13. Optimización de consultas

La **optimización de consultas SQL** se refiere, del orden en el que se ejecutan los comandos que la forman es decir cómo mejorar las consultas a base de datos y para ello es muy importante comprender el orden de ejecución de cada uno de los comandos que la integran

1. Evitar la condiciones IN (SELECT…) sustituyéndolas por joins: cuando se utiliza un conjunto de valores en la clausula IN, se traduce por una condición compuesta con el operador OR. Esto es lento, ya que por cada fila debe comprobar cada una de las condiciones simples. Suele ser mucho más rápido mantener una tabla con los valores que están dentro del IN, y hacer un join normal. Por ejemplo, esta consulta:

```
SELECT * FROM datos WHERE campo IN ('a', 'b', 'c', 'd', ... , 'x',
'y', 'z');
```

se puede sustituir por la siguiente consulta, siempre que la tabla "letras" contenga una fila por cada valor contenido en el conjunto del IN:

```
SELECT * FROM datos d, letras l WHERE d.campo = l.letra;
```

14. Optimización de consultas

2. También hay que tener cuidado cuando se mete un SELECT dentro del IN, ya que esa consulta puede retornar muchas filas, y se estaría cayendo en el mismo error. Normalmente, una condición del tipo "WHERE campo IN (SELECT...)" se puede sustituir por una consulta con //join//.

3. Cuando se hace una consulta multi-tabla con //joins//, el orden en que se ponen las tablas en el FROM influye en el plan de ejecución. Aquellas tablas que retornan más filas deben ir en las primeras posiciones, mientras que las tablas con pocas filas deben situarse al final de la lista de tablas.

4. Si en la cláusula WHERE se utilizan campos indexados como argumentos de funciones, el índice quedará desactivado. Es decir, si tenemos un índice por un campo IMPORTE, y utilizamos una condición como WHERE ROUND(IMPORTE) > 0, entonces el índice quedará desactivado y no se utilizará para la consulta

15. Optimización de consultas

5. Una condición negada con el operador NOT desactiva los índices.

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition
```

6. Una consulta cualificada con la cláusula DISTINCT debe ser ordenada por el servidor aunque no se incluya la cláusula ORDER BY.

7. Para comprobar si existen registros para cierta condición, no se debe hacer un SELECT COUNT(*) FROM X WHERE xxx, sino que se hace un SELECT DISTINCT 1 FROM X WHERE xxx. De este modo evitamos al servidor que cuente los registros.

8) Si vamos a realizar una operación de inserción, borrado o actualización masiva, es conveniente desactivar los índices, ya que por cada operación individual se actualizarán los datos de cada uno de los índices. Una vez terminada la operación, volvemos a activar los índices para que se regeneren.

16. Optimización de consultas

9. La mejor optimización es rediseñar y normalizar la base de datos. Las bases de datos relacionales están diseñadas para funcionar lo más rápidamente posible para un buen diseño relacional, pero con diseños erróneos, se vuelven muy lentas. La mayoría de los problemas de rendimiento tienen un problema de fondo de mal diseño, y muchos de ellos no podrán ser optimizados si no se rediseña el esquema de base de datos.

10. Toda consulta SELECT se ejecuta dentro del servidor en varios pasos. Para la misma consulta, pueden existir distintas formas para conseguir el mismo resultados, por lo que el servidor es el responsable de decidir qué camino seguir para conseguir el mejor tiempo de respuesta. La parte de la base de datos que se encarga de estas decisiones se llama **Optimizador.** El camino seguido por el servidor para la ejecución de una consulta se denomina “Plan de ejecución”