INDICES EN MYSQL (II)

INDICE

1. ¿Qué es un índice?

Un índice es un grupo de datos que MySQL asocia con una o varias columnas de la tabla. En este grupo de datos aparece la relación entre el contenido y el número de fila donde está ubicado.

Los índices sirven para optimizar las consultas y las búsquedas de datos en las tablas, evitando que MySQL tenga que revisar todos los datos disponibles para devolver el resultado. Mediante su uso es mucho más rápido localizar filas con determinados valores de columnas, o seguir un determinado orden. La alternativa es hacer búsquedas secuenciales, que en tablas grandes requieren mucho tiempo.

2. Donde crear índices

Podemos crear índices para un campo de la tabla o para varios. Algunas reglas para la creación de índices son:

- Se deben crear índices sobre aquellas columnas que vayan a ser usadas en una cláusula WHERE.
- Son mejores candidatas a indexar aquellas columnas que presentan muchos valores distintos, mientras que no son buenas candidatas las que tienen muchos valores idénticos, como por ejemplo sexo (masculino y femenino) porque cada consulta implicará siempre recorrer prácticamente la mitad del índice.
- Si necesitamos un select del tipo SELECT ... WHERE columna_1 = X AND columna_2 = Y y ya tenemos un índice con la columna_1, podemos crear un segundo índice con la columna 2 o, mejor todavía, crear un único índice combinado con las columnas 1 y 2. Estos son los índices multicolumna, o compuestos.

  No obstante los índices multicolumna en las claúsulas WHERE deben incluir siempre de izquierda a derecha las columnas indexadas o el índice no se usará. Por ejemplo, supongamos que tenemos un índice USUARIO (ID, NOMBRE, DIRECCION), y una cláusula SELECT ... WHERE NOMBRE = x. Este select no aprovechará el índice.

  Tampoco lo haría un SELECT ... WHERE ID =x AND DIRECCION = y. Cualquier consulta que incluya una columna parte del índice sin incluir además las columnas a su izquierda, no usará el índice. Por tanto en nuestro ejemplo solo sacarían provecho del indice las consultas SELECT ... WHERE ID = x, o WHERE ID = X AND NOMBRE = y o WHERE ID = x AND NOMBRE = y AND DIRECCION = Z.

La creación de índices también tiene desventajas que es necesario conocer para poder mejorar el rendimiento de nuestra base de datos:

- Los índices se actualizan cada vez que se modifica la columna o columnas que utiliza. Por ello no es aconsejable usar como índices columnas en las que serán frecuentes operaciones de escritura (INSERT, UPDATE, DELETE).
- Tampoco tendría sentido crear índices sobre columnas cuando cualquier select sobre ellos va a devolver una gran cantidad de resultados; por ejemplo una columna booleana que admita los valores Y/N.

- Tampoco es necesario usar índices en tablas demasiado pequeñas, ya que en estos casos no hay ganancia de rapidez frente a una consulta normal.

3. Tipos de índices

Tenemos cuatro tipos de índices. El primero corresponde a las claves primarias, que como vimos, también se pueden crear en la parte de definición de columnas.

3.1 Claves primarias

La sintaxis para definir claves primarias es:

definición_columnas | PRIMARY KEY (index_nombre_col,...)

Un ejemplo, usando esta sintaxis, quedaría así:

```
CREATE TABLE ciudad4 (
        nombre CHAR(20) NOT NULL,
        poblacion INT NULL DEFAULT 5000,
        PRIMARY KEY (nombre)
);
```

Pero esta forma tiene más opciones, por ejemplo, entre los paréntesis podemos especificar varios nombres de columnas, para construir claves primarias compuestas por varias columnas:

```
CREATE TABLE mitabla1 (
        id1 CHAR(2) NOT NULL,
        id2 CHAR(2) NOT NULL,
        texto CHAR(30),
        PRIMARY KEY (id1, id2));
```

3.2 Índices

El segundo tipo de índice permite definir índices sobre una columna, sobre varias, o sobre partes de columnas. Para definir estos índices se usan indistintamente las opciones KEY o INDEX.

```
CREATE TABLE mitabla2 (
        id INT,
        nombre CHAR(19),
        INDEX (nombre));
```

O su equivalente:

```
CREATE TABLE mitabla3 (
        id INT,
        nombre CHAR(19),
        KEY (nombre));
```

También podemos crear un índice sobre parte de una columna:

```
CREATE TABLE mitabla4 (
        id INT,
        nombre CHAR(19),
        INDEX (nombre(4)));
```

Este ejemplo usará sólo los cuatro primeros caracteres de la columna 'nombre' para crear el índice.

Se pueden añadir índices a una tabla después de creada con la sentencia ALTER TABLE:

```
ALTER TABLE nombre_tabla ADD INDEX nombre_indice (columna_indexada);
```

Para eliminar un índice usamos también ALTER TABLE:

```
ALTER TABLE nombre_tabla DROP INDEX nombre_indice.
```

3.3 Índices únicos

El tercero permite definir índices con claves únicas, también sobre una columna, sobre varias o sobre partes de columnas. Para definir índices con claves únicas se usa la opción UNIQUE. La diferencia entre un índice único y uno normal es que en los únicos no se permite la inserción de filas con claves repetidas. La excepción es el valor NULL, que sí se puede repetir.

```
CREATE TABLE mitabla5 (
        id INT,
        nombre CHAR(19),
        UNIQUE (nombre));
```

Una clave primaria equivale a un índice de clave única, en la que el valor de la clave no puede tomar valores NULL. Tanto los índices normales como los de claves únicas sí pueden tomar valores NULL. Por lo tanto, las definiciones siguientes son equivalentes:

```
CREATE TABLE mitabla6 (
        id INT,
        nombre CHAR(19) NOT NULL,
        UNIQUE (nombre));
```

Y

```
CREATE TABLE mitabla7 (
        id INT,nombre CHAR(19),
        PRIMARY KEY (nombre));
```

3.4 Índices fulltext

Se usan en tablas del tipo MyISAM, y pueden contener uno o más campos del tipo CHAR, VARCHAR y TEXT.

Un índice de texto completo está diseñado para facilitar y optimizar la búsqueda de palabras clave en tablas que tienen grandes cantidades de información en campos de texto.

Las búsquedas de texto completo son ejecutadas con la función MATCH( ). Esta función ejecuta la búsqueda de una cadena en una colección de texto (un conjunto de una o más columnas incluídas en un índice FULLTEXT). La cadena que se busca es dada como un argumento en la función AGAINST( ), y es ejecutada en modo no sensitivo, es decir, no importa el uso de mayúsculas y minúsculas.

Veamos algunos ejemplos de la utilización de índices Full-Text:

```
CREATE TABLE articulos(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        titulo VARCHAR(200),
        contenido TEXT,
        FULLTEXT indice_tc(titulo,contenido) ) engine=myisam;
```

```
INSERT INTO articulos VALUES (0,'MySQL con Java en MSWindows', 'En este artículo se explica como combinar las ...');

INSERT INTO articulos VALUES (0,'Manejo de datos BLOB con PHP y MySQL', 'Los detalles del almacenamiento y recuperación de ...');

INSERT INTO articulos VALUES (0,'Tutorial básico de MySQL', 'Se explica el uso del programa cliente mysql ...');

INSERT INTO articulos VALUES (0,'MySQL con Java en Linux', 'Conozca como utilizar estas dos herramientas ...');

INSERT INTO articulos VALUES (0,'Manejo de campos BLOB con MySQL y Visual Basic', 'En este artículo se explican los detalles del manejo ...');

INSERT INTO articulos VALUES (0,'MySQL bajoMSWindows', 'A continuación se explica el procedimiento de ...');

SELECT id, titulo, contenido FROM artículos WHERE MATCH (titulo,contenido) AGAINST ('Java');
```

Devuelve las filas que contengan „Java‟ en las columnas del índice FullText (2 filas).

SELECT id, titulo, contenido FROM artículos WHERE MATCH (titulo,contenido) AGAINST ('MySQL');

Igual que la anterior, pero no devuelve nada. La razón es que las palabras que aparecen en más de un 50% de los casos se consideran palabras que "hacen ruido" y no se tienen en cuenta.

SELECT id, titulo, contenido FROM artículos WHERE MATCH (titulo,contenido) AGAINST ('Java Visual');

Se puede incluir en la búsqueda más de una palabra. En este caso buscamos filas que contengan las palabras Java o Visual. Al invertir el orden de las palabras de búsqueda se obtiene el mismo resultado.

SELECT id, titulo, contenido FROM artículos WHERE MATCH (titulo,contenido) AGAINST ('PHP');

Buscamos PHP, pero no se devuelve nada, ya que las palabras de menos de tres caracteres se excluyen de los índices.

Normalmente, como mencionamos antes, los índices se crean al crear la tabla con la que están relacionados, pero también pueden ser creados de forma independiente con:

CREATE INDEX: CREATE [UNIQUE|FULLTEXT] INDEX index_name ON tbl_name (index_col_name,...)