UNIDAD III - ÍNDICES

Introducción

- Los índices mejoran la velocidad de recuperación de datos en consultas.
- Son esenciales cuando se manejan grandes volúmenes de datos.
- Funcionan como el índice de un libro, permitiendo encontrar información rápidamente.

Concepto Básico

- Un índice es una estructura de base de datos creada sobre una o más columnas de una tabla.
- Su objetivo es acelerar la búsqueda de datos.
- Sin un índice, MySQL revisa fila por fila para encontrar los datos.

Ventajas y Desventajas

- **Ventajas:**
  a. Mejoran la velocidad de consulta.
  b. Optimizan el ordenamiento de resultados.
  c. Hacen más eficientes las búsquedas.

- **Desventajas:**
  a. Ocupan espacio adicional.
  b. Afectan el rendimiento en inserciones, actualizaciones y eliminaciones.

Tipo de Índices

- **Índice Clustered (Agrupado):**
  - Determina el orden físico de los datos en la tabla.
  - Solo puede haber uno por tabla.
  - Ejemplo:

```
CREATE CLUSTERED INDEX
IX_Clientes_Apellido ON Clientes (Apellido);
```

- **Índice Non-Clustered (No Agrupado):**
  - No altera el orden físico de los datos.
  - Puede haber varios por tabla.
  - Ejemplo:

```
CREATE INDEX IX_Clientes_Email ON
Clientes (Email);
```

Índices Compuestos

- Creado sobre más de una columna.
- Útil cuando las consultas filtran datos en varias columnas.
- Ejemplo:

  CREATE INDEX IX_Clientes_Apellido_Nombre ON Clientes (Apellido, Nombre);

Mantenimiento

- Los índices pueden ralentizar las operaciones de modificación de datos.
- Se recomienda realizar mantenimiento periódico.
- Ejemplo de optimización:

```
OPTIMIZE TABLE Clientes;
```

Buenas Prácticas

- No crear índices en todas las columnas.
- Evaluar la frecuencia de uso de las columnas antes de indexarlas.
- Realizar mantenimiento regular de los índices.
- Usar índices compuestos en columnas consultadas frecuentemente juntas.

Modificar y Eliminar Índices

- **Modificar un índice:**

  Se debe eliminar y volver a crear.

  Ejemplo:
  DROP INDEX IX_Clientes_Apellido ON Clientes;
  CREATE INDEX IX_Clientes_Nombre_Apellido ON Clientes (Nombre, Apellido);

- **Eliminar un índice:**

  Ejemplo:

  DROP INDEX IX_Clientes_Email ON Clientes;

Explain

- **EXPLAIN** permite ver cómo MySQL ejecuta una consulta y si usa un índice.
- Ayuda a optimizar consultas y mejorar el rendimiento.
- Ejemplo de uso:
  EXPLAIN SELECT * FROM Clientes WHERE Apellido = 'González';
- Columnas clave en el resultado:
  - **type**: Muestra el tipo de acceso (idealmente ref o index).
  - **possible_keys**: Índices que podrían usarse.
  - **key**: Índice realmente utilizado.
  - **rows**: Número de filas examinadas.

Índices B-Tree

Estructura de B-Tree

- Se organiza en una estructura jerárquica de nodos (raíz, internos y hojas).
- Cada nodo contiene claves ordenadas y punteros a sus nodos hijos.
- Las hojas contienen referencias a las filas de datos.

Búsqueda en Índices B-Tree

1. **Búsqueda Binaria en los Nodos:** MySQL compara la clave de búsqueda con los valores en el nodo raíz.

2. **Descenso por el Árbol:** Se sigue el puntero hacia el nodo hijo que contiene el rango correcto de valores.

3. **Localización en el Nodo Hoja:** Si el índice es **clustered** (como en InnoDB), los nodos hoja contienen los registros reales. Si es **non-clustered**, los nodos hoja contienen punteros a las filas en la tabla.

4. **Acceso a Datos:** debe acceder a la tabla para obtener el resto de los datos de la tabla (**table lookup**).

Índices Full-Text

Estructura de Full-Text

- Se basa en una estructura de **inverted index**.
- Divide el texto en **tokens** (palabras clave).
- Almacena una lista de documentos donde aparece cada palabra.

Búsqueda en Índices Full-Text

- **Tokenización:** Se descompone el texto en palabras individuales.
- **Eliminación de Palabras Comunes (Stop Words):** Se filtran palabras muy frecuentes como "el", "de", "y".
- **Normalización:** Se convierten las palabras a minúsculas y se eliminan caracteres especiales.
- **Indexación:** Se almacena un mapeo de palabras clave a documentos que las contienen.
- **Búsqueda:** Cuando se ejecuta una consulta, MySQL compara los términos de búsqueda con el índice invertido y devuelve los documentos relevantes.

Ejemplo de Búsqueda en Índices Full-Text

```
SELECT * FROM articulos WHERE MATCH(titulo, contenido) AGAINST ('base de datos' IN NATURAL LANGUAGE MODE);
```

B-Tree vs. Full-Text

Característica	B-Tree	Full-Text
Activación Automática	Sí	No
Estructura	Árbol balanceado, optimizado para búsquedas exactas y rangos (=, <, >, BETWEEN).	Índice invertido, optimizado para búsquedas de palabras clave en texto largo.
Tipos de Búsqueda	Comparación de valores exactos, rangos, ordenamiento rápido (ORDER BY).	Búsqueda semántica, ranking por relevancia.
Columnas Soportadas	Casi cualquier tipo (INT, VARCHAR, DATE, etc.).	Solo VARCHAR, TEXT, y sus variantes.