

L U M B R E



# Optimización de consultas

**Grado:** 3

**Materia:** SQL y Consultas de Datos · Bases de Datos

11/10/2026

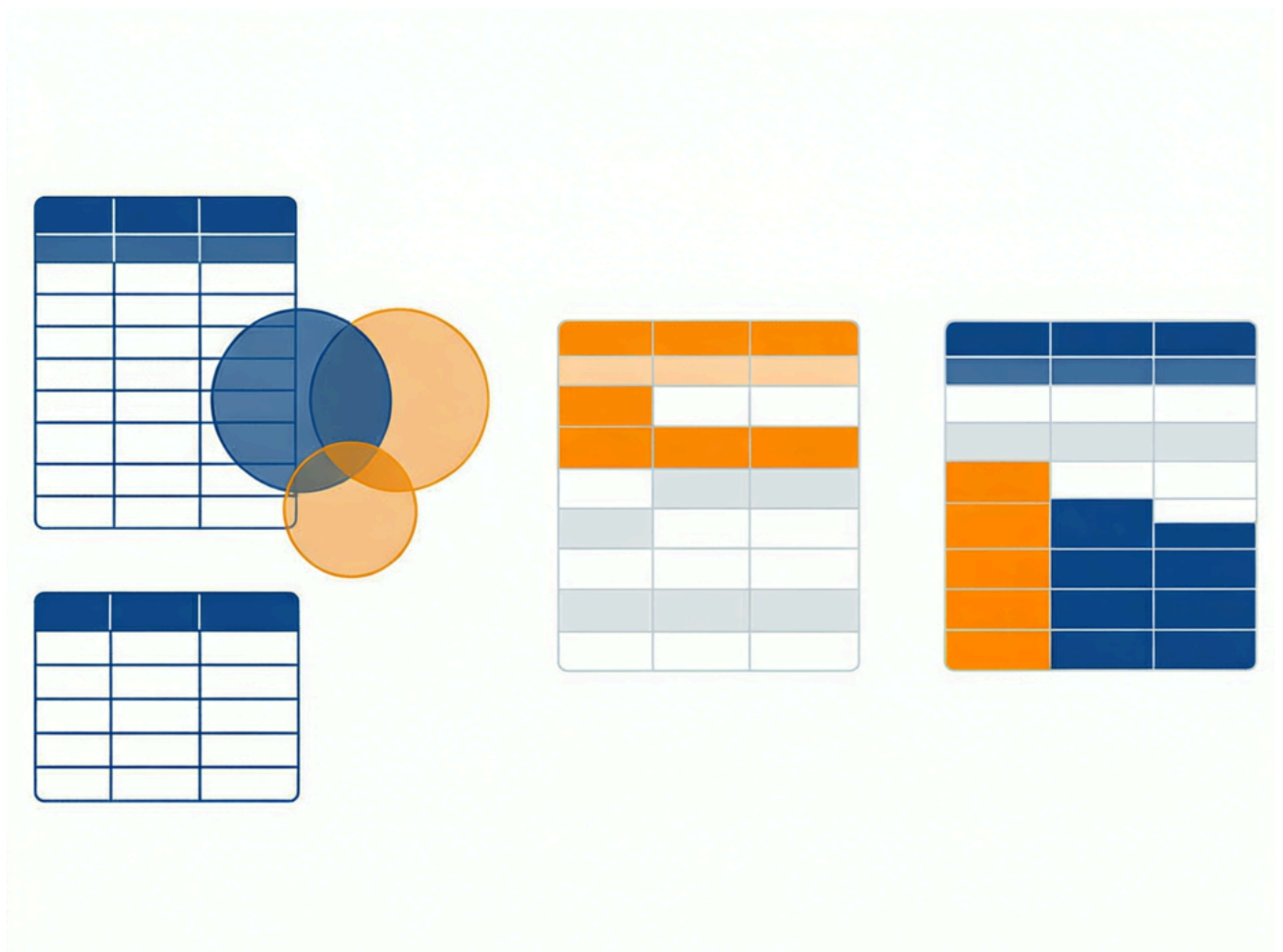
## Antes de empezar

Una consulta correcta puede ser inaceptablemente lenta. Esta lección enseña a leer un plan de ejecución, entender qué hace un índice y reescribir SQL para que el motor trabaje con menos datos y menos esfuerzo.

### QUÉ APRENDERÁS

- Diagnosticar el coste de una consulta con su plan de ejecución y aplicar índices, reescrituras y buenas prácticas para reducir tiempos de respuesta.

## Optimización de consultas



*SQL: consultas, joins, agregacion y optimizacion de consultas.*

## Mapa conceptual

- **SQL**
  - Básico
    - SELECT

- WHERE
- ORDER BY
- JOINS
  - INNER
  - LEFT
  - FULL
- Agregación
  - GROUP BY
  - HAVING
- Avanzado
  - Subconsultas
  - CTEs
  - Ventana
- Rendimiento
  - EXPLAIN
  - Índices

El mapa mental interactivo es interactivo en la versión online.



### Mismo resultado, menos trabajo

El mismo resultado se puede obtener tardando milisegundos o minutos. La diferencia está en cuánto dato toca mirar y en cuántas pasadas. Optimizar es conseguir la misma respuesta leyendo lo mínimo necesario.

## El optimizador

El motor traduce tu SQL declarativo en un plan: una secuencia concreta de operaciones (barridos, búsquedas por índice, tipos de JOIN, ordenaciones). El optimizador elige el plan estimando coste. Tu trabajo es darle opciones buenas y estadísticas frescas para que acierte.

## Leer el plan de ejecución

Con EXPLAIN se ve el plan sin ejecutar y con su versión analizada, el coste real por operador. Se buscan los cuellos: barridos completos de tablas enormes, ordenaciones gigantes o JOIN que multiplican filas. Sin mirar el plan, optimizar es a ciegas.

## Qué es un índice

Un índice es una estructura auxiliar, casi siempre un árbol ordenado, que permite localizar filas por una columna sin recorrer toda la tabla. Acelera búsquedas por igualdad o rango a cambio de espacio y de coste en cada escritura, porque hay que mantenerlo.

## Índice que sirve y índice que no

- Sirve: filtrar u ordenar por las columnas iniciales del índice, en su mismo orden.
- No sirve: aplicar una función a la columna indexada, o filtrar por una columna que no encabeza el índice compuesto.
- Un índice que se usa poco solo paga su coste en cada escritura.

## Selectividad, la clave

Un índice solo compensa si es selectivo: si el filtro deja fuera la mayor parte de las filas. Para un predicado que devuelve la mitad de la tabla, un barrido completo puede ser más rápido que saltar por índice. La utilidad depende de cuántas filas sobreviven.

## Cobertura y evitar el acceso a la tabla

Un índice que incluye todas las columnas que la consulta necesita puede responder sin tocar la tabla base: es un índice cubriente. Ese ahorro de ir a por las filas completas explica por qué `SELECT` asterisco mata la cobertura y ralentiza.

## Reescrituras que ayudan

1. Filtra pronto: pon las condiciones más selectivas para reducir filas lo antes posible.
2. Proyecta solo lo necesario: enumera columnas en lugar de asterisco.
3. Sustituye subconsultas correlacionadas por `JOIN` cuando el plan lo pida.
4. Cambia `DISTINCT` innecesario por un `GROUP BY` más barato o elimínalo.
5. Parte consultas enormes en pasos si el optimizador se pierde.

## El coste de los JOIN

Cada `JOIN` añade filas según su cardinalidad; una unión mal elegida puede explotar el resultado antes de filtrar. Unir por claves indexadas y filtrar antes de unir reduce el trabajo. El orden de las tablas y las condiciones `ON` importan al plan.

## Estadísticas y cardinalidad

El optimizador decide con estadísticas sobre distribución de valores. Si están obsoletas, elige planes malos. Mantener estadísticas actualizadas es tan importante como crear índices: sin información fiable, el motor adivina.

## Paginación cara

Un LIMIT con OFFSET grande obliga a generar y descartar todas las filas previas. Para paginar profundo, se usa paginación por clave (busca donde id mayor que el último visto) en lugar de OFFSET, que es constante en coste.

## El problema de N más uno

Desde el lado de la aplicación, lanzar una consulta por cada fila de un listado (el problema de N más uno) es devastador. Se arregla con un JOIN o una consulta agrupada que traiga todo de golpe. Optimizar no es solo el SQL suelto, es el patrón de acceso.

*Cada operador caro apunta a una causa y a una intervención concreta.*

| Síntoma en el plan        | Causa probable                    | Remedio                              |
|---------------------------|-----------------------------------|--------------------------------------|
| barrido completo de tabla | sin índice útil para el filtro    | índice sobre la columna de WHERE     |
| ordenación costosa        | no hay índice con el orden pedido | índice que cubra el ORDER BY         |
| unión que explota filas   | JOIN por clave no única           | filtrar antes y revisar cardinalidad |

### 1. Una consulta filtra por una columna y devuelve casi toda la tabla, y sigue lenta pese a tener índice en esa columna. ¿Qué explica el poco beneficio?

- ☐ El índice está corrupto
- ☐ La columna es poco selectiva: un barrido completo puede ser mejor
- ☐ Los índices nunca aceleran
- ☐ Falta un JOIN

## Medir antes y después

Toda optimización debe contrastarse: tiempo, filas leídas, coste del plan. Cambiar a ciegas por superstición (poner índices en todo) degrada escrituras y ocupa memoria. El método es hipótesis, intervención y medición.

## Cuándo desnormalizar

A veces la respuesta a una lentitud crónica es sacrificar normalización: duplicar una columna calculada o pre-agrupar en una tabla resumen para no recalcularse cada vez. Es

un intercambio consciente entre integridad y velocidad, típico de almacenes de datos.

## Errores frecuentes

- Poner índices en cada columna «por si acaso», penalizando escrituras.
- Aplicar funciones a columnas indexadas en el WHERE e inhabilitar el índice.
- Optimizar sin mirar el plan de ejecución, guiándose por corazonadas.

### 2. Crear un índice siempre acelera las consultas que filtran por esa columna.

☐ Verdadero

☐ Falso

## Mantenimiento del rendimiento

El rendimiento no se logra una vez: los planes cambian al crecer los datos y al modificarse la distribución. Monitorizar consultas lentas, renovar estadísticas y revisar índices sin uso es una disciplina continua del ingeniero de datos.

ACTIVIDAD INTERACTIVA DISPONIBLE EN  
LUMBRE

### 3. Vocabulario de optimización



### 4. Ordena el protocolo para acelerar una consulta lenta:

**Elementos:**

- toma el tiempo y el plan actuales
- localiza el operador caro y su causa
- crea índice o reescribe la consulta
- repite el plan y confirma la mejora

|                     |       |
|---------------------|-------|
| <b>Medir</b>        | . . . |
| <b>Diagnosticar</b> | . . . |
| <b>Intervenir</b>   | . . . |
| <b>Verificar</b>    | . . . |

Use The Index, Luke <https://use-the-index-luke.com/> Guía práctica de índices y rendimiento SQL.

## VIDEO

**Video complementario**

Recurso audiovisual para reforzar los conceptos.



Escanea para verlo en Lumbre

<https://www.youtube.com/watch?v=HX6q1bsBz1w>

**5. Una consulta con WHERE año = 2026 es lenta sobre una tabla de 50 millones de filas pese a tener índice en fecha. Explica dos posibles causas relacionadas con cómo se usa ese índice y qué revisarías en el plan.**

SQLZoo <https://sqlzoo.net/Ejercicios> interactivos de SQL en línea.

## Referencia curricular

|                 |                                           |
|-----------------|-------------------------------------------|
| <b>Grado:</b>   | 3                                         |
| <b>Materia:</b> | SQL y Consultas de Datos · Bases de Datos |

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://univ.lumbre.lat/object/cdia-sql-006-optimizacion-de-consultas>

Material educativo generado desde Lumbre · [lumbre.lat](https://lumbre.lat)