Índices en bases de datos

Ing. Federico Ribeiro

1. ¿Qué es un índice?

Un índice es un puntero a una fila de una determinada tabla de nuestra base de datos. Pero… ¿qué significa exactamente un puntero? Pues bien, un puntero no es más que una referencia que asocia el valor de una determinada columna (o el conjunto de valores de una serie de columnas) con las filas que contienen ese valor (o valores) en las columnas que componen el puntero.

Los índices mejoran el tiempo de recuperación de los datos en las consultas realizadas contra nuestra base de datos. Pero los índices no son todo ventajas, la creación de índices implica un aumento en el tiempo de ejecución sobre aquellas consultas de inserción, actualización y eliminación realizadas sobre los datos afectados por el índice (ya que tendrán que actualizarlo). Del mismo modo, los índices necesitan un espacio para almacenarse, por lo que también tienen un coste adicional en forma de espacio en disco.

La construcción de los índices es el primer paso para realizar optimizaciones en las consultas realizadas contra nuestra base de datos. Por ello, es importante conocer bien su funcionamiento y los efectos colaterales que pueden producir.

2. ¿Para qué los usa MySQL?

MySQL emplea los índices para encontrar las filas que contienen los valores específicos de las columnas empleadas en la consulta de una forma más rápida. Si no existiesen índices, MySQL empezaría buscando por la primera fila de la tabla hasta la última buscando aquellas filas que cumplen los valores establecidos para las columnas empleadas en la consulta. Esto implica que, cuanto más filas tenga la tabla, más tiempo tardará en realizar la consulta. En cambio, si la tabla contiene índices en las columnas empleadas en la consulta, MySQL tendría una referencia directa hacia los datos sin necesidad de recorrer secuencialmente todos ellos.

En general, MySQL emplea los índices para las siguientes acciones:

- Encontrar las filas que cumplen la condición WHERE de la consulta cuyas columnas están indexadas.
- Para recuperar las filas de otras tablas cuando se emplean operaciones de tipo JOIN. Para ello, es importante que los índices sean del mismo tipo y tamaño ya que aumentará la eficiencia de la búsqueda. Por ejemplo: una operación de tipo JOIN sobre dos columnas que tengan un índice del tipo INT(10).
- Disminuir el tiempo de ejecución de las consultas con ordenación (ORDER BY) o agrupamiento (GROUP BY) si todas las columnas presentes en los criterios forman parte de un índice.

- Si la consulta emplea una condición simple cuya columna de la condición está indexada, las filas serán recuperadas directamente a partir del índice, sin pasar a consultar la tabla.

3. ¿Qué tipos de índices hay?

A continuación, vamos a analizar los distintos tipos de índices que se pueden crear y las condiciones que deben cumplir cada uno de ellos:

- INDEX (NON-UNIQUE): este tipo de índice se refiere a un índice normal, no único. Esto implica que admite valores duplicados para la columna (o columnas) que componen 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.
- UNIQUE: este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único. Esto implica que no admite valores duplicados para la columna (o columnas) que componen el índice. Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.
- PRIMARY: este tipo de índice se refiere a un índice en el que todas las columnas deben tener un valor único (al igual que en el caso del índice UNIQUE) pero con la limitación de que sólo puede existir un índice PRIMARY en cada una de las tablas. Aplica la restricción de que los datos de la columna (o columnas) deben tener un valor único.

- 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.
- 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.

Es importante destacar que todos estos índices pueden construirse empleando una o más columnas. Del mismo modo, el orden de las columnas que se especifique al construir el orden es relevante para todos los índices menos para el FULLTEXT (ya que este índice mira en TODAS las columnas que componen el índice).

Para crear un índice, se empleará la siguiente estructura:

- «CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name ON table_name (index_col_name…) index_type;»
- Donde:
    - index_name: es el nombre del índice.

    - table_name: es el nombre de la tabla donde se va a crear el índice.
    - index_col_name: nombre de la columna (o columnas) que formarán el índice.
    - index_type: es el tipo del índice. Se emplea con USING [BTREE | HASH].

Un ejemplo sería:

```
Shell
1  CREATE UNIQUE INDEX mi_indice_unico ON mi_tabla (mi_columna) USING HASH;
```

4. ¿En qué estructuras se almacenan los índices?

Una vez hemos visto los tipos de índices, vamos a ver los distintos tipos de estructuras que se pueden crear para almacenar los índices junto con las características de cada uno de ellas:

- B-TREE: este tipo de índice se usa para comparaciones del tipo =, >, <, >=, <=, BETWEEN y LIKE (siempre y cuando se utilice sobre constantes que no empiecen por %). Para realizar búsquedas empleando este tipo de índice, se empleará cualquier columna (o conjunto de columnas) que forman el prefijo del índice. Por ejemplo: si un índice está formado por las columnas [A, B, C], se podrán realizar búsquedas sobre: [A], [A, B] o [A, B, C].
- HASH: este tipo de índice sólo se usa para comparaciones del tipo = o <=>. Para este tipo de operaciones son muy rápidos en

comparación a otro tipo de estructura. Para realizar búsquedas empleando este tipo de índice, se emplearán todas las columnas que componen el índice.

Un índice puede ser almacenado en cualquier tipo de estructura pero, en función del uso que se le vaya a dar, puede interesar crear el índice en un tipo determinado de estructura o en otro. Por norma general, un índice siempre se creará con la estructura de B-TREE, ya que es la estructura más empleada por la mayoría de operaciones.

5. ¿Por qué es bueno utilizar índices?

Como hemos visto en los apartados anteriores, los índices permiten optimizar las consultas sobre los elementos de la base de datos. Esto desemboca en una reducción considerable del tiempo de ejecución de la consulta. Esta reducción de tiempo es visible sobre tablas de gran tamaño. Sobre tablas pequeñas, el uso de índices no aporta una disminución drástica del tiempo de ejecución de la consulta ya que, prácticamente, MySQL tarda lo mismo en acceder al índice que en acceder de forma secuencial al contenido de la tabla para buscar la fila deseada.

6. Entonces… ¿por qué no crear índices para todas las columnas?

No todas las soluciones son perfectas y tampoco lo iba a ser la creación de índices. La creación de índices también tiene efectos negativos. Estos efectos

negativos es bueno conocerlos ya que pueden ocasionar efectos colaterales no deseados.

Uno de ellos es que las operaciones de inserción, actualización y eliminación que se realicen sobre tablas que tengan algún tipo de índice (o índices), verán aumentado su tiempo de ejecución. Esto es debido a que, después de cada una de estas operaciones, es necesario actualizar el índice (o los índices) presentes en la tabla sobre la que se ha realizado alguna de las operaciones anteriores.

Otro efecto negativo es que los índices deben ser almacenados en algún lugar. Para ello, se emplea espacio de disco. Por ello, el uso de índices aumenta el tamaño de la base de datos en mayor o menor medida.