Informe de Optimización de Consultas – Análisis de Patrones de Sueño.

Integrantes:

Cobos Valle Joshua Alexander
Forero Villota Katherine Sheila
Intriago Celi Romina Anahí
Olalla Sacancela Miguel Sebastián
Ponce Pincay Jhon Lenin

Fecha:

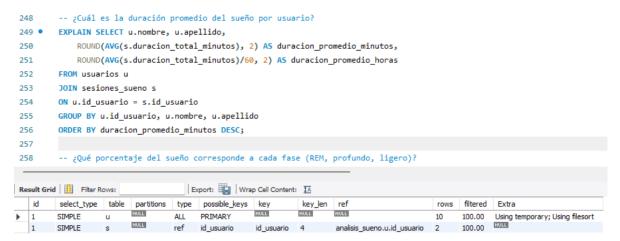
20 de septiembre de 2025

1. Introducción

Este informe analiza y optimiza un conjunto de consultas SQL diseñadas para responder preguntas sobre patrones de sueño. Se utiliza la sentencia EXPLAIN para evaluar los planes de ejecución generados por MySQL, identificando posibles cuellos de botella y proponiendo mejoras mediante índices, reescritura de consultas y normalización adecuada.

2. Consultas Analizadas

2.1. Duración promedio del sueño por usuario



- La tabla usuarios tiene un type ALL, lo que significa que se hace un recorrido completo de la tabla. Como la tabla tiene solo 10 usuarios, esto no representa un problema de rendimiento.
- La tabla sesiones_sueno utiliza un type ref sobre el índice id_usuario. Esto indica que la búsqueda se hace de forma eficiente usando el índice de la clave foránea que apunta a usuarios.
- Se observa que se utiliza una tabla temporal y un filesort para el GROUP BY y el ORDER BY. Esto es normal porque se están agregando los datos y ordenando.
- Optimización: Para los datos actuales no hace falta ningún índice adicional, ya que los JOINs utilizan los índices existentes.

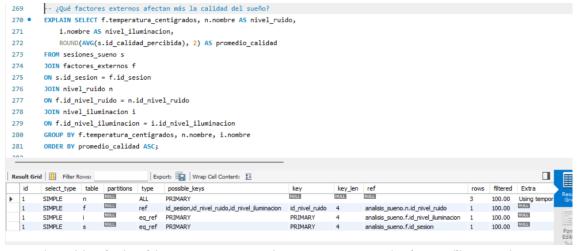
2.2. Porcentaje de sueño en cada fase

```
-- ¿Qué porcentaje del sueño corresponde a cada fase (REM, profundo, ligero)?
259 •
        EXPLAIN SELECT tipo fase,
260
            ROUND(AVG(porcentaje_por_sesion), 2) AS porcentaje_promedio
261

⊖ FROM (
262
            SELECT f.id sesion, f.tipo fase, f.duracion minutos * 100.0 / SUM(f.duracion minutos)
263
            OVER (PARTITION BY f.id sesion) AS porcentaje por sesion
264
            FROM fases sueno f
       ) t
265
        GROUP BY tipo fase
266
        ORDER BY porcentaje promedio DESC;
267
Result Grid Filter Rows:
                                     Export: Wrap Cell Content: IA
                              partitions type possible_keys key
                                                                                    filtered
   id
        select_type table
                                                                key_len
                                                                       ref
                                                                              rows
                                                                                            Extra
                                                               NULL
                              NULL
                                             NULL
                                                         NULL
                                                                       NULL
        PRIMARY
                   <derived2>
                                                                                    100.00
                                                                                           Using temporary; Using filesort
                              NULL
                                            NULL
                                                       NULL NULL
                                                                       NULL 60
       DERIVED f
                                                                                    100.00 Using filesort
  2
```

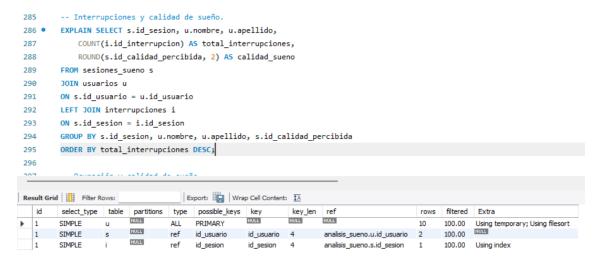
- En esta consulta, la tabla fases_sueno tiene un type ALL (gracias a la función ventana, no es optimizable como tal), lo que significa que se hace un recorrido completo de las 60 filas. Esto es aceptable para la cantidad de datos actuales.
- Se observa el uso de filesort para el GROUP BY, lo que indica que MySQL necesita ordenar temporalmente los datos.
- Optimización: Para tablas grandes, conviene tener un índice sobre id_sesion en la tabla fases_sueno, pero al ser FK ya posee uno, así que no hay cambios.

2.3. Factores externos que afectan la calidad



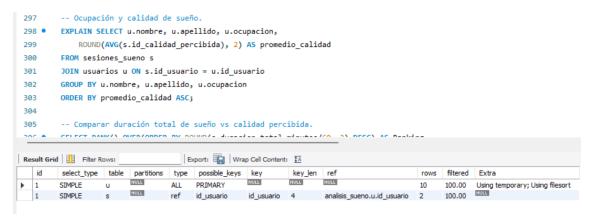
- La tabla nivel_ruido se recorre completamente, pero solo tiene 3 filas, por lo que es trivial.
- La tabla nivel_iluminacion utiliza un eq_ref, lo que significa que se busca una fila exacta mediante la clave primaria.
- La tabla factores_externos usa un índice por id_nivel_ruido para hacer la búsqueda de forma eficiente, mientras que sesiones_sueno busca por id_sesion usando la clave primaria.
- Se usan tablas temporales y filesort por el GROUP BY y el ORDER BY, lo que es normal.
- Optimización: Ya está optimizado puesto que no hay ningún filtro masivo ni alguna condición para usar un índice.

2.4. Interrupciones y calidad



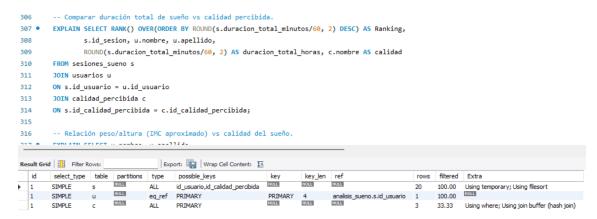
- La tabla usuarios hace un recorrido completo de 10 filas, rápido debido a su tamaño (Necesario por las funciones de agregación).
- La tabla sesiones_sueno utiliza un índice por id_usuario para el JOIN, lo que es eficiente.
- La tabla interrupciones usa un índice sobre id_sesion para la búsqueda y el uso de Using index indica que la consulta solo necesita leer el índice, lo cual es rápido.
- Se genera una tabla temporal y filesort para el GROUP BY y ORDER BY.
- Optimización: La consulta ya está bien optimizada. Ningún índice adicional es necesario.

2.5. Ocupación y calidad



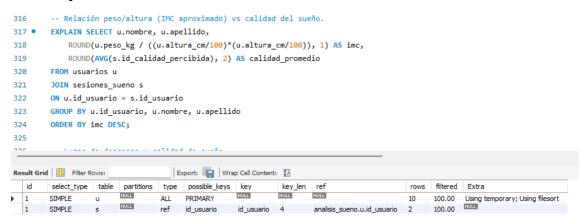
- La tabla usuarios se recorre completamente (Por la función de agregación), mientras que la tabla sesiones_sueno utiliza un índice por id_usuario.
- Se utiliza una tabla temporal y filesort por el GROUP BY y ORDER BY.
- Optimización: No hay ningún filtro como tal, y ya que todos los FK y PK tienen índices, ya está todo bien optimizado. No hay cambios.

2.6. Comparación duración vs calidad



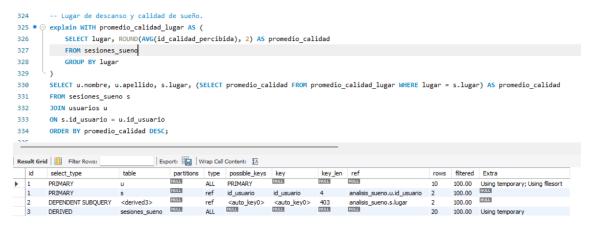
- La tabla calidad_percibida se recorre completamente, pero solo tiene 3 filas, por lo que no afecta el rendimiento.
- La tabla sesiones_sueno usa el índice de la clave foránea id_calidad_percibida, lo que permite un acceso rápido.
- La tabla usuarios utiliza la clave primaria para buscar el usuario correspondiente.
- Se genera temporal y filesort para el ordenamiento del ranking, lo que es normal.
- Optimización: Si observamos bien, hay un filtered al 33.33% en la tabla calidad_percibida, por lo que supone que solo un tercio de las filas coincidirán.
 Sin embargo, es una tabla sumamente pequeña que no tiende a cambiar ni a guardar altos volúmenes de datos, por lo que una optimización no es necesaria.

2.7. IMC y calidad



- La tabla usuarios hace un recorrido completo de 10 filas y la tabla sesiones_sueno usa el índice por id_usuario.
- Se genera tabla temporal y filesort para el GROUP BY y ORDER BY.
- Optimización: Dada la naturaleza de la consulta, es normal que se recorra la tabla usuarios por completo. Por lo tanto, la consulta está bien optimizada. Ningún índice adicional es necesario.

2.8. Lugar y calidad

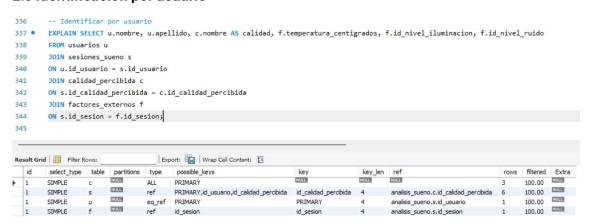


- La tabla usuarios hace un recorrido completo, mientras que sesiones_sueno usa el índice por id_usuario.
- El CTE genera una tabla temporal y la subconsulta dependiente se ejecuta para cada fila, lo que podría volverse lento si hay muchas sesiones.
- Optimización: Sería mejor reemplazar la subconsulta dependiente por un JOIN directo con la CTE. En todo caso, un índice sobre la columna lugar en sesiones_sueno aceleraría la búsqueda en la subquery (WHERE lugar = s.lugar)

Índice sugerido:

CREATE INDEX idx_sesiones_lugar ON sesiones_sueno(lugar);

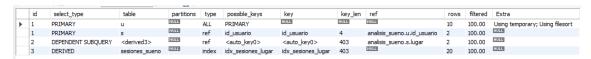
2.9 Identificación por usuario



- La tabla usuarios utiliza la clave primaria, la tabla sesiones_sueno usa el índice por id_calidad_percibida, la tabla calidad_percibida hace full scan de 3 filas y la tabla factores_externos usa el índice por id_sesion.
- Optimización: La consulta está bien optimizada ya. La tabla calidad_percibida como se mencionó es una tabla cuyo propósito no es almacenar grandes volúmenes de datos. Por ello, no es necesario usar un índice con ella.

3. Consultas Optimizadas (según corresponda)

3.1. Lugar y calidad



 Podemos observar que pasó de recorrer toda la tabla sesiones_sueno a un index, optimizando correctamente la consulta planteada.

4. Conclusiones

- Las consultas están correctamente planteadas mayormente.
- La mayor ganancia en rendimiento se obtiene mediante índices en columnas de JOIN y WHERE. Sin embargo, como los FK y PK ya poseen un índice creado automáticamente por MySQL Workbench, solo nos queda por optimizar los filtros aplicados, que no se aplicaron en casi todas las tablas (se usaron más funciones ventana y de agregación). Por ello, solo obtuvimos optimización en una sola table exitosamente.
- Al tener funciones de agregación y de ventana es normal que se tenga que hacer un full scan de ciertas tablas y es algo inevitable hasta cierto punto, por lo que no se puede utilizar índices para la optimización.