:** estos índices se emplean para realizar búsquedas sobre datos que componen formas geométricas representadas en el espacio. Este tipo de índices sólo están soportados por InnoDB y MyISAM en MySQL 5.7.

MySQL puede crear índices espaciales utilizando una sintaxis similar a la que se utiliza para crear índices normales, pero extendida con la palabra clave SPATIAL. Las columnas espaciales que están indexadas, deben ser declaradas, actualmente, como NOT NULL. Los siguientes ejemplos demuestran cómo crear índices espaciales.

- Con CREATE TABLE:

```
mysql> CREATE TABLE geom (g GEOMETRY NOT NULL, SPATIAL INDEX(g));
```

- Con ALTER TABLE:

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
```

- Con CREATE INDEX:

```
mysql> CREATE SPATIAL INDEX sp_index ON geom (g);
```

# MySQL\_ Índices.

Para eliminar índices espaciales, utilice ALTER TABLE o DROP INDEX:

- Con ALTER TABLE:  
`mysql> ALTER TABLE geom DROP INDEX g;`
- Con DROP INDEX:  
`mysql> DROP INDEX sp_index ON geom;`

Ejemplo: Suponga una tabla geom que contiene más de 32000 geometrías, que están almacenadas en la columna g del tipo GEOMETRY. La tabla también tiene una columna AUTO\_INCREMENT llamada fid para almacenar valores de ID de objetos.

```
mysql> DESCRIBE geom;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| fid   | int(11) |      | PRI | NULL    | auto_increment |
| g     | geometry |      |      |          |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT COUNT(*) FROM geom;
+-----+
| count(*) |
+-----+
| 32376 |
+-----+
1 row in set (0.00 sec)
```

Para añadir un índice espacial en la columna g, utilice esta sentencia:

```
mysql> ALTER TABLE geom ADD SPATIAL INDEX(g);
Query OK, 32376 rows affected (4.05 sec)
Records: 32376 Duplicates: 0 Warnings: 0
```

## MySQL\_ Índices.