OPTIMIZACIÓN DE CONSULTAS

Cuando en una consulta buscamos un valor específico en la columna de una tabla se recorre toda la tabla comparando fila a fila hasta encontrar el valor que coincide con el valor buscado. Para tablas con pocas filas puede que esto no sea un problema, pero imagina las operaciones de comparación que tendría que realizar sobre una tabla con millones de filas.

La mejor forma de optimizar el rendimiento de una consulta es creando índices sobre las columnas que se utilizan en la cláusula `WHERE`. Los índices se comportan como punteros sobre las filas de la tabla y nos permiten determinar rápidamente cuáles son las filas que cumplen la condición de la cláusula `WHERE`.

Todos los tipos de datos de MySQL pueden ser indexados, pero tenga en cuenta que no es conveniente crear un índice para cada una de las columnas de una tabla, ya que el exceso de índices innecesarios pueden provocar un incremento del espacio de almacenamiento y un aumento del tiempo para MySQL a la hora de decidir qué índices necesita utilizar. Los índices además añaden una sobrecarga a las operaciones de inserción, actualización y borrado, porque cada índice tiene que ser actualizado después de realizar cada una de estas operaciones.

Se debe intentar reducir el tiempo de respuesta de la consulta utilizando el menor número de índices posible.

1. Tipos de índices

Los sistemas gestores de bases de datos utilizan diferentes tipos de índices, algunos de los más utilizados son los siguientes:

- **Índices de clave primaria.**
- **Índices de clave ajena.**
- **Índices únicos.**
- **Índices con valores repetidos.** Permiten optimizar búsquedas sobre columnas que contienen valores repetidos.
- **Índices de múltiples columnas.** Utilizan varias columnas en lugar de una sola.
- **Índices de texto completo.** Se utilizan para optimizar las búsquedas en campos de texto.

2. Gestión de índices

La sintaxis para crear índices en MySQL es la siguiente:

```
CREATE [UNIQUE|FULLTEXT] INDEX nombre_índice ON nombre_tabla
(nombre_columna);
```

Ejemplos:

- Crear un índice llamado idx_pais sobre la columna país de la tabla cliente

  ```
  CREATE INDEX idx_pais ON cliente (pais);
  ```

- Crear un índice de tipo unique con el nombre idx_email sobre la columna email de la tabla empleado.

  ```
  CREATE UNIQUE INDEX idx_email ON empleado (email);
  ```

- Crear un índice con el nombre idx_apellido_nombre compuesto por las columnas apellido y nombre de la tabla cliente.

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

  En este ejemplo se crea un índice con varias columnas que será útil en las consultas donde se realicen búsquedas por el apellido y el nombre del cliente, o solamente por el apellido, pero no será útil en aquellas consultas donde sólo se utilice el nombre, ya que tendría que recorrer toda la tabla para encontrarlo.

- Podemos crear un índice con menos caracteres que el campo para reducir el tamaño que ocupara el índice. Esto se realizaría cuando los campos tienes una cantidad de caracteres muy elevada. Por ejemplo, tenemos el campo nombre de la tabla cliente de tipo varchar(50). Podemos declarar un índice sobre el campo nombre solo de 20 caracteres.

  ```
  CREATE INDEX idx_nombre ON cliente (nombre(20));
  ```

  Hay que tener en cuenta que para que el índice sea eficiente habrá que buscar un tamaño de índice adecuado que nos permita diferenciarlos con el menor número de caracteres posibles.

- Para buscar palabras o frases completas (cuando en la consulta utilizamos el like) en un campo de texto se utiliza el índice FULLTEXT. Solo se puede utilizar un índice FULLTEXT por tabla.

Dada la siguiente consulta:

```
SELECT *
FROM producto
WHERE nombre LIKE '%acero%' OR descripcion LIKE '%acero%';
```

Para realizar búsquedas más eficientes creamos un índice de tipo `FULLTEXT` compuesto por las columnas `nombre` y `descripcion` de la tabla `producto`.

```
CREATE FULLTEXT INDEX idx_nombre_descripcion ON producto(nombre,
descripcion);
```

Siempre que utilicemos un índice FULLTEXT tenemos que realizar la consulta haciendo uso de `MATCH` y `AGAINST` de la siguiente forma:

```
SELECT *
FROM producto
WHERE MATCH(nombre, descripcion) AGAINST ('acero');
```

En MATCH ponemos los nombres de los campos indexados y en AGAINST la palabra o frase a buscar.

Si en la consulta tenemos varias condiciones una con like y otra normal se realizara de la siguiente forma:

Tenemos la siguiente consulta:

Select *

From producto

Where nombre like 'm%' and descripción = 'mesa baja';

Crearemos un índice fulltext para el campo nombre porque es el que utiliza el operados like

Create fulltext index idx_nombre on producto(nombre)

Y la consulta la realizaremos de la siguiente forma:

```
SELECT *
FROM producto
WHERE MATCH(nombre) AGAINST ('m* in Boolean mode') and
descripcion='mesa baja';

IN BOOLEAN MODE permite usar * como comodín para prefijos
Ejemplo:
Que empiece por l → against ('l* in Boolean mode')
```

- También es posible crear los índices anteriores con la sentencia `ALTER TABLE`

```
ALTER TABLA nombre_tabla
ADD INDEX nombre_indice (nombre_columna);
```

Ejemplos:

```
ALTER TABLE cliente
ADD INDEX idx_pais (pais);

ALTER TABLE empleado
ADD UNIQUE INDEX idx_email (email);


ALTER TABLE cliente
ADD INDEX idx_apellido_nombre (apellido, nombre);

ALTER TABLE cliente
ADD INDEX idx_nombre (nombre(20));

ALTER TABLE producto
ADD FULLTEXT INDEX idx_nombre_descripcion (nombre, descripcion);
```

- También es posible crear índices al crear la tabla con la sentencia `CREATE TABLE`

Ejemplos:

```
CREATE TABLE cliente (
  id VARCHAR(10) PRIMARY KEY,
  nombre VARCHAR(50),
  email VARCHAR(15),
  telefono VARCHAR(9),
  INDEX idx_nombre (nombre)
);
```

3. Mostrar índices

Para mostrar los índices de una tabla utilizaremos la siguiente sintaxis:

SHOW INDEX FROM nombre_tabla;

Por ejemplo, para mostrar los índices de la tabla cliente pondríamos lo siguiente:

SHOW INDEX FROM cliente;

Donde el resultado sería:

Table	Non_unique	Key_name	Seq_in_index	Column_name
cliente	0	PRIMARY	1	codigo_cliente
cliente	1	codigo_empleado_rep_ventas	1	codigo_empleado_rep_ventas

Donde vemos que la tabla cliente tiene dos índices, la columna código_cliente que es la clave primaria y la columna código_empleado_rep_ventas.

- También podemos obtener información sobre los índices que existen en una tabla con la sentencia `DESCRIBE`. Aquí se nos presentaran todos los campos de la tabla y en la columna Key se indicaran los campos que son índices.

  Ejemplo:

```
DESCRIBE cliente;
```

Field	Type	Null	Key	Default	Extra
codigo_cliente	int(11)	NO	PRI	NULL	
nombre_cliente	varchar(50)	NO		NULL	
nombre_contacto	varchar(30)	YES		NULL	
apellido_contacto	varchar(30)	YES		NULL	
telefono	varchar(15)	NO		NULL	
fax	varchar(15)	NO		NULL	
linea_direccion1	varchar(50)	NO		NULL	
linea_direccion2	varchar(50)	YES		NULL	
ciudad	varchar(50)	NO		NULL	
region	varchar(50)	YES		NULL	
pais	varchar(50)	YES		NULL	
codigo_postal	varchar(10)	YES		NULL	
codigo_empleado_rep_ventas	int(11)	YES	MUL	NULL	
limite_credito	decimal(15,2)	YES		NULL	

4. Eliminar índices

La sintaxis para eliminar índices es la siguiente:

```
DROP INDEX nombre_indice ON nombre_tabla;
```

Ejemplo: Eliminar el índice con el nombre idx_nombre de la tabla cliente

```
DROP INDEX idx_nombre ON cliente;
```

- También es posible eliminar índices en la sentencia ALTER TABLE.

```
ALTER TABLE nombre_tabla
DROP INDEX nombre_indice;
```

El siguiente ejemplo elimina un índice con el nombre idx_nombre de la tabla cliente.

```
ALTER TABLE cliente
DROP INDEX idx_nombre;
```

5. Ejemplo de optimización de consultas

Para obtener información de cómo se está realizando una consulta ejecutaremos la consulta con la instrucción EXPLAIN, de esta forma sabremos la cantidad de filas que tiene que examinar.

Supongamos que queremos optimizar la siguiente consulta:

```
SELECT nombre_contacto, telefono
FROM cliente
WHERE pais = 'France';
```

Lo primero que tenemos que hacer es hacer uso de `EXPLAIN` para obtener información sobre cómo se está realizando la consulta.

```
EXPLAIN SELECT nombre_contacto, telefono
FROM cliente
WHERE pais = 'France';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	SIMPLE	cliente	NULL	ALL	NULL	NULL	NULL	NULL	36

Tenemos que fijarnos en los valores que nos aparecen en las columnas `type` y `rows`. En este caso tenemos `type = ALL`, que quiere decir que es necesario realizar un escaneo completo de todas las filas de la tabla. Y `rows = 36`, quiere decir que en este caso ha tenido que examinar 36 filas.

Si utilizamos SHOW INDEX o DESCRIBE para saber los índices que tiene la tabla veremos que sobre el campo *país* no existe ningún índice. Creamos un índice sobre la columna *país*.

```
CREATE INDEX idx_pais ON cliente(pais);
```

Si volvemos a utilizar DESCRIBE o SHOW INDEX veremos que ahora existe un índice en el campo *país*.

Una vez que hemos comprobado que el índice se ha creado de forma correcta podemos volver a ejecutar la consulta con `EXPLAIN` para comprobar si hemos conseguido optimizarla.

```
EXPLAIN SELECT nombre_contacto, telefono
FROM cliente
WHERE pais = 'France';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	fil
1	SIMPLE	cliente	NULL	ref	idx_pais	idx_pais	203	const	2	1

De nuevo tenemos que fijarnos en los valores que nos aparecen en las columnas `type` y `rows`. En este caso ambos valores han cambiado, ahora `type` es igual a `ref`, y por lo tanto ya no es necesario realizar un escaneo completo de todas las filas de la tabla. Y el valor de `rows` es igual a `2`, que quiere decir que en este caso ha tenido que examinar solamente 2 filas. Por lo tanto hemos optimizado la consulta.