Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

ARTÍCULO
Original

Análisis de *logs* y *wait events* como metodología estructurada para el diagnóstico del rendimiento en bases de datos relacionales gestionadas

Log and Wait Event Analysis as a Structured Methodology for Performance Diagnosis in Managed Relational Databases

https://doi.org/10.61325/rnzrw588

Angel Enrique Romero Cuevas (autor corresponsal)
Ro440753@uaeh.edu.mx
ORCID: 0009-0004-1703-9435
Universidad Autónoma del Estado de Hidalgo, Hidalgo, México

Gabriel Sánchez Bautista
gabriel_sanchez@uaeh.edu.mx
ORCID: 0000-0002-9955-8711
Universidad Autónoma del Estado de Hidalgo, Hidalgo, México

Recepción	Aceptación	Publicación
17 de octubre de 2026	11 de noviembre de 2026	16 de noviembre de 2026

Resumen

El diagnóstico de problemas de rendimiento en bases de datos relacionales suele realizarse de manera reactiva y mediante el análisis aislado de métricas, lo que dificulta identificar sus causas. El objetivo fue medir la efectividad de una metodología estructurada basada en *logs* y *wait events*. Se empleó un enfoque cuantitativo, con diseño cuasi-experimental de carácter aplicado, comparando dos condiciones sobre una misma instancia de PostgreSQL 16: una condición base sin índices de apoyo y otra optimizada mediante una intervención derivada del diagnóstico. La carga concurrente se generó con pgbench sobre un conjunto sintético de 32,050,000 registros. Se utilizaron Performance Insights, EXPLAIN ANALYZE y pg_stat_statements para analizar los eventos de espera, los planes de ejecución y las estadísticas de las consultas. Durante los tres minutos completos de la condición base, la carga activa promedio fue de 14.867 sesiones frente a 2 vCPU disponibles. Tras la intervención, el rendimiento aumentó de 0.43 a 203.48 transacciones por segundo y la latencia media disminuyó de 18,409 a 39 milisegundos. Los resultados aportaron evidencia favorable sobre la efectividad de la metodología en las condiciones evaluadas, aunque su aplicación a otros entornos requiere validaciones adicionales.

Palabras clave: Rendimiento; Wait Events; Bases de Datos; Diagnóstico; Optimización; Índices; Concurrencia; Observabilidad

Abstract

Performance issues in relational databases are often diagnosed reactively through the isolated analysis of metrics, making it difficult to identify their underlying causes. The objective of this study was to measure the effectiveness of a structured methodology based on logs and wait events. A quantitative approach with an applied quasi-experimental design was employed to compare two conditions on the same PostgreSQL 16 instance: a baseline condition without supporting indexes and an optimized condition following a diagnosis-driven intervention. A concurrent workload was generated using `pgbench` on a synthetic dataset comprising 32,050,000 records. Performance Insights, EXPLAIN ANALYZE, and `pg_stat_statements` were used to analyze wait events, execution plans, and query statistics. During the three complete minutes of the baseline test, the average active load was 14.867 sessions, compared with 2 available vCPUs. Following the intervention, throughput increased from 0.43 to 203.48 transactions per second, while mean latency decreased from 18,409 to 39 milliseconds. The findings provided favorable evidence of the methodology's effectiveness under the evaluated conditions, although its applicability to other environments requires further validation.

Keywords: Performance; Wait Events; Databases; Diagnosis; Optimization; Indexes; Concurrency; Observability

Análisis de logs y wait events como metodología estructurada para el diagnóstico del rendimiento en bases de datos relacionales gestionadas
Angel Enrique Romero Cuevas ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista ORCID: 0000-0002-9955-8711
https://doi.org/10.61325/rnzrw588

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

Introducción

Durante los últimos años, la adopción de plataformas en la nube ha transformado las arquitecturas de los sistemas gestores de bases de datos, favoreciendo el desarrollo de servicios gestionados y nuevas formas de almacenamiento y procesamiento (Stonebraker y Pavlo, 2024). En este contexto, el rendimiento constituye un aspecto fundamental para garantizar el funcionamiento de las aplicaciones, debido a que las variaciones en la carga de trabajo pueden afectar los tiempos de respuesta y la capacidad del sistema para atender las solicitudes. Asimismo, los problemas de rendimiento o las fallas en los componentes pueden comprometer la confiabilidad y la calidad del servicio, lo que hace necesario analizar el comportamiento de los sistemas ante diferentes condiciones de carga (Kleppmann, 2017).

El tiempo de respuesta de un motor de base de datos comprende el tiempo dedicado al procesamiento de las operaciones y el tiempo de espera durante su ejecución. De esta distinción surge el enfoque de diagnóstico basado en los *wait events*, que, en lugar de centrarse únicamente en cómo y dónde se utilizan los recursos, permite identificar qué está esperando el motor y contribuir a localizar los cuellos de botella que afectan su rendimiento (Kompalli, 2024). Los *wait events* permiten identificar diferentes causas de espera, como la lectura de archivos, la contención de estructuras internas y las operaciones de escritura de datos, mientras que el análisis de la actividad de CPU proporciona información complementaria sobre el uso de los recursos. De este modo, el análisis conjunto de las métricas de rendimiento y los registros de telemetría contribuye al diagnóstico y la optimización de los sistemas de bases de datos gestionadas (Abadi et al., 2022; Kompalli, 2024).

La literatura científica ha documentado que las deficiencias en las estrategias de acceso a los datos y la selección de planes de ejecución inadecuados pueden degradar considerablemente el rendimiento de las bases de datos (Leis et al., 2015). Asimismo, las variaciones en las cargas de trabajo requieren anticipar su comportamiento para seleccionar oportunamente las estrategias de optimización (Ma et al., 2018). La observabilidad y la optimización de bases de datos han evolucionado hacia el uso de métricas de rendimiento, el análisis de patrones de carga y el desarrollo de sistemas de ajuste automático mediante aprendizaje por refuerzo (Zhang et al., 2019; Li et al., 2023). Sin embargo, los estudios revisados se orientan principalmente hacia la automatización del ajuste o la evaluación de los optimizadores, mientras que la integración de *logs* de consultas lentas, planes de ejecución y *wait events* en un mismo procedimiento de diagnóstico constituye un aspecto que requiere mayor investigación aplicada en entornos de bases de datos gestionadas.

En la práctica, el diagnóstico de los problemas de rendimiento puede resultar prolongado y realizarse de manera fragmentada, debido a las dificultades para integrar las métricas y los registros del sistema e identificar las causas de la degradación (Huang et al., 2025). Ante situaciones de saturación, una de las alternativas que puede considerarse es ampliar la instancia, lo cual puede resultar costoso y no necesariamente corregir el problema de fondo, especialmente cuando la degradación se relaciona con estrategias de ejecución inadecuadas (Currim et al., 2024). En ese sentido, el análisis de los registros de telemetría y de las estructuras de acceso permite orientar las decisiones de optimización antes de recurrir a una mayor asignación de recursos (Abadi et al., 2022). En este contexto, el problema que aborda la investigación radica en la dificultad de relacionar las evidencias de diagnóstico con la identificación de la causa raíz y los cambios medibles en el rendimiento. Es aquí donde radica la importancia del estudio: proponer un procedimiento de diagnóstico reproducible sobre infraestructura gestionada y evaluar si la metodología basada en *wait events* permite explicar el comportamiento del motor e identificar las causas de la degradación del rendimiento.

Con base en lo anterior, surgió la siguiente pregunta de investigación: ¿hasta qué punto una metodología estructurada basada en logs y wait events permite diagnosticar y corregir cuellos de botella en una base de datos relacional gestionada?

Se planteó la hipótesis de que una metodología estructurada basada en logs y wait events permite identificar la causa raíz de los cuellos de botella y fundamentar intervenciones que reduzcan la latencia y aumenten el rendimiento de una base de datos relacional gestionada. En consecuencia, el objetivo general fue medir la efectividad de una metodología estructurada basada en *logs* y *wait events*. Para ello, el estudio se orientó a: (1) describir el perfil de espera de una carga concurrente en la condición base; (2) identificar la causa raíz mediante el análisis conjunto de *logs*, planes de ejecución y *wait events*; y (3) cuantificar el efecto de la intervención sobre la latencia y el rendimiento.

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711

https://doi.org/10.61325/rnzrw588

Método

Enfoque metodológico

El estudio tuvo un enfoque cuantitativo, puesto que se trabajó con datos numéricos así como métricas estadísticas descriptivas para evaluar el rendimiento del motor de base de datos bajo distintas condiciones.

Diseño de estudio

Se utilizó un diseño cuasi-experimental con medición antes y después sobre una única instancia, sometida inicialmente a una condición base y, posteriormente, a una condición optimizada mediante una intervención derivada del diagnóstico. Se manipuló deliberadamente la variable independiente, manteniendo constantes el esquema, el volumen de datos y el patrón de carga. La ausencia de un grupo de control y de asignación aleatoria, junto con la intervención deliberada en un entorno controlado, fundamenta la clasificación del diseño como cuasi-experimental y delimita su alcance metodológico.

Las condiciones experimentales fueron idénticas en ambas fases: la misma instancia db.t3.medium (2 vCPU y 4 GB de memoria) con PostgreSQL 16.9 y almacenamiento gp3 de 20 GB, el mismo grupo de parámetros del motor, el mismo conjunto de 32,050,000 registros y el mismo script de pgbench, con 8 clientes concurrentes, 4 hilos y 180 segundos por corrida.

- Variable independiente: la presencia de cuatro índices B-tree secundarios, ausentes en la condición base y creados antes de la condición optimizada.
- Variables dependientes: el rendimiento en transacciones por segundo, la latencia media por transacción y su desviación estándar, el tiempo de ejecución de cada consulta según EXPLAIN (ANALYZE, BUFFERS) y la carga activa promedio por tipo de *wait event*.
- Variables controladas: hardware, versión y parámetros del motor, volumen y distribución de los datos, consultas, nivel de concurrencia, duración de la medición y estadísticas del optimizador, que se actualizaron con ANALYZE en las dos condiciones.

Este diseño tiene límites para atribuir los cambios únicamente a la intervención. Cada condición se midió una sola vez y de forma consecutiva, por lo que la segunda corrida pudo encontrar parte de los datos ya cargados en caché; además, la clase db.t3 es de rendimiento ampliable (*burstable*) y el saldo de créditos de CPU pudo variar entre corridas. Aun así, la magnitud del cambio, de 0.43 a 203.48 transacciones por segundo, y la modificación directa de los planes de ejecución, de recorridos secuenciales a accesos indexados, hacen poco probable que esos factores expliquen por sí solos la diferencia.

Tipo de estudio

El estudio fue de carácter aplicado y explicativo: aplicado, porque buscó resolver un problema concreto de ingeniería de rendimiento mediante conocimiento existente sobre optimización de bases de datos; y explicativo, dado que procuró establecer la relación causal entre la estrategia de acceso a los datos y el comportamiento observado del sistema.

Población y muestra

La población conceptual estuvo conformada por las cargas de trabajo transaccionales ejecutadas sobre bases de datos relacionales gestionadas de tipo PostgreSQL en entornos de nube.

La muestra fue una instancia gestionada de PostgreSQL 16.9, con dos núcleos virtuales (vCPU) y cuatro gigabytes de memoria, desplegada en un proveedor de nube, sobre la que se ejecutó una carga sintética reproducible. Esta selección fue intencional, pensada para un escenario controlado donde se pudiera aislar el efecto de las estructuras de acceso. Este conjunto de datos tuvo 32,050,000 de registros repartidos en cuatro tablas, de la siguiente forma:

• Clientes: 2,000,000 de registros.
• Productos: 50,000 registros.
• Órdenes: 10,000,000 de registros (898 MB).
• Líneas de detalle: 20,000,000 de registros (1,731 MB).

Todos los datos se generaron de forma sintética, es decir, no se usó información personal real. Los identificadores de infraestructura (cuenta, punto de conexión y direcciones de red) se anonimizaron con marcadores genéricos.

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711

https://doi.org/10.61325/rnzrw588

Criterios de selección
Criterios de inclusión

Se incluyeron tres consultas representativas de un sistema transaccional:

- Búsqueda de órdenes por cliente: consulta mediante la clave foránea del cliente, incluida en el script de pgbench y ejecutada bajo concurrencia sostenida.
- Agregación de ventas por categoría: consulta realizada en una ventana de tiempo que, debido a su duración de varios segundos, se midió individualmente mediante EXPLAIN (ANALYZE, BUFFERS) en cada condición.
- Recuento de líneas de detalle por producto: consulta incluida en el script de pgbench y ejecutada bajo concurrencia sostenida.

Criterios de exclusión

Se excluyeron del análisis los siguientes elementos para evitar interferencias en las estadísticas recopiladas:

- Consultas administrativas: aquellas generadas por el propio sistema de monitoreo.
- Transacciones de calentamiento: aquellas ejecutadas antes del inicio de la ventana de medición.

Técnica recolección de datos

Se usó la observación instrumental automatizada del comportamiento del motor durante la carga. Las métricas se registraron de forma no intrusiva, tomándolas del propio sistema gestor y de la plataforma de monitoreo del proveedor de nube.

Instrumentos

Se utilizaron cuatro instrumentos complementarios para la generación de carga, el monitoreo y el diagnóstico del rendimiento:

- pgbench: generador de carga configurado con ocho clientes concurrentes, cuatro hilos y una duración de 180 segundos por condición.
- Performance Insights: herramienta de monitoreo utilizada para medir la carga activa promedio (Average Active Sessions, AAS), desagregada por tipo de espera.
- EXPLAIN (ANALYZE, BUFFERS): herramienta empleada para obtener y analizar el plan de ejecución real de cada consulta.
- pg_stat_statements y registros de consultas lentas: instrumentos utilizados conjuntamente, mediante la configuración de log_min_duration_statement, para identificar las sentencias SQL y analizar sus tiempos de ejecución.

Procedimientos

El procedimiento se desarrolló en cinco fases:

1. Aprovisionamiento de la infraestructura: configuración de la instancia de base de datos gestionada y del servidor generador de carga dentro de la misma red virtual privada.
2. Preparación de los datos: carga del conjunto de datos sintéticos en la base de datos.
3. Ejecución de la condición base: realización de las pruebas de carga, registrando simultáneamente los *wait events,* los planes de ejecución y las estadísticas por sentencia.
4. Diagnóstico e intervención: análisis de las evidencias recopiladas para identificar la causa raíz y definir la intervención correspondiente.
5. Ejecución de la condición optimizada: repetición de las pruebas con los mismos parámetros de carga para comparar los resultados de ambas condiciones bajo parámetros experimentales equivalentes.

Análisis de datos

El análisis fue descriptivo y comparativo. Se compararon el rendimiento (transacciones por segundo) y la latencia media entre las dos condiciones. También los perfiles de *wait events,* en Average Active Sessions, contra el número de vCPU, que sirvió de umbral de saturación. Los planes de ejecución se revisaron de forma cualitativa, para confirmar que la intervención sí cambió la estrategia de acceso.

Consideraciones éticas

La investigación no involucró seres humanos ni datos personales, dado que la totalidad de la información fue generada de manera sintética, por lo que no se requirió la aprobación de un comité de ética. Los identificadores de la infraestructura utilizada se anonimizaron con el fin de preservar la seguridad del entorno de pruebas.

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

Resultados

A continuación se presentan los resultados obtenidos en las dos condiciones experimentales. El conjunto de datos estuvo compuesto por 32,050,000 de registros sintéticos; la tabla de órdenes ocupó 898 MB y la de líneas de detalle 1,731 MB. Juntas suman cerca de 2.63 GB, por debajo de los 4 GB de memoria total de la instancia, pero muy por encima del área de búferes compartidos de PostgreSQL (shared_buffers), que en Amazon RDS se configura por defecto en alrededor del 25 % de la memoria, es decir, cerca de 1 GB; el resto de la memoria lo comparten el sistema operativo y los procesos del motor. Los registros respaldan esta lectura: en la condición base, la tabla de órdenes tuvo apenas un 5.3 % de aciertos en el búfer y acumuló 6.88 millones de bloques leídos fuera de él, y solo la consulta de órdenes por cliente sumó 47.2 segundos de tiempo de lectura medido con track_io_timing, acumulado entre sus procesos paralelos, es decir, unos 0.54 ms por bloque de 8 kB, un valor compatible con lecturas a almacenamiento y no a memoria. Este aspecto es determinante para interpretar el comportamiento de entrada/salida observado.

Rendimiento en la condición base

Tabla 1

Comparativo de rendimiento y planes de ejecución entre la condición base y la optimizada

Métrica	Condición base	Condición optimizada	Cambio
Rendimiento (TPS)	0.43	203.48	x472.17
Latencia media (ms)	18,408.8	39.31	–99.79 %
Desviación estándar (ms)	7,154.3	46.83	–99.35 %
Transacciones completadas	81	36,613	x452
Duración efectiva medida (s)	187.96	179.93	
Órdenes por cliente -> plan	Seq Scan (16,188.8 ms)	Index Scan (2.6 ms)	–99.98 %
Detalle por producto -> plan	Seq Scan (2,522.3 ms)	Bitmap Index Scan (51.0 ms)	–97.98 %
Agregación por categoría -> plan	Seq Scan + Hash Join (13,197.1 ms)	Bitmap Heap Scan + Hash Join (7,495.8 ms)	-43.2 %

Nota. Carga generada con pgbench (8 clientes, 4 hilos, 180 s) sobre PostgreSQL 16.9. Los tiempos de plan corresponden a la query "EXPLAIN (ANALYZE, BUFFERS)". De elaboración propia. El rendimiento (TPS) es el valor que reporta pgbench al final de cada corrida, con seis decimales: 0.430949 en la condición base y 203.481191 en la optimizada, calculado sin contar el tiempo inicial de conexión (79.5 ms y 87.2 ms, respectivamente). pgbench no informa la duración efectiva de forma directa, así que se obtuvo dividiendo las transacciones completadas entre ese TPS: 81 / 0.430949 = 187.96 s y 36,613 / 203.481191 = 179.93 s. El valor de 203.52 TPS que resulta al dividir entre 179.9 s se debe a que la duración se había redondeado a un decimal. Las dos corridas se programaron con 180 s (opción -T 180). Con esa opción, pgbench deja de iniciar transacciones al cumplirse el tiempo, pero espera a que terminen las que están en curso. En la condición base cada transacción tardaba en promedio 18.4 s, de modo que las últimas terminaron cerca de 8 s después del límite; por eso las marcas de tiempo del registro de ejecución muestran 188 s de reloj, de 22:49:57 a 22:53:05 UTC. En la condición optimizada, con transacciones de 39 ms, la corrida terminó a los 180 s de reloj, de 22:57:29 a 23:00:29 UTC, y la duración efectiva quedó apenas por debajo por el descuento del tiempo de conexión. Los registros completos de pgbench se adjuntan como material suplementario. La agregación por categoría se midió de forma individual con EXPLAIN (ANALYZE, BUFFERS).

En la Tabla 1 se presentan los resultados de rendimiento y los planes de ejecución correspondientes a la condición base. La carga concurrente alcanzó un rendimiento de 0.43 transacciones por segundo, con una latencia media de 18,408.8 milisegundos (desviación estándar de 7,154.3 ms) y un total de 81 transacciones completadas durante una prueba programada de 180 segundos, cuya duración efectiva se extendió a 187.96 segundos debido a que pgbench esperó la finalización de las transacciones en curso, según se explica en la nota de la tabla. El análisis de los planes de ejecución evidenció recorridos secuenciales sobre las tablas de mayor tamaño: la consulta de órdenes por cliente realizó un recorrido secuencial paralelo con un tiempo de ejecución de 16,188.8 milisegundos, mientras que la consulta de detalle por producto requirió 2,522.3 milisegundos. Asimismo, el registro de consultas lentas y las estadísticas de pg_stat_statements permitieron identificar dos sentencias que concentraron el tiempo de ejecución, con 6.7 y 4.1 millones de bloques leídos desde disco, respectivamente. Adicionalmente, la tabla de órdenes presentó una tasa de aciertos en caché de apenas el 5.3 %, lo que evidencia una baja proporción de lecturas atendidas desde los búferes y

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

una elevada necesidad de acceder a los datos fuera de ellos.

Perfil de *wait events* y analisís de la causa raíz

El análisis de los *wait events* permitió identificar el comportamiento de los recursos durante la condición base. Al considerar los tres intervalos completos de un minuto correspondientes a la ejecución de pgbench, la carga activa promedio alcanzó 14.867 sesiones (Average Active Sessions, AAS), equivalente a 7.43 veces las 2 vCPU disponibles. La distribución de esta carga mostró un predominio de CPU (11.794 AAS), seguida de IO (1.189 AAS), LWLock (1.494 AAS) y otros eventos (0.389 AAS). La carga activa total alcanzó un máximo de 14.900 sesiones.

Estos resultados, junto con los planes de ejecución y las estadísticas de las consultas, permitieron identificar la ausencia de índices de apoyo como una de las principales causas de la degradación del rendimiento. Los recorridos secuenciales sobre tablas de gran tamaño incrementaban el procesamiento de filas y las operaciones de lectura, lo que explica la elevada carga asociada a CPU y las esperas de entrada/salida observadas durante la condición base.

Tabla 2

Carga activa registrada por Performance Insights en las ventanas de monitoreo de ambas condiciones

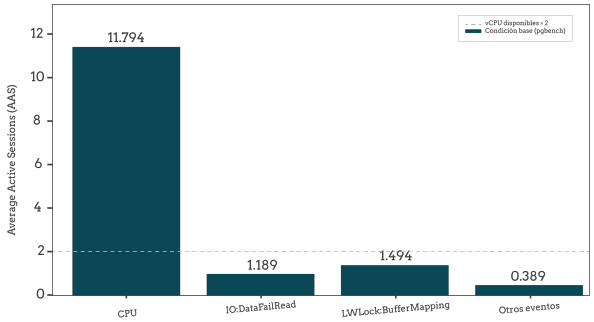
Wait event	Base prom.	Base máx.	Optim. prom.	Optim. máx.
DBLoad total	6.603	14.900	3.638	7.800
CPU	4.933	11.933	2.101	4.560
IO:DataFileRead	0.797	1.367	1.432	3.160
LWLock:BufferMapping	0.583	1.467	0.030	0.083
Otros eventos	0.290	n/d	0.075	n/d

Nota. Datos obtenidos mediante Performance Insights (db.load.avg), con una resolución de 60 segundos. La instancia dispone de 2 vCPU. Los promedios se calcularon sobre ventanas de monitoreo de cinco minutos para la condición base y siete para la optimizada, incluyendo actividades adicionales a las pruebas de pgbench. Por tanto, los valores no representan mediciones experimentales directamente comparables. La categoría "Otros eventos" agrupa esperas menores de IPC, Timeout, Client e I/O. Los máximos de los componentes no necesariamente ocurrieron en el mismo intervalo.

La Tabla 2 presenta los promedios registrados por Performance Insights en las ventanas de monitoreo de ambas condiciones, las cuales se definieron según el momento de consulta a la plataforma y no coincidieron exactamente con el inicio y la finalización de las pruebas de pgbench. En la condición base, la ventana de cinco minutos incluyó tres minutos previos a la ejecución y dos minutos completos de la prueba. Sin embargo, al analizar exclusivamente los tres minutos completos de pgbench, se obtuvo una carga activa promedio de 14.867 sesiones (AAS), distribuida en CPU (11.794), IO:DataFileRead (1.189), LWLock:BufferMapping (1.494) y otros eventos (0.389), con un máximo de 14.900 AAS. Por su parte, la ventana de siete minutos de la condición optimizada incluyó, además de la ejecución de pgbench, la creación de los índices (73 segundos) y las consultas EXPLAIN. Debido a que no se conservaron los registros minuto a minuto de esta condición, no fue posible aislar los valores correspondientes exclusivamente a la prueba. En consecuencia, los promedios de ambas condiciones incluyen actividades adicionales y no permiten establecer una comparación directa de la carga activa experimental ni atribuir sus diferencias únicamente a la intervención realizada.

Figura 1

Perfil de carga activa durante la condición base



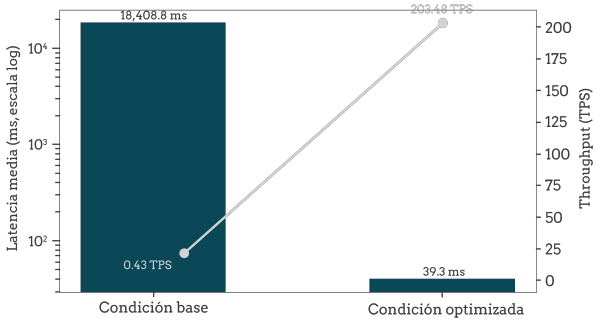
Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

Nota. Distribución de la carga activa promedio (AAS) por tipo de evento durante los tres minutos completos de ejecución de pgbench en la condición base. La línea discontinua representa las 2 vCPU disponibles en la instancia. Datos obtenidos mediante Performance Insights (db.load.avg).

Rendimiento en la condición optimizada

Figura 2
Rendimiento y latencia media: condición base frente a optimizada



Nota. La latencia se representa en escala logarítmica.

En la Figura 2 se presenta la comparación del rendimiento transaccional y la latencia media entre la condición base y la optimizada. La intervención consistió en crear cuatro índices B-tree sobre las columnas de filtrado y combinación, proceso que requirió 73 segundos. Tras la implementación de los índices, la condición optimizada, evaluada con la misma carga de trabajo, alcanzó un rendimiento de 203.48 transacciones por segundo, con 36,613 transacciones completadas. La latencia media disminuyó a 39.31 milisegundos, con una desviación estándar de 46.83 ms. Los planes de ejecución mostraron cambios de recorridos secuenciales a accesos indexados (Tabla 1): la consulta de órdenes por cliente pasó a utilizar un Index Scan, con un tiempo de ejecución de 2.638 milisegundos, mientras que la consulta de detalle por producto empleó un Bitmap Index Scan, con 50.958 milisegundos. Por su parte, la agregación por categoría redujo su tiempo de ejecución de 13,197.1 a 7,495.8 milisegundos, equivalente a una mejora del 43.2 %. En esta consulta, el índice sobre la fecha de la orden permitió utilizar un Bitmap Heap Scan para acceder a la tabla de órdenes; sin embargo, las líneas de detalle continuaron procesándose mediante un recorrido secuencial, debido al elevado número de filas involucradas en la combinación. En cuanto a los *wait events,* la ventana de monitoreo de la condición optimizada registró una carga activa promedio de 3.638 sesiones (AAS) y 0.030 AAS asociados a LWLock:BufferMapping. No obstante, estos valores incluyen actividades adicionales a la prueba de pgbench y, por tanto, no permiten cuantificar directamente una reducción de la carga activa respecto de la condición base. En conjunto, los resultados de pgbench evidenciaron que el rendimiento transaccional se multiplicó aproximadamente por 472 y la latencia media se redujo un 99.79 %, mientras que las consultas individuales registraron reducciones de hasta el 99.98 % en sus tiempos de ejecución.

Los cuatro índices se definieron a partir de las columnas que aparecían como filtro o como condición de combinación en los recorridos secuenciales detectados en los planes de ejecución. Las instrucciones utilizadas fueron las siguientes:

- CREATE INDEX idx_orders_customer_id ON orders (customer_id); en la tabla de órdenes, para el filtro por cliente de la primera consulta (112 MB).
- CREATE INDEX idx_order_items_product_id ON order_items (product_id); en la tabla de líneas de detalle, para el filtro por producto de la tercera consulta (136 MB).
- CREATE INDEX idx_orders_order_date ON orders (order_date); en la tabla de órdenes, para el filtro por rango de fechas de la agregación por categoría (214 MB).
- CREATE INDEX idx_order_items_order_id ON order_items (order_id); en la tabla de líneas de detalle, para la combinación con órdenes dentro de la agregación (321 MB).

Al terminar se ejecutó ANALYZE sobre ambas tablas para actualizar las estadísticas del optimizador. Las tablas de clientes y productos no recibieron índices nuevos: la primera no intervenía en las consultas evaluadas y la segunda, con 50,000 filas, se recorría completa en unos 30 a 40 milisegundos.

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711

https://doi.org/10.61325/rnzrw588

Discusión

Dentro del alcance de un diseño cuasi-experimental de un solo grupo, con una medición por condición, los resultados sugieren que una metodología estructurada basada en el análisis de *logs* y *wait events* puede contribuir al diagnóstico y la corrección de cuellos de botella en una base de datos relacional gestionada. Tras la intervención derivada del diagnóstico, el rendimiento transaccional aumentó de 0.43 a 203.48 transacciones por segundo y la latencia media disminuyó de 18,409 a 39 milisegundos. Estos resultados son consistentes con la hipótesis planteada y sugieren que, en las condiciones evaluadas, la ausencia de índices de apoyo constituyó una de las principales causas de la degradación del rendimiento. Estos hallazgos han sido coherentes con lo señalado por Leis et al. (2015) respecto de la influencia de las estrategias de acceso y la selección de planes de ejecución sobre el rendimiento. Asimismo, se relacionaron con la importancia de considerar las características de la carga de trabajo para orientar las decisiones de optimización, aspecto abordado por Ma et al. (2018). A diferencia de estos antecedentes, la identificación del problema en el presente estudio se sustentó en el análisis conjunto de los wait events, los planes de ejecución y las estadísticas por sentencia, mientras que las diferencias de rendimiento posteriores a la intervención se evaluaron mediante pgbench y los cambios observados en los planes de ejecución. No obstante, la ausencia de ventanas equivalentes en Performance Insights impidió cuantificar directamente la reducción de la carga activa entre ambas condiciones.

Respecto al primer objetivo, orientado a describir el perfil de espera en la condición base, la carga activa promedio durante los tres minutos completos de ejecución de pgbench fue de 14.867 sesiones (AAS), equivalente a 7.43 veces las 2 vCPU disponibles, con un máximo de 14.900 sesiones. Predominaron las sesiones asociadas a CPU (11.794 AAS), seguidas de LWLock (1.494 AAS) e I/O (1.189 AAS). Estos resultados guardaron relación con lo planteado por Abadi et al. (2022) acerca de la importancia del análisis de los registros de telemetría para comprender el comportamiento de las bases de datos y orientar las decisiones de optimización. La elevada carga activa, considerada de manera aislada, podría sugerir la necesidad de ampliar los recursos de la instancia; sin embargo, el análisis conjunto de los wait events, los planes de ejecución y las estadísticas de las consultas permitió identificar recorridos secuenciales sobre tablas de gran tamaño, asociados con una elevada actividad de CPU y operaciones de entrada/salida. Este comportamiento fue consistente con lo señalado por Harizopoulos et al. (2018) acerca de la influencia de las operaciones internas del motor sobre el rendimiento del procesamiento transaccional, particularmente los costos asociados con la gestión de búferes y otros mecanismos del sistema. No obstante, las diferencias entre las ventanas de monitoreo impidieron establecer una comparación directa de la carga activa entre las condiciones base y optimizada.

En torno al segundo objetivo, identificar la causa raíz mediante el análisis conjunto de las fuentes de información, los resultados mostraron la importancia de examinar las métricas de manera complementaria. Las estadísticas de pg_stat_statements permitieron cuantificar los bloques leídos y los tiempos de ejecución de las consultas, mientras que los registros de consultas lentas contribuyeron a identificar las sentencias con mayor duración. Por su parte, los *wait events* proporcionaron información sobre la actividad de CPU y las esperas de entrada/salida, y los planes de ejecución evidenciaron recorridos secuenciales sobre las tablas de mayor tamaño. La integración de estas evidencias permitió vincular la degradación del rendimiento con la ausencia de índices de apoyo en las columnas utilizadas para el filtrado y la combinación de datos. Estos hallazgos coincidieron con Van Aken et al. (2021) en cuanto a la importancia de analizar las métricas del sistema y las características de las consultas para orientar las decisiones de optimización, aunque su investigación se centró en el ajuste automático de parámetros mediante aprendizaje automático. Asimismo, Huang et al. (2025) destacaron la necesidad de relacionar las métricas de rendimiento con las consultas y sus operadores para identificar las causas de las anomalías, lo que resulta coherente con el enfoque de diagnóstico integrado aplicado en el presente estudio.

En lo referente al tercer objetivo, dirigido a cuantificar el efecto de la intervención sobre la latencia y el rendimiento, los resultados evidenciaron que las consultas no mejoraron en la misma proporción. Las consultas selectivas registraron reducciones considerables en sus tiempos de ejecución, mientras que la agregación con combinaciones de gran volumen presentó una mejora del 43.2 %, al disminuir de 13,197.1 a 7,495.8 milisegundos. Esta diferencia se alineó con las características de acceso a los datos: los índices favorecieron las consultas que recuperaban un conjunto reducido de filas, mientras que las operaciones que requerían procesar una proporción elevada de registros conservaron recorridos secuenciales. En ese sentido, Tong et al. (2025) señalaron que la utilidad de los índices depende de la proporción de datos que deben recuperarse, siendo mayor en consultas selectivas y menor cuando se necesita acceder a una parte considerable

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711

https://doi.org/10.61325/rnzrw588

de los registros. Los resultados del presente estudio fueron coherentes con este planteamiento, dado que las consultas evaluadas presentaron mejoras de distinta magnitud y la indexación no eliminó todos los costos asociados a su procesamiento. Por su parte, Shin et al. (2022) abordaron la optimización de operaciones de combinación mediante un algoritmo especializado para índices invertidos, orientado a reducir comparaciones innecesarias durante la ejecución de las consultas. Si bien su propuesta se desarrolló para una estructura de indexación diferente de la utilizada en esta investigación, ambos estudios han reconocido que las mejoras del rendimiento no solo dependen de la incorporación de índices, sino también de las características de las operaciones y de las estrategias empleadas para ejecutarlas.

La comparación con estudios previos permitió identificar tres aspectos relevantes. En primer lugar, la importancia de las estrategias de acceso y optimización de consultas, abordada por Leis et al. (2015) y Solohub y Pashkevych (2026), quienes examinaron la influencia de los planes de ejecución y las técnicas de optimización sobre el rendimiento. En segundo lugar, el papel de las estructuras de indexación en el rendimiento de los sistemas relacionales, evaluado por Anchlia (2024) y Hashim (2026), cuyos hallazgos se relacionaron con la reducción del tiempo de ejecución de la consulta de órdenes por cliente, de 16,188.8 a 2.6 milisegundos. En tercer lugar, las limitaciones de la indexación frente a operaciones que requieren procesar grandes volúmenes de datos, aspecto abordado por Tong et al. (2025), quienes señalaron que la utilidad de los índices disminuye cuando las consultas necesitan acceder a una proporción elevada de registros, cuyo planteamiento ha sido coherente con la mejora del 43.2 % registrada en la consulta de agregación del presente estudio. Asimismo, Abbasi et al. (2024) documentaron experimentalmente que las mejoras asociadas a distintas estrategias de indexación no son uniformes, con reducciones del tiempo de ejecución de entre 32.4 % y 48.6 % en los escenarios evaluados. A diferencia de los enfoques de automatización del ajuste de parámetros o selección de índices desarrollados por Trummer (2022), Lao et al. (2024) y Barbosa et al. (2024), el presente estudio aplicó un procedimiento de diagnóstico basado en la interpretación conjunta de los *wait events*, los planes de ejecución y las estadísticas de las consultas para fundamentar la intervención realizada.

Desde una perspectiva práctica, los resultados sugirieron que el análisis estructurado de las métricas disponibles puede contribuir a identificar problemas de rendimiento en bases de datos gestionadas, utilizando herramientas del propio motor y de la plataforma de monitoreo. Esta característica resulta pertinente en entornos donde el acceso a determinados recursos de infraestructura es limitado y se relaciona con la importancia de analizar el comportamiento del motor para orientar las decisiones de optimización, aspecto abordado por Ma et al. (2021) en el desarrollo de sistemas de gestión autoadministrables. Por su parte, Lim et al. (2024) exploraron la recopilación eficiente de datos para entrenar modelos de comportamiento en sistemas autónomos, a diferencia del presente estudio, que se centró en la interpretación conjunta de evidencias para orientar una intervención específica. En el caso evaluado, la revisión de los *wait events* y los planes de ejecución permitió dirigir la intervención hacia la creación de índices de apoyo, sin modificar la capacidad de la instancia. Por ello, los hallazgos respaldaron la conveniencia de examinar las estrategias de acceso a los datos antes de considerar una ampliación de los recursos de cómputo. No obstante, dado que no se evaluó experimentalmente una condición de escalamiento de hardware ni se compararon diferentes metodologías de diagnóstico, no es posible establecer conclusiones sobre su eficacia relativa o sus costos.

Entre las fortalezas del estudio se encontró la posibilidad de reproducir el procedimiento experimental mediante una herramienta estándar de generación de carga, parámetros de ejecución explícitos e instrucciones documentadas para la creación de los índices. A su vez, el empleo de datos sintéticos permitió desarrollar las pruebas en un entorno controlado, mientras que el análisis conjunto de los *wait events*, los planes de ejecución y las estadísticas por sentencia proporcionó diferentes fuentes de información para fundamentar el diagnóstico.

Al momento de interpretar los resultados es importante considerar las limitaciones del estudio, entre ellas, que las pruebas se realizaron en una única instancia de tamaño moderado, con datos sintéticos y una sola medición por condición, por lo que la magnitud de las mejoras observadas no puede generalizarse directamente a entornos de producción. Aunque cada corrida se programó para 180 segundos, los registros de pgbench evidenciaron variaciones del rendimiento durante la condición optimizada, con valores entre 96.3 y 459.0 transacciones por segundo en los intervalos de 30 segundos. Por consiguiente, no es posible afirmar que el rendimiento se estabilizó durante el periodo evaluado ni determinar las causas de dichas variaciones con la información disponible. Además, la duración de las pruebas no permitió evaluar efectos a largo plazo, como la fragmentación de los índices o los cambios de carga a lo largo del día. Al respecto, Kurmanji y Triantafillou

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711

https://doi.org/10.61325/rnzrw588

(2023) han resaltado la necesidad de evaluar las estrategias de optimización bajo condiciones variables, lo cual no ha sido abordado en el presente estudio. Tampoco se cuantificó el costo de mantenimiento de los índices durante las operaciones de escritura, aspecto señalado por Das et al. (2019), quienes documentaron que estos costos pueden afectar negativamente el rendimiento de dichas operaciones. Sumado a ello, el posible efecto de los datos almacenados en caché durante la segunda corrida, la variación de los créditos de CPU de la instancia db.t3 y las diferencias entre las ventanas de monitoreo de Performance Insights impidieron comparar directamente la carga activa experimental de ambas condiciones.

En conjunto, los resultados aportaron evidencia favorable sobre la utilidad del análisis combinado de wait events, planes de ejecución y estadísticas por sentencia para identificar problemas relacionados con las estrategias de acceso a los datos. El principal aporte del estudio radicó en la aplicación documentada de un procedimiento de diagnóstico que permitió fundamentar la selección de índices y evaluar los cambios de rendimiento en una base de datos relacional gestionada. Sin embargo, la confirmación de su efectividad en otros escenarios requiere mediciones repetidas y evaluaciones con diferentes cargas de trabajo.

Conclusiones

Los resultados obtenidos aportaron evidencia favorable sobre la efectividad de una metodología estructurada basada en el análisis de *logs* y *wait events* para diagnosticar problemas de rendimiento en una base de datos relacional gestionada, dentro de las condiciones evaluadas. La integración de los eventos de espera, los planes de ejecución y las estadísticas de las consultas permitió identificar la ausencia de índices de apoyo como un factor relevante de la degradación del rendimiento y fundamentar una intervención que produjo mejoras considerables en la latencia y el rendimiento transaccional. Sin embargo, las diferencias observadas entre las consultas evidenciaron que los beneficios de la indexación dependen de las características de acceso y del volumen de datos procesados.

Estos hallazgos respaldan la utilidad de un procedimiento de diagnóstico guiado por la evidencia para identificar problemas relacionados con las estrategias de acceso a los datos y orientar decisiones de optimización antes de considerar la ampliación de los recursos de cómputo. No obstante, el estudio se desarrolló con datos sintéticos, en una única instancia y con una sola medición por condición. Asimismo, las diferencias entre las ventanas de monitoreo de Performance Insights impidieron cuantificar directamente la reducción de la carga activa atribuible a la intervención. Por tanto, los resultados deben interpretarse dentro de las condiciones experimentales evaluadas, sin generalizar su magnitud a otros entornos.

Finalmente, se recomienda ampliar la evaluación mediante mediciones repetidas y cargas de trabajo reales, así como incorporar el análisis del equilibrio entre las operaciones de lectura y escritura. Estas líneas de investigación permitirían examinar la consistencia de los resultados y explorar la automatización del diagnóstico mediante la detección continua de anomalías en los eventos de espera.

Agradecimientos

El autor agradece a la Universidad Autónoma del Estado de Hidalgo por el apoyo brindado durante el desarrollo de esta investigación.

Referencias

Abadi, D., Ailamaki, A., Andersen, D. G., Bailis, P., Balazinska, M., Bernstein, P. A., Boncz, P. A., Chaudhuri, S., Cheung, A., Doan, A., Dong, L., Franklin, M. J., Freire, J., Halevy, A. Y., Hellerstein, J. M., Idreos, S., Kossmann, D., Kraska, T., Krishnamurthy, S., ... Suciu, D. (2022). The Seattle report on database research. Communications of the ACM, 65(8), 72–79. https://doi.org/10.1145/3524284

Abbasi, M., Bernardo, M. V., Váz, P., Silva, J., & Martins, P. (2024). Revisiting database indexing for parallel and accelerated computing: A comprehensive study and novel approaches. Information, 15(8), Article 429. https://doi.org/10.3390/info15080429

Anchlia, A. (2024). Enhancing query performance through relational database indexing. International Journal of Computer

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

Trends and Technology, 72(8), 130–133. https://doi.org/10.14445/22312803/IJCTT-V72I8P119

Barbosa, D., Gruenwald, L., D'Orazio, L., & Bernardino, J. (2024). QRLIT: Quantum reinforcement learning for database index tuning. Future Internet, 16(12), Article 439. https://doi.org/10.3390/fi16120439

Currim, S., Snodgrass, R. T., & Suh, Y. K. (2024). Identifying the root causes of DBMS suboptimality. ACM Transactions on Database Systems, 49(1), Article 4. https://doi.org/10.1145/3636425

Das, S., Grbic, M., Ilic, I., Jovandic, I., Jovanovic, A., Narasayya, V. R., Radulovic, M., Stikic, M., Xu, G., & Chaudhuri, S. (2019). Automatically indexing millions of databases in Microsoft Azure SQL Database. In Proceedings of the 2019 International Conference on Management of Data (pp. 666–679). Association for Computing Machinery. https://doi.org/10.1145/3299869.3314035

Harizopoulos, S., Abadi, D. J., Madden, S., & Stonebraker, M. (2018). OLTP through the looking glass, and what we found there. En M. L. Brodie (Ed.), Making databases work: The pragmatic wisdom of Michael Stonebraker (pp. 409–439). Association for Computing Machinery. https://doi.org/10.1145/3226595.3226635

Hashim, H. B. (2026). Performance evaluation of B-Tree and hash indexing under varying data sizes in relational database systems. Computing and Artificial Intelligence, 4(1), Article 4104. https://doi.org/10.59400/cai4104

Huang, S., Wang, Z., Wu, Y., Tu, Y., Wang, J., & Cui, B. (2025). OpDiag: Unveiling database performance anomalies through query operator attribution. IEEE Transactions on Knowledge and Data Engineering, 37(6), 3613–3626. https://doi.org/10.1109/TKDE.2025.3557049

Kleppmann, M. (2017). Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. O'Reilly Media.

Kompalli, K. (2024). Automated monitoring and alerting frameworks for enterprise databases: Architecture, thresholds, and root-cause identification. International Journal of Scientific Research in Science and Technology, 11(4), 766–780. https://doi.org/10.32628/IJSRST25125216

Kurmanji, M., & Triantafillou, P. (2023). Detect, distill and update: Learned DB systems facing out of distribution data. Proceedings of the ACM on Management of Data, 1(1), 1–27. https://doi.org/10.1145/3588713

Lao, J., Wang, Y., Li, Y., Wang, J., Zhang, Y., Cheng, Z., Chen, W., Tang, M., & Wang, J. (2024). GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization. Proceedings of the VLDB Endowment, 17(8), 1939–1952. https://doi.org/10.14778/3659437.3659449

Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How good are query optimizers, really? Proceedings of the VLDB Endowment, 9(3), 204–215. https://doi.org/10.14778/2850583.2850594

Li, Z., Tu, Y., & Ma, Z. (2023). A sample-aware database tuning system with deep reinforcement learning. Journal of Database Management, 35(1), 1–25. https://doi.org/10.4018/JDM.333519

Lim, W. S., Ma, L., Zhang, W., Butrovich, M., Arch, S., & Pavlo, A. (2024). Hit the gym: Accelerating query execution to efficiently bootstrap behavior models for self-driving database management systems. Proceedings of the VLDB Endowment, 17(11), 3680–3693. https://doi.org/10.14778/3681954.3682030

Ma, L., Van Aken, D., Hefny, A., Mezerhane, G., Pavlo, A., & Gordon, G. J. (2018). Query-based workload forecasting for self-driving database management systems. In Proceedings of the 2018 International Conference on Management of Data (pp. 631–645). Association for Computing Machinery. https://doi.org/10.1145/3183713.3196908

Ma, L., Zhang, W., Jiao, J., Wang, W., Butrovich, M., Lim, W. S., Menon, P., & Pavlo, A. (2021). MB2: Decomposed behavior modeling for self-driving database management systems. In Proceedings of the 2021 International Conference on Management of Data (pp. 1248–1261). Association for Computing Machinery. https://doi.org/10.1145/3448016.3457276

Shin, Y., Ahn, J., & Im, D. (2022). Join optimization for inverted index technique on relational database management systems. Expert Systems with Applications, 198, Article 116956. https://doi.org/10.1016/j.eswa.2022.116956

Solohub, V., & Pashkevych, V. (2026). Comparative analysis of query optimization techniques in modern relational database

Sciencevolution
ISSN: 2810-8728 (En línea)
5.4
OCTUBRE - DICIEMBRE
2026
Artículo Original
7- 18

Gabriel Sánchez Bautista
ORCID: 0000-0002-9955-8711
Angel Enrique Romero Cuevas
ORCID: 0009-0004-1703-9435
https://doi.org/10.61325/rnzrw588

systems. Journal of Computer Sciences Institute, 39, 132–137. https://doi.org/10.35784/jcsi.8944

Stonebraker, M., & Pavlo, A. (2024). What goes around comes around... and around... ACM SIGMOD Record, 53(2), 21–37. https://doi.org/10.1145/3685980.3685984

Tong, Y., Liu, F., Xu, J., Wang, H., Zhou, K., & Miao, J. (2025). BISLearner: Block-aware index selection using attention-based reinforcement learning for data analytics. ACM Transactions on Database Systems, 51(1), Article 2. https://doi.org/10.1145/3760773

Trummer, I. (2022). DB-BERT: A database tuning tool that reads the manual. In Proceedings of the 2022 International Conference on Management of Data (pp. 190–203). Association for Computing Machinery. https://doi.org/10.1145/3514221.3517843

Van Aken, D., Yang, D., Brillard, S., Fiorino, A., Zhang, B., Billian, C., & Pavlo, A. (2021). An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. Proceedings of the VLDB Endowment, 14(7), 1241–1253. https://doi.org/10.14778/3450980.3450992

Zhang, J., Liu, Y., Zhou, K., Li, G., Xiao, Z., Cheng, B., Xing, J., Wang, Y., Cheng, T., Liu, L., Ran, M., & Li, Z. (2019). An end-to-end automatic cloud database tuning system using deep reinforcement learning. In Proceedings of the 2019 International Conference on Management of Data (pp. 415–432). Association for Computing Machinery. https://doi.org/10.1145/3299869.3300085

Conflicto de intereses
La autora declara que no existe ningún conflicto de interés relacionado con la investigación, la autoría o la publicación del presente artículo.

Financiación
Todos los costos asociados a la realización de este estudio fueron cubiertos por el autor. No se recibió financiación, subvención ni patrocinio externo.

Nota del editor
La revista Sciencevolution mantiene una posición neutral respecto a posibles reclamaciones jurisdiccionales derivadas de mapas publicados o afiliaciones institucionales.

© El/Los autor(es) 2026. Este artículo se distribuye bajo los términos de la Licencia Internacional Creative Commons Atribución 4.0 (CC BY 4.0), que permite el uso, la distribución, la adaptación y la reproducción en cualquier medio o formato, siempre que se otorgue el crédito adecuado al/los autor(es) y a la fuente, se incluya un enlace a la licencia y se indique si se han realizado cambios.

Las imágenes u otro material de terceros incluidos en este artículo están cubiertos por la licencia Creative Commons del artículo, salvo que se indique lo contrario en una línea de crédito. Si el material no está incluido en la licencia Creative Commons y el uso previsto no está permitido por la legislación vigente o excede el uso permitido, será necesario obtener autorización directamente del titular de los derechos.

Puede consultarse una copia de la licencia en: http://creativecommons.org/licenses/by/4.0/

La renuncia a la dedicación al dominio público de Creative Commons (CC0 1.0; http://creativecommons.org/publicdomain/zero/1.0/)
Se aplica a los datos disponibles en este artículo, salvo indicación contraria en la línea de crédito correspondiente.