

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

```
248 -- ¿Cuál es la duración promedio del sueño por usuario?
249 • EXPLAIN SELECT u.nombre, u.apellido,
250     ROUND(AVG(s.duracion_total_minutos), 2) AS duracion_promedio_minutos,
251     ROUND(AVG(s.duracion_total_minutos)/60, 2) AS duracion_promedio_horas
252 FROM usuarios u
253 JOIN sesiones_sueno s
254 ON u.id_usuario = s.id_usuario
255 GROUP BY u.id_usuario, u.nombre, u.apellido
256 ORDER BY duracion_promedio_minutos DESC;
257
258 -- ¿Qué porcentaje del sueño corresponde a cada fase (REM, profundo, ligero)?
```

Result Grid												
Filter Rows: <input type="text"/>												
Export: Wrap Cell Content: TA												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	u	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
	1	SIMPLE	s	NULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	NULL

- 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

```
258 -- ¿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
265   ) t
266   GROUP BY tipo_fase
267   ORDER BY porcentaje_promedio DESC;
268
```

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	60	100.00	Using temporary; Using filesort
	2	DERIVED	f	NULL	ALL	NULL	NULL	NULL	NULL	60	100.00	Using filesort

- 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

```
269 -- ¿Qué factores externos afectan más la calidad del sueño?
270 • EXPLAIN SELECT f.temperatura_centigrados, n.nombre AS nivel_ruido,
271     i.nombre AS nivel_iluminacion,
272     ROUND(AVG(s.id_calidad_percibida), 2) AS promedio_calidad
273   FROM sesiones_sueno s
274   JOIN factores_externos f
275   ON s.id_sesion = f.id_sesion
276   JOIN nivel_ruido n
277   ON f.id_nivel_ruido = n.id_nivel_ruido
278   JOIN nivel_iluminacion i
279   ON f.id_nivel_iluminacion = i.id_nivel_iluminacion
280   GROUP BY f.temperatura_centigrados, n.nombre, i.nombre
281   ORDER BY promedio_calidad ASC;
282
```

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	n	NULL	ALL	PRIMARY	NULL	NULL	NULL	3	100.00	Using temporary
	1	SIMPLE	f	NULL	ref	id_sesion, id_nivel_ruido, id_nivel_iluminacion	id_nivel_ruido	4	analisis_sueno.n.id_nivel_ruido	1	100.00	Using temporary
	1	SIMPLE	i	NULL	eq_ref	PRIMARY	PRIMARY	4	analisis_sueno.f.id_nivel_iluminacion	1	100.00	Using temporary
	1	SIMPLE	s	NULL	eq_ref	PRIMARY	PRIMARY	4	analisis_sueno.f.id_sesion	1	100.00	Using temporary

- 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


```
285 -- Interrupciones y calidad de sueño.
286 • EXPLAIN SELECT s.id_sesion, u.nombre, u.apellido,
287     COUNT(i.id_interrupcion) AS total_interrupciones,
288     ROUND(s.id_calidad_percibida, 2) AS calidad_sueno
289 FROM sesiones_sueno s
290 JOIN usuarios u
291 ON s.id_usuario = u.id_usuario
292 LEFT JOIN interrupciones i
293 ON s.id_sesion = i.id_sesion
294 GROUP BY s.id_sesion, u.nombre, u.apellido, s.id_calidad_percibida
295 ORDER BY total_interrupciones DESC;
```

Result Grid												
Filter Rows: <input type="text"/> Export: <input type="button" value=""/> Wrap Cell Content: <input type="text"/>												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	u	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
	1	SIMPLE	s	NULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	NULL
	1	SIMPLE	i	NULL	ref	id_sesion	id_sesion	4	analisis_sueno.s.id_sesion	1	100.00	Using index

- 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

```
297 -- Ocupación y calidad de sueño.
298 • EXPLAIN SELECT u.nombre, u.apellido, u.ocupacion,
299     ROUND(AVG(s.id_calidad_percibida), 2) AS promedio_calidad
300 FROM sesiones_sueno s
301 JOIN usuarios u ON s.id_usuario = u.id_usuario
302 GROUP BY u.nombre, u.apellido, u.ocupacion
303 ORDER BY promedio_calidad ASC;
304
305 -- Comparar duración total de sueño vs calidad percibida.
306 • SELECT SUM(s.duracion) AS duracion_total, SUM(s.id_calidad_percibida) AS calidad_promedio
```

Result Grid												
Filter Rows: <input type="text"/>												
Export:  Wrap Cell Content: fA												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	u	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
	1	SIMPLE	s	NULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	NULL

- 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

```
306 -- Comparar duración total de sueño vs calidad percibida.
307 • EXPLAIN SELECT RANK() OVER(ORDER BY ROUND(s.duracion_total_minutos/60, 2) DESC) AS Ranking,
308     s.id_sesion, u.nombre, u.apellido,
309     ROUND(s.duracion_total_minutos/60, 2) AS duracion_total_horas, c.nombre AS calidad
310 FROM sesiones_sueno s
311 JOIN usuarios u
312 ON s.id_usuario = u.id_usuario
313 JOIN calidad_percibida c
314 ON s.id_calidad_percibida = c.id_calidad_percibida;
315
316 -- Relación peso/altura (IMC aproximado) vs calidad del sueño.
```

Result Grid											
Filter Rows:											
Export: Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	HULL	ALL	id_usuario, id_calidad_percibida	HULL	HULL	HULL	20	100.00	Using temporary; Using filesort
1	SIMPLE	u	HULL	eq_ref	PRIMARY	PRIMARY	4	analisis_sueno.s.id_usuario	1	100.00	HULL
1	SIMPLE	c	HULL	ALL	PRIMARY	HULL	HULL	HULL	3	33.33	Using where; Using join buffer (hash join)

- 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

```
316 -- Relación peso/altura (IMC aproximado) vs calidad del sueño.
317 • EXPLAIN SELECT u.nombre, u.apellido,
318     ROUND(u.peso_kg / ((u.altura_cm/100)*(u.altura_cm/100)), 1) AS imc,
319     ROUND(AVG(s.id_calidad_percibida), 2) AS calidad_promedio
320 FROM usuarios u
321 JOIN sesiones_sueno s
322 ON u.id_usuario = s.id_usuario
323 GROUP BY u.id_usuario, u.nombre, u.apellido
324 ORDER BY imc DESC;
325
```

Result Grid											
Filter Rows:											
Export: Wrap Cell Content:											
id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	u	HULL	ALL	PRIMARY	HULL	HULL	HULL	10	100.00	Using temporary; Using filesort
1	SIMPLE	s	HULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	HULL

- 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

```
324 -- Lugar de descanso y calidad de sueño.
325 • explain WITH promedio_calidad_lugar AS (
326     SELECT lugar, ROUND(AVG(id_calidad_percibida), 2) AS promedio_calidad
327     FROM sesiones_sueno
328     GROUP BY lugar
329 )
330 SELECT u.nombre, u.apellido, s.lugar, (SELECT promedio_calidad FROM promedio_calidad_lugar WHERE lugar = s.lugar) AS promedio_calidad
331 FROM sesiones_sueno s
332 JOIN usuarios u
333 ON s.id_usuario = u.id_usuario
334 ORDER BY promedio_calidad DESC;
```

Result Grid												
Filter Rows:												
Export: Wrap Cell Content:												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
	1	PRIMARY	u	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
	1	PRIMARY	s	NULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	NULL
	2	DEPENDENT SUBQUERY	<derived3>	NULL	ref	<auto_key0>	<auto_key0>	403	analisis_sueno.s.lugar	2	100.00	NULL
	3	DERIVED	sesiones_sueno	NULL	ALL	NULL	NULL	NULL	NULL	20	100.00	Using temporary














- 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

```
336 -- Identificar por usuario
337 • EXPLAIN SELECT u.nombre, u.apellido, c.nombre AS calidad, f.temperatura_centigrados, f.id_nivel_iluminacion, f.id_nivel_ruido
338 FROM usuarios u
339 JOIN sesiones_sueno s
340 ON u.id_usuario = s.id_usuario
341 JOIN calidad_percibida c
342 ON s.id_calidad_percibida = c.id_calidad_percibida
343 JOIN factores_externos f
344 ON s.id_sesion = f.id_sesion;
```

Result Grid												
Filter Rows:												
Export:  Wrap Cell Content: 												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	c		ALL	PRIMARY				3	100.00	
	1	SIMPLE	s		ref	PRIMARY,id_usuario,id_calidad_percibida	id_calidad_percibida	4	analisis_sueno.c.id_calidad_percibida	6	100.00	
	1	SIMPLE	u		eq_ref	PRIMARY	PRIMARY	4	analisis_sueno.s.id_usuario	1	100.00	
	1	SIMPLE	f		ref	id_sesion	id_sesion	4	analisis_sueno.s.id_sesion	1	100.00	

- 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

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	PRIMARY	u	NULL	ALL	PRIMARY	NULL	NULL	NULL	10	100.00	Using temporary; Using filesort
	1	PRIMARY	s	NULL	ref	id_usuario	id_usuario	4	analisis_sueno.u.id_usuario	2	100.00	NULL
	2	DEPENDENT SUBQUERY	<derived3>	NULL	ref	<auto_key0>	<auto_key0>	403	analisis_sueno.s.lugar	2	100.00	NULL
	3	DERIVED	sesiones_sueno	NULL	index	idx_sesiones_lugar	idx_sesiones_lugar	403	NULL	20	100.00	NULL

- 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.