

Índices

Índice

- 01 [Introducción](#)
- 02 [Sintaxis](#)
- 03 [Tipos de índices](#)
- 04 [Estructuras de almacenamiento](#)



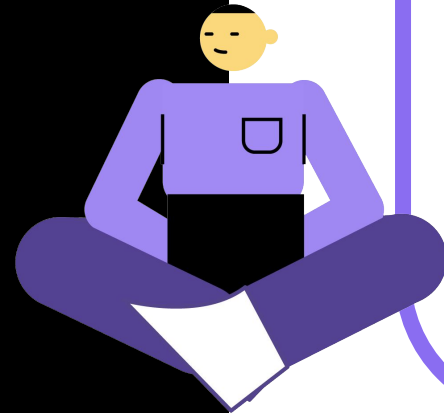
01

Introducción



Un índice es una estructura de datos.

Su función es aumentar la velocidad de las consultas en una tabla.



Instrucciones INSERT, UPDATE y SELECT

Veamos la diferencia de performance entre las instrucciones utilizando índices.

- Las instrucciones INSERT y UPDATE llevan más tiempo en tablas con índices.
- Las instrucciones SELECT son más rápidas en tablas con índices.

Esto se debe a que, para realizar una inserción o actualización, una base de datos también necesita agregar o actualizar los valores en el índice.

Índices agrupados

El índice agrupado (CLUSTERED) se identifica como primario (PRIMARY), se almacena junto con los datos en la propia tabla y ordena físicamente los registros

Cada vez que se insertan o actualizan nuevos datos, los datos se reescriben en el índice para mantenerlos ordenados.

Solo puede haber un índice principal por tabla.

Por defecto, son índices primarios los campos definidos como clave primaria, clave foránea y constraint UNIQUE.

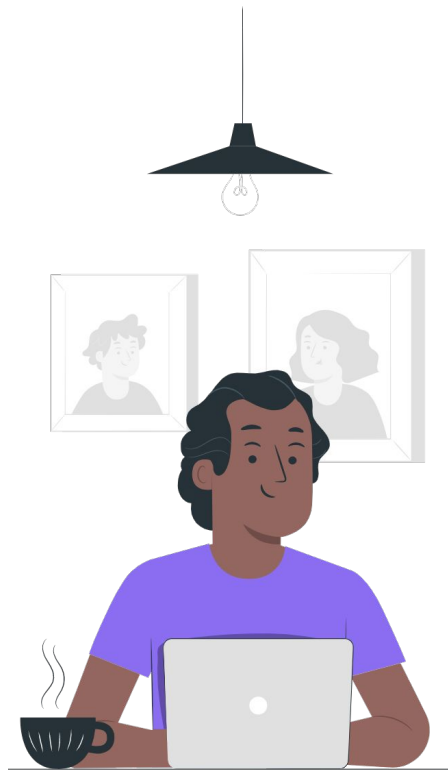
Índices no-agrupados

Se pueden crear otros índices en campos diferentes y de uso más frecuente. Estos son índices secundarios, no agrupados (NON-CLUSTERED).

El índice no agrupado almacena la columna de clave principal en su estructura, así como las columnas especificadas para crear el índice. Esto crea un puntero, que apunta a la posición real de los datos.

Se pueden crear múltiples índices no agrupados en una tabla. El tamaño de cada índice se suma al tamaño de la tabla.

Tengamos en cuenta que hay que evaluar la cantidad de índices que se crean para que no afecte el rendimiento.



Acciones del índice en MySQL

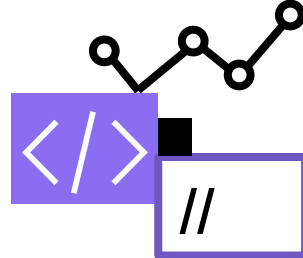
En general, MySQL usa índices para realizar las siguientes acciones:

- ❑ Encontrar rápidamente registros que coincidan con una cláusula **WHERE**.
- ❑ Recuperar registros de otras tablas al usar operaciones **JOIN**.
- ❑ Disminuir el tiempo de ejecución de las consultas con ordenación (**ORDER BY**) o agrupamiento (**GROUP BY**), en el caso que todas las columnas utilizadas en los criterios hagan parte de un índice.

Los índices deben ser del mismo tipo y tamaño, para aumentar la eficiencia de la búsqueda.

02

Sintaxis



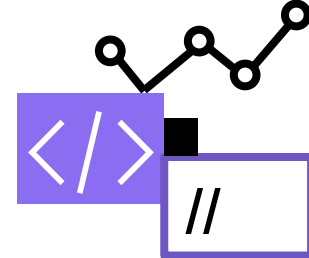
Creación de índice

- Para crear un índice, utilizamos el comando CREATE:

```
SQL CREATE INDEX NOMBRE_INDICE ON NOMBRE_TABLA(columna, columna1...);
```

→ Ejemplo:




```
SQL CREATE INDEX idx_store ON store(name);
```

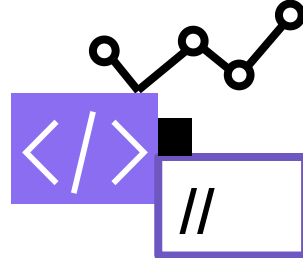


Visualización de índice

- Para visualizar los índices de una tabla, utilizamos la cláusula SHOW INDEX. En la segunda imagen, podemos observar el resultado de su ejecución:

```
SQL SHOW INDEX FROM store;
```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 								
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
▶	store	0	PRIMARY	1	CustomerID	A	699	NULL
	store	1	idx_store	1	Name	A	699	NULL



Eliminación de índice

- Para eliminar un índice, utilizamos ALTER TABLE, seguido de la cláusula DROP INDEX:

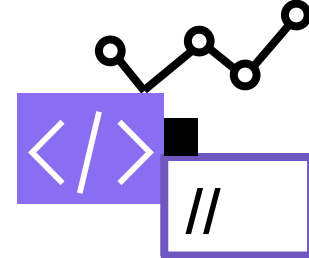
SQL

```
ALTER TABLE NOMBRE_TABLA  
DROP INDEX NOMBRE_INDICE;
```

→ Ejemplo:

SQL

```
ALTER TABLE store  
DROP INDEX idx_store;
```



Confirmación de exclusión de índice

- Enumerarlos índices para verificar si el índice fue excluido, utilizando la cláusula SHOW INDEX.

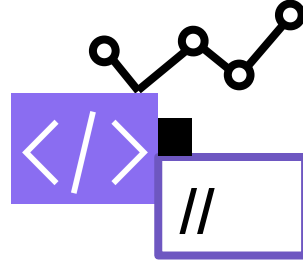
```
SQL SHOW INDEX FROM store;
```

Observe abajo que el índice **idx_store** fue excluido.

Result Grid								
		Filter Rows:		Export:		Wrap Cell Content: IA		
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
▶	store	0	PRIMARY	1	CustomerID	A	699	NULL

03

Tipos de índices



Index (no exclusivo)

INDEX es un tipo de índice normal, no exclusivo. Esto significa que admite valores duplicados para las columnas que lo componen.

Es ampliamente utilizado para mejorar el tiempo de ejecución de consultas.

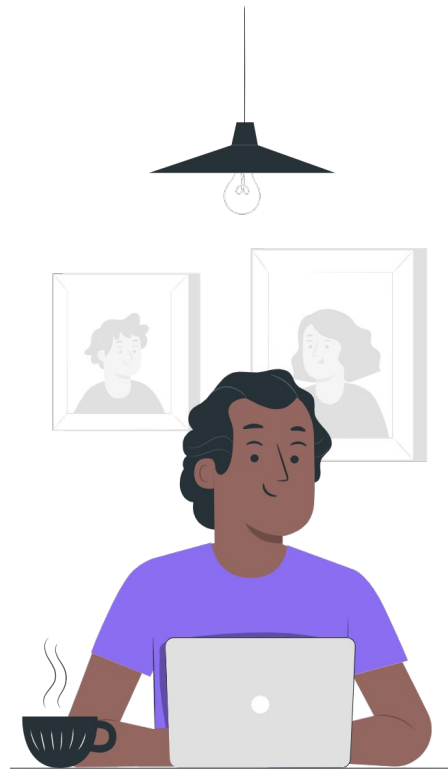
SQL

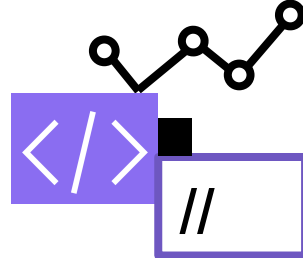
```
CREATE INDEX NOMBRE_INDICE ON NOMBRE_TABLA(columna, columna1...);
```

Unique

El tipo de índice unique indica que todas las columnas utilizadas en la creación del índice deben tener un valor único.

Es decir, la columna —o columnas— que componen el índice no puede tener valores repetidos.





Unique

- El UNIQUE puede ser creado junto con la tabla:

```
SQL CREATE TABLE NOMBRE_TABLA (nombre_columna1, nombre columna 2...)
      UNIQUE [nombre_indice] (nombre_columna)
```

También es posible agregarlo después de crear la tabla:

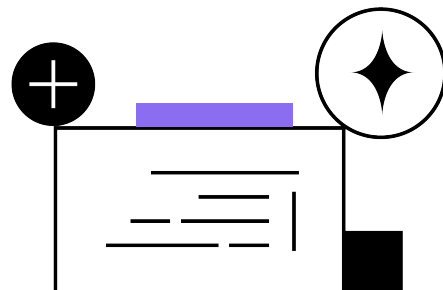
```
SQL ALTER TABLE nombre_tabla ADD UNIQUE [nombre_indice] (nombre_columna,
[nombre_columna2]...);
```

FullText

Este índice se utiliza para realizar búsquedas en cadenas de texto con mayor precisión.

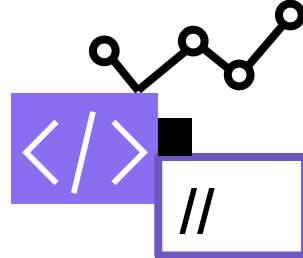
FullText es más potente que usar like porque, además de ordenar el resultado por similitud semántica, ofrece más opciones para filtrar la consulta.

Es adecuado para aplicaciones con una gran cantidad de texto y que necesitan realizar búsquedas basadas en la relevancia.



Ejemplos:

- ❑ Páginas de búsqueda que devuelven los resultados más relevantes al principio.
- ❑ Bibliotecas virtuales.
- ❑ Búsquedas en archivos de registro.
- ❑ Búsquedas en documentos almacenados en la base de datos.

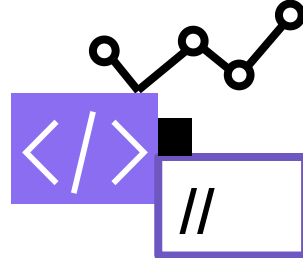


FullText: comando CREATE INDEX

- Los tipos de datos admitidos son VARCHAR, TEXT y CHAR.
- Este índice puede ser creado a través de los comandos CREATE INDEX, CREATE TABLE o ALTER TABLE.

Ejemplo **CREATE INDEX**:

```
SQL CREATE FULLTEXT INDEX NOMBRE_INDICE ON NOMBRE_TABLA  
(COLUMN1,COLUMN2...) ;
```

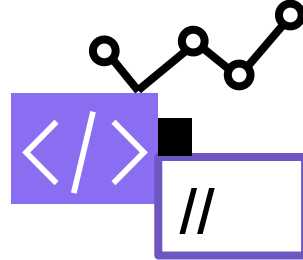


FullText: comando CREATE TABLE

Ejemplo **CREATE TABLE**:

SQL

```
CREATE TABLE NOMBRE_TABLA(  
    columna tipo_dato(),  
    columna1 tipo_dato(),  
    PRIMARY KEY (columna),  
    FULLTEXT (columna, columna1));
```



FullText: comando ALTER TABLE

Ejemplo **ALTER TABLE**:

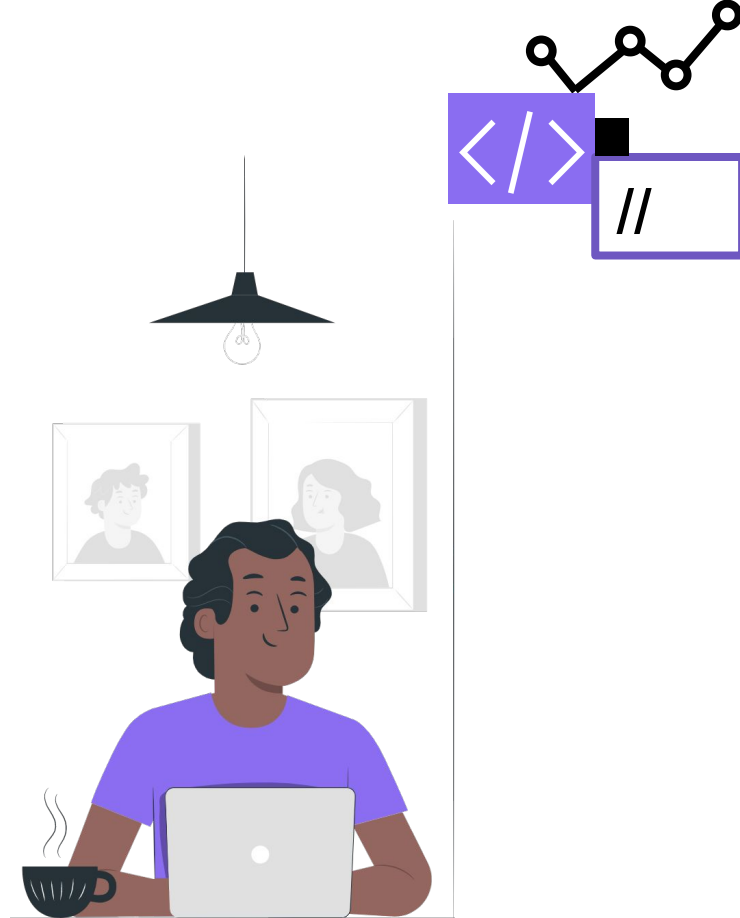
```
SQL ALTER TABLE NOME_TABELA ADD FULLTEXT(coluna, coluna1...); ;
```

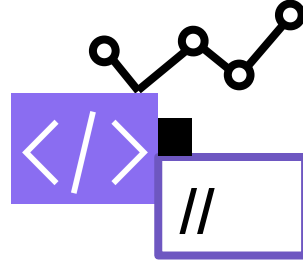
FullText: MATCH AGAINST

Las tablas que se someterán a alguna rutina de importación no se recomiendan para este tipo de índice. Esto se debe a que, en este caso, la carga de registros es más lenta. Por lo tanto, debe completar la importación antes de crear el índice.

Para consultas con un índice FullText, usamos la función MATCH AGAINST. Esta función recibe el nombre de los campos y el valor a buscar, respectivamente.

En la consulta, debemos informar **todas las columnas** que pertenecen al índice.





Eliminación del índice FullText

- Para eliminar el índice FULLTEXT, utilizamos ALTER TABLE seguido de la cláusula DROP INDEX.

```
SQL ALTER TABLE NOME_TABELA  
DROP INDEX NOME_INDICE;
```

Listamos los índices de la tabla para comprobar si se eliminó:

```
SQL SHOW INDEX FROM NOME_TABELA;
```

Consideraciones FullText

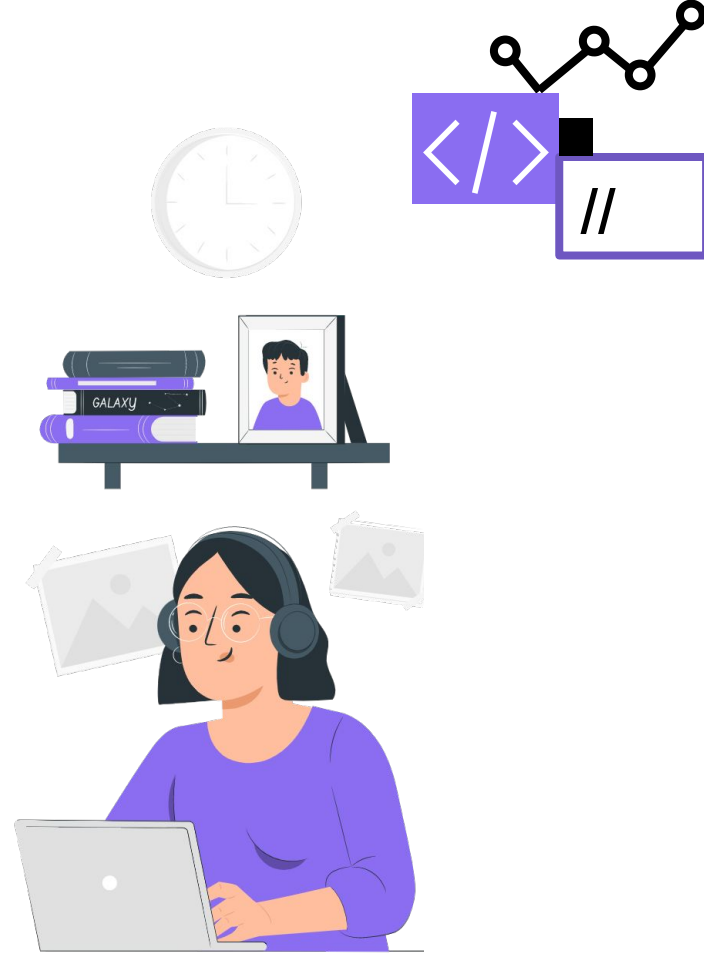
Veamos algunos puntos importantes sobre FullText.

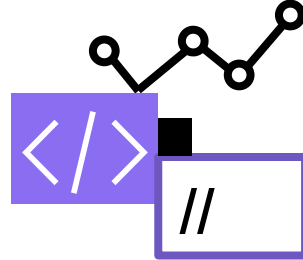
- En consultas con índice FullText, MySQL descarta palabras con menos de 4 caracteres.
- Expresiones como "de", "que" y "o" se excluyen automáticamente de la búsqueda.
- Una palabra presente en más del 50% de los registros será excluida de la búsqueda.

Tipos de búsqueda FullText

IN NATURAL LANGUAGE MODE - Es el tipo de búsqueda de Fulltext predeterminado. No hay operadores especiales y las búsquedas consisten en una o más palabras clave separadas por una coma. Las búsquedas se devuelven en orden descendente de relevancia.

IN BOOLEAN MODE - Permite el uso de varios operadores especiales. Las búsquedas no se devuelven en orden de relevancia, ni se aplica el límite del 50% y puede buscar palabras con 4 caracteres o menos.





Tipos de búsqueda FullText

- Ejemplo:

SQL

```
SELECT FirstName, LastName, EmailAddress, Phone  
FROM adventureworks.contact  
WHERE MATCH (FirstName, EmailAddress) AGAINST ('pet*' IN BOOLEAN MODE);
```

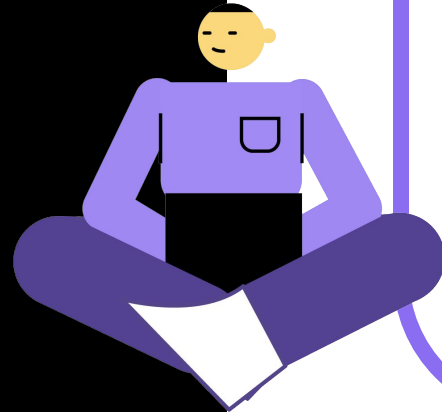
04

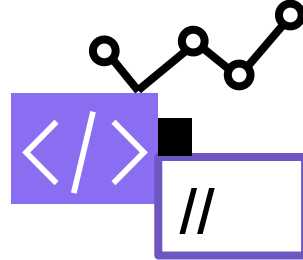
Estructuras de almacenamiento



Los índices crean estructuras para almacenar los datos de la tabla indexada.

Cada tipo de estructura tiene sus características y finalidades.



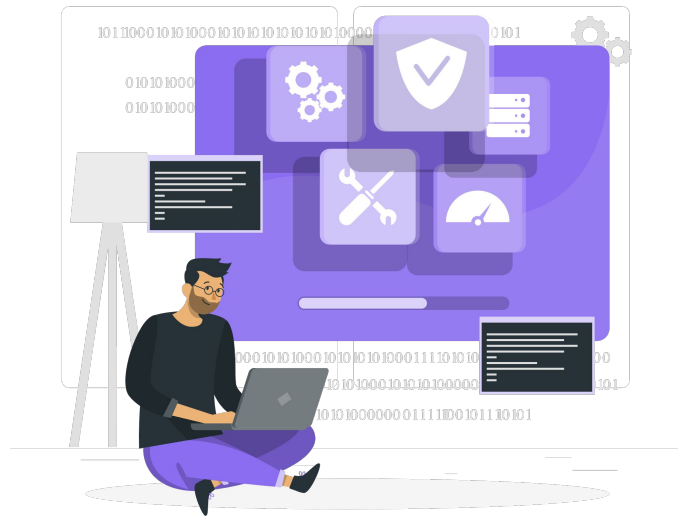


Estructuras de almacenamiento: B-Tree

Se usa un índice de árbol B para comparaciones del tipo =, >, <, >=, <=, BETWEEN y LIKE —siempre que se use en constantes que no comiencen con %—.

Para realizar búsquedas utilizando este tipo de índice, se utilizará cualquier columna —o conjunto de columnas que forme el prefijo del índice.

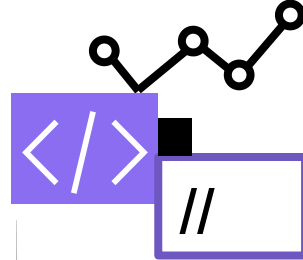
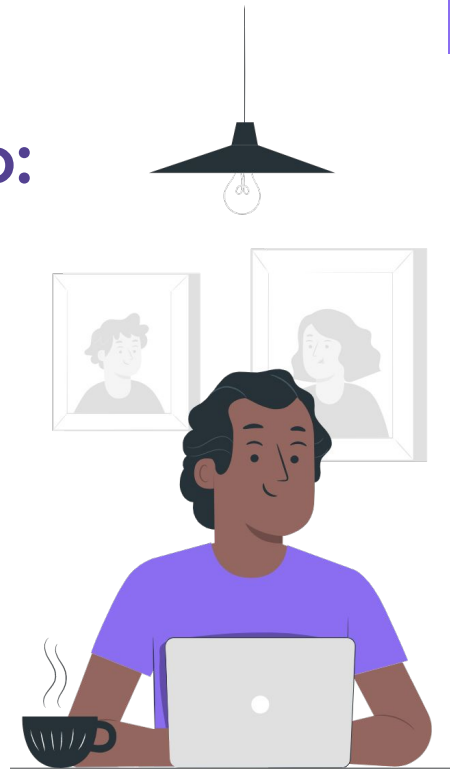
Ejemplo: si un índice está compuesto por columnas [A, B, C], las búsquedas se pueden realizar en: [A], [A, B] o [A, B, C]



Estructuras de almacenamiento: Hash

El tipo de estructura Hash solo se usa para comparaciones de tipo = o <=>.

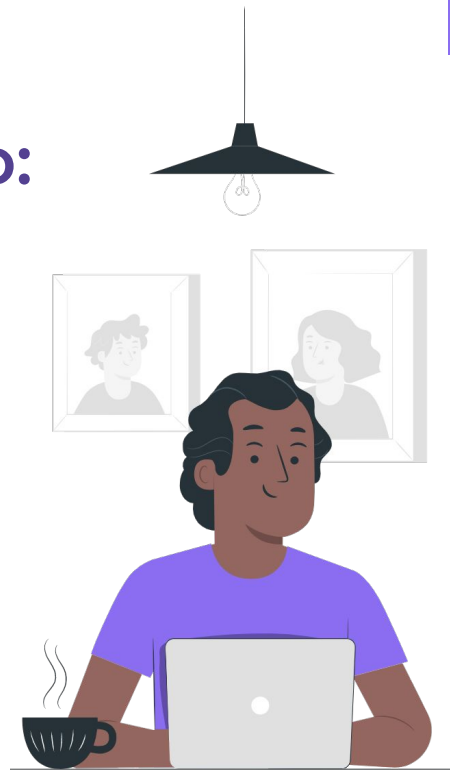
No se utilizan para operadores de comparación, como encontrar un rango de valores.

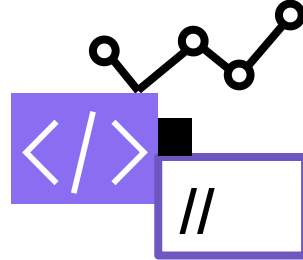


Estructuras de almacenamiento: Hash

El optimizador no puede usar un índice hash para acelerar las operaciones **ORDER BY** —el Hash no se puede usar para buscar la siguiente entrada en orden—.

Para realizar búsquedas utilizando este tipo de índice, se utilizarán **todas las columnas** que componen el índice.





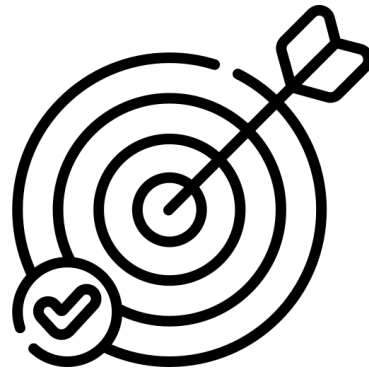
Estructuras de almacenamiento: consideraciones

A veces, MySQL no usa un índice, incluso si hay uno disponible.

Esto ocurre, por ejemplo, cuando el optimizador estima que usar el índice requeriría que MySQL acceda a un porcentaje muy grande de las filas de la tabla.

En este caso, es probable que la exploración de una tabla sea mucho más rápida, ya que requiere menos búsquedas.

Sin embargo, si logramos limitar el rango de búsqueda, MySQL usa el índice disponible, agilizando la consulta.



¡Muchas gracias!