EcomLabs

GUÍA PASO A PASO

EXPLAIN ANALYZE e índices en Postgres

Cómo leer un plan de EXPLAIN ANALYZE, crear el índice correcto para WHERE y ORDER BY, medir antes y después y encontrar los índices que no se usan. Con la tabla de prueba y los resultados medidos.

PARA	Apps con PostgreSQL que tienen una consulta lenta
PROBADO CON	Node 22.22.3 · PGlite 0.5.10 (PostgreSQL 18.6) · Windows 11
ACTUALIZADA	12 de noviembre de 2026
DOCUMENTACIÓN	postgresql.org/docs/current/using-explain.html
ARTÍCULO	ecomlabs.dev/g/indices
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a hacer
Paso 01	La tabla de prueba
Paso 02	Leer el plan
Paso 03	Crear el índice y medir
Paso 04	Revisar los índices que ya tienes
Al terminar	Checklist

ANTES DE EMPEZAR

Qué vas a hacer

Un ciclo de cuatro pasos: medir la consulta con `EXPLAIN ANALYZE`, leer el plan, crear el índice que pide y volver a medir. Lo probamos el 12 de noviembre de 2026 en PGlite 0.5.10 (PostgreSQL 18.6). `EXPLAIN ANALYZE` ejecuta la consulta: úsalo en una copia o en una base de pruebas, y envuelve cualquier `INSERT`, `UPDATE` o `DELETE` en `BEGIN` y `ROLLBACK`.

PASO 01

La tabla de prueba

200,000 pedidos inventados, 5,000 clientes, sin más índice que la clave primaria. Sustituye la tabla por la tuya; lo importante es tener un volumen parecido al real:

```
CREATE TABLE orders (
  id          bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  customer_id int           NOT NULL,
  status      text          NOT NULL,
  created_at  timestamptz   NOT NULL,
  total       numeric(10,2) NOT NULL
);

INSERT INTO orders (customer_id, status, created_at, total)
SELECT (i * 7919) % 5000 + 1,
       (ARRAY['paid','paid','paid','fulfilled','refunded'])[i % 5 + 1],
       timestamptz '2026-01-01' + i * interval '2 minutes',
       round(((i * 37) % 20000) / 100.0, 2)
FROM generate_series(1, 200000) AS i;

ANALYZE orders;
```

PASO 02

Leer el plan

Los últimos 20 pedidos de un cliente:

```
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, created_at, total
FROM orders
WHERE customer_id = 4242
ORDER BY created_at DESC
LIMIT 20;
```

LÍNEA DEL PLAN	QUÉ DICE	MALA SEÑAL
`Seq Scan`	Leyó la tabla entera	Para devolver pocas filas
`Rows Removed by Filter`	Filas leídas y descartadas	Mucho mayor que `rows`
`Sort`	Ordenó las filas después de leerlas	Sobre muchas filas, con un `LIMIT` pequeño
`actual time` y `loops`	Milisegundos por ejecución y veces que corrió	Tiempo por `loops` grande
`Buffers: shared hit/read`	Páginas leídas de caché y de disco	Miles de páginas para pocas filas

Nuestro resultado sin índice: `Seq Scan` con `Rows Removed by Filter: 199960`, 1667 buffers y 15.432 ms.

PASO 03

Crear el índice y medir

Primero las columnas con `=` del `WHERE`, después las del `ORDER BY`:

```
CREATE INDEX orders_customer_created_idx ON orders (customer_id, created_at DESC);
ANALYZE orders;
```

Vuelve a ejecutar la misma consulta, tres veces, y quédate con la tercera. Compara `Buffers` y milisegundos. En producción usa `CREATE INDEX CONCURRENTLY`: no bloquea las escrituras, pero tarda más, no puede ir dentro de una transacción y, si falla, deja un índice `INVALID` que debes borrar y recrear.

VARIANTE	NODO DE LECTURA	BUFFERS	EXECUTION TIME
A · sin índice	Seq Scan + Sort	1667	15.432 ms
B · `(customer_id)`	Bitmap Heap Scan + Sort	42	0.468 ms
D · `(created_at, customer_id)`	Index Scan Backward	400	3.596 ms
C · `(customer_id, created_at DESC)`	Index Scan, sin Sort	24	0.123 ms

Insertar 100,000 filas tardó 703 ms (mediana de tres) con solo la clave primaria y 1257 ms con tres índices más: cada índice cobra en las escrituras.

PASO 04

Revisar los índices que ya tienes

La primera consulta lista los índices sin ningún recorrido registrado (excluye los únicos); la segunda, los que quedaron `INVALID` tras un `CONCURRENTLY` fallido:

```
SELECT s.relname AS table_name, s.indexrelname AS index_name, s.idx_scan,
       pg_size_pretty(pg_relation_size(s.indexrelid)) AS size
FROM pg_stat_user_indexes s
JOIN pg_index i ON i.indexrelid = s.indexrelid
WHERE s.idx_scan = 0 AND NOT i.indisunique
ORDER BY pg_relation_size(s.indexrelid) DESC;

SELECT indexrelid::regclass AS index_name FROM pg_index WHERE NOT indisvalid;
```

> **CUIDADO**
>
> Las estadísticas se reinician tras una caída, un apagado inmediato, una copia base o una recuperación a un punto en el tiempo. Mira un periodo representativo, con los reportes de fin de mes, antes de borrar, y ten a mano el `CREATE INDEX` para recrearlo.

AL TERMINAR

Checklist

- [ ] Lees `actual time`, `Rows Removed by Filter` y `Buffers`, no solo el `cost`.
- [ ] El índice tiene primero las columnas con `=` y después las del `ORDER BY`.
- [ ] Haces `ANALYZE` y comparas antes y después con el mismo volumen de datos.
- [ ] En producción, `CONCURRENTLY` y revisión de índices `INVALID`.
- [ ] `pg_stat_user_indexes` revisado en un periodo representativo antes de borrar.