ISSN-E: 2542-3401, ISSN-P: 1316-4821

Universidad, Ciencia y Tecnología,
vol.26, Núm. 114, (pp. 29-38)

https://doi.org/10.47460/uct.v26i114.588

Buenas prácticas para la optimización de base de datos relacionales usando Microsoft SQL

Gisella Luisa Elena Maquen Niño
https://orcid.org/0000-0002-9224-5456
gluisamn@unprg.edu.pe
Universidad Nacional Pedro Ruiz Gallo
Lambayeque, Perú

Franklin Edinson Terán Santa Cruz
https://orcid.org/0000-0002-3197-7979
fteran@unprg.edu.pe
Universidad Nacional Pedro Ruiz Gallo
Lambayeque, Perú

Consuelo Ivonne del Castillo Castro
https://orcid.org/0000-0002-1512-006X
cdelcastilloc@unprg.edu.pe
Universidad Nacional Pedro Ruiz Gallo
Lambayeque, Perú

Rafael Damián Villón Prieto
https://orcid.org/0000-0002-5248-4858
villonpr@ucvvirtual.edu.pe
Universidad Cesar Vallejo
Lambayeque, Perú.

Recibido (27/02/2022), Aceptado (07/04/2022)

Resumen.- El principal problema al consultar una base de datos es el tiempo de respuesta. La investigación fue de tipo aplicada, utilizó dos bases de datos: control y experimental. Se han utilizado tres computadoras para la ejecución de diez consultas a las dos bases de datos, ejecutándose 4 veces consecutivas y obteniendo un promedio. Los resultados encontrados fueron que, de las diez consultas realizadas, siete obtuvieron mejores resultados en la BD experimental y tres en la BD control. Se concluye que las buenas prácticas para optimizar una base de datos son: crear índices agrupados en columnas usadas frecuentemente para realizar búsqueda u ordenamientos, crear índices no agrupados en columnas usadas para comparaciones que no tengan índices agrupados, utilizar columnas calculadas, operadores y listado de columnas adecuadas en las consultas, sin embargo, la utilización de índices debe restringirse porque afectan a las operaciones de inserción, actualización y eliminación.

Palabras clave: base de datos relacionales, optimización de base de datos, SQL, indexación

Best Practices for Relational Database Optimization using Microsoft SQL

Abstract.- The main problem when querying a database is the response time. The research was of an applied type, using two databases: control and experimental. Three computers have been used to execute ten queries to the two databases, running 4 consecutive times and obtaining an average. The results found were that, of the ten consultations carried out, seven consultations obtained better results in the experimental DB and three in the control DB. It is concluded that the best practices to optimize a database are: create clustered indexes on columns frequently used in searches or to perform sorts, create non-clustered indexes on primary or foreign keys that do not have clustered indexes, use calculated columns, operators and listing of proper columns in queries, however, the use of indexes should be restricted because they affect insert, update, and delete operations.

Keywords: relational database, database optimization, SQL, indexing

I. Introducción

Las Bases de Datos se han convertido en un elemento primordial en el ámbito empresarial, ya que almacenan la información relacionada con el proceso básico de la organización, y para poder cumplir correctamente su función se deben asegurar ciertas características como son: la escalabilidad, seguridad de la información y tener un buen rendimiento [1].

Se puede medir el rendimiento de una base de datos en términos de tiempo, haciendo referencia al tiempo que toma al servidor de base de datos responder una solicitud o requerimiento de información solicitada por un cliente [2]. En tal sentido, es crucial para la empresa que el tiempo de respuesta a los datos que se requieren de la base de datos sea el mínimo posible para que puedan atender solicitudes en menor tiempo lo que se traduce en mayor cantidad de clientes atendidos, y que tiene un impacto directo en la economía de la empresa.

El diseño de la base de datos obedece esencialmente a que se debe asegurar el almacenamiento y recuperación de todos los datos que la empresa requiere para la automatización eficiente del proceso elegido [3].

Existen cuatro operaciones que debe realizar el servidor de base de datos: registro, actualización, eliminación y consulta de datos; siendo la última operación la más utilizada [4]. La operación de consulta de datos es potencialmente más utilizada en aquellos sistemas informáticos que trabajan con datos confidenciales, como son sistemas para entidades financieras, sistemas para el área de salud y sistemas para identificación de personas que manejan miles de transacciones de datos al día" [5].

de consultas SQL es rápido, pero cuando la base de datos lleva un tiempo funcionando y la cantidad de registros almacenados es bastante grande, llegando a tener en una tabla hasta millones de registros, el tiempo de acceso es bastante lento lo que hace que el sistema informático tarde varios segundos en conseguir el dato requerido. Las compañías tienen como requerimiento esencial para los servidores de bases de datos relacionales la obtención de nuevos datos en milisegundos; requerimiento que muchas veces no se alcanza por limitaciones de hardware o la escasa preparación del personal responsable del servidor.

Es en este punto, donde juega una vital importancia utilizar técnicas de optimización de base de datos que permitan agilizar las consultas de datos. La optimización de base de datos puede ser abordada de dos diferentes maneras: basada en el diseño físico y basada en parámetros de configuración [5]. La primera alude a la forma en cómo se realiza el diseño de la base de datos encontrando en la selección eficiente de índices un indicador de optimización, la segunda busca elegir los parámetros de configuración apropiados como son la asignación de memoria y la administración de procesamiento [5]. También es importante abordar la optimización de consultas que permitan ahorrar costos de procesamiento [6]. De esta manera es necesario diseñar consultas SQL que devuelvan datos específicos y que realicen la menor cantidad de operaciones para que puedan tener un tiempo de respuesta menor. Se debe tener en cuenta que las consultas escritas de forma deficiente, degradan el rendimiento de la base de datos, por ello, actualmente se aplican algoritmos de búsqueda aleatoria como el algoritmo de colonia de hormigas y el algoritmo de enjambre de partículas que han permitido obtener resultados favorables para ciertos contextos específicos como es el problema de la optimización de consultas de base de datos para sistemas integrados y sistema de búsquedas de palabras clave [7] [8].

El proyecto de investigación tiene como objetivo determinar buenas prácticas para la Optimización de Base de Datos utilizando Microsoft SQL Server, planteándose como hipótesis que, si se ejecutan estas buenas prácticas, los tiempos de respuestas de las consultas realizadas a la base de datos serán menores.

II. Desarrollo

Se han abordado técnicas y recomendaciones relacionadas a la creación de base de datos y a las sentencias de consulta SQL que permitirán mejorar el tiempo de respuesta de la base de datos.

A. Buenas prácticas propuestas para la optimización de Base de datos:

Índices en la creación de tablas

Los índices asignados que se pueden crear en una tabla pueden ser de dos tipos: agrupados (clustered) y no agrupados (nonclustered) que permitan reducir tiempos de espera al momento de realizar una consulta. [9]. Se debe tener en cuenta que Microsoft SQL Server permite tener solo un índice agrupado por tabla, por tanto, los demás índices que se creen deberán ser de tipo no agrupados. Cuando se crea una clave primaria en una tabla, por defecto SQL Server crea un índice agrupado [10]. Sin embargo, cuando creamos claves foráneas, por defecto SQL Server no crea ningún índice.

Si se analiza el contexto, las claves primarias no necesariamente son aquellas que se utilizan en los ordenamientos de las búsquedas, como por ejemplo en el caso de la tabla Cliente, no se busca por su código sino por Apellidos y Nombres, siendo en estos casos más útil crear un índice agrupado compuesto por apellidos y nombres. En resumen, el índice agrupado se debe de crear en las columnas de aquellas tablas que son frecuentemente utilizadas en las consultas de búsqueda, en ordenamientos y, en último caso, utilizadas en comparaciones continuas.

Con respecto a los índices no agrupados, aunque también ayudan a disminuir tiempos de respuestas, son menos eficientes, y teniendo en cuenta que una tabla solo puede tener un índice agrupado, se deben crear índices no agrupados en las columnas que se utilizan para realizar comparaciones, como por ejemplo en claves primarias y claves foráneas si previamente no se les ha generado un índice agrupado.

Por otro lado, es importante no abusar de la creación de índices porque afectan negativamente las operaciones de inserción, actualización y eliminación e incrementa el tamaño de base de datos y por lo tanto de la copia de seguridad [11], sin embargo, mejoran el tiempo de respuesta de las consultas con reuniones y agilizan las comparaciones.

Columnas calculadas

Las columnas calculadas vienen a ser un campo dentro de una tabla que guardan resultados de operaciones matemáticas (precálculos) con otros campos [12], lo que beneficia el rendimiento ya que evita realizar operaciones para obtener dichos valores.
La utilización de columnas calculadas acorta los tiempos de respuesta de las consultas, aunque afecta negativamente las operaciones de inserción y actualización de los datos involucrados debido al cálculo necesario para mantener la coherencia de los datos.

Operadores adecuados y listado de columnas en consultas realizadas

Es importante elegir cuidadosamente los operadores que se utilizan en las consultas porque dependiendo de la estructura de la base de datos y en especial de sus índices los tiempos de respuesta pueden variar considerablemente. Es necesario, según sea el caso, elegir el operador adecuado cuando existen operador que realizan funciones similares como es el caso del operador EXISTS/NOT EXISTS y el operador IN/NOT IN. En las cláusulas GROUP BY y ORDER BY de ser posible se debe utilizar columnas y no operaciones para permitir al optimizador de consultas del servidor de base de datos elegir los índices apropiados en beneficio de un mejor rendimiento. Así mismo, evitar utilizar el operador *

*Í*cuando se realiza la sentencia SELECT y de preferencia listar campos específicos que se desean; aunque esto parece ser algo obvio es una mala práctica muy difundida en los programadores.

B. Antecedentes de la Investigación

La optimización de bases de datos es en la actualidad una necesidad imperativa para las empresas; la cuales utilizan diferentes enfoques para obtener tiempos mínimos en la ejecución de órdenes SQL. Aunque al inicio de las operaciones esta necesidad pasa desapercibida e incluso abordada de forma descuidada por los responsables del diseño e implementación de las bases de datos; con el correr del tiempo y con el crecimiento de los datos almacenados, se puede comprobar un incremento preocupante en los tiempos de respuesta [13].

Hoy en día a nivel de hardware, los servidores son muy avanzados teniendo como prioridad su capacidad de procesamiento, enfatizando en cada uno de los aspectos relacionados: velocidad y cantidad de procesadores, velocidad y cantidad de RAM así como velocidad de acceso a las unidades de almacenamiento entre otros. Incluso se dispone de herramientas como RAID de discos duros o arquitecturas complejas para implementar servidores distribuidos para obtener mejor rendimiento. De tal manera que al mejorar la capacidad de procesamiento de los servidores mejoran los tiempos de respuesta de las consultas a la base de datos.

Hay un incremento de investigaciones en entornos distribuidos para la optimización de base de datos. En entornos de múltiples procesadores se aplican la optimización de consultas paralelas en el servidor OLAP (aplicación de soporte de decisiones) para encontrar un plan paralelo que entregue el resultado de la consulta en un tiempo mínimo [14].

En la optimización de consultas se están tendiendo a usar algoritmos probabilísticos para mejorar la velocidad en la obtención de resultados. Un algoritmo usado es el SDD-1 y junto con el método Semi- JOIN permite un incremento en la velocidad de la consulta de hasta un 300% [4]. Otro algoritmo usado es el algoritmo de Ascensión de colina que junto con una adecuada indexación y métodos de optimización de consultas en subprocesos múltiples se reduce el tiempo de consulta en 45% en comparación con otros métodos tradicionales [15].

Las empresas propietarias de los sistemas gestores de bases de datos relacionales para afrontar la lentitud de sus operaciones proporcionan una herramienta dedicada a generar tiempos de respuestas mínimos llamada Motor de Optimización de Consultas. Este motor utiliza como insumos la metadata de las tablas (índices, tipos de índices creados, columnas clave de los índices, archivos y ubicación de los mismos) así como de las estadísticas almacenadas de cada una de ellas para elegir la ruta de ejecución más eficiente, la cual se denomina Plan de Ejecución [16].

Las soluciones basadas en hardware permiten aumentar la capacidad de procesamiento, pero involucran un alto coste en la compra de equipos apropiados y contratar personal altamente capacitado en nuevas técnicas de ejecución de consultas a la base de datos.

Sin embargo, una solución con menor costo sería la optimización de consultas a la base de datos. Los optimizadores de consultas utilizan las características del Álgebra Relacional para reordenar los operadores (selección, proyección, unión, reuniones, entre otros) con el objetivo de disminuir la cantidad de datos involucrados en la implementación de las operaciones solicitadas; en especial la solicitud de datos (operación más utilizada por los sistemas). Disminuir la cantidad de datos es importante porque reduce los accesos a la base de datos y necesita menor velocidad de trasmisión de datos.

Existe en el mercado software licenciado que permite optimizar base de datos; como son SQL Developer que "analiza el rendimiento de una sentencia SQL para mostrar un plan de ejecución con información estimada y real" [17]. TOAD es otra herramienta que sirve para analizar el rendimiento de una consulta que se base en el plan de ejecución para trabajar parámetros sin estimación que luego serán visualizados por el usuario. [17].

A través del uso de herramientas informáticas para optimizar base de datos podemos obtener un plan de ejecución que conllevaría a una reducción en el tiempo de la consulta; sin embargo, hace uso intensivo del CPU por lo que si no se cuenta con hardware adecuado puede conllevar a un incremento del tiempo transcurrido; por lo que el uso permanente de estas herramientas en servidores de producción no es recomendado.

Otra opción para mejorar los tiempos de respuesta es utilizar la técnica de optimización basada en costo consistente en utilizar las estadísticas de los datos almacenados en las tablas o índices creados [18]. Ambos métodos pueden combinarse para generar mejores resultados.

Aunque los motores de optimización estén implementados con los mejores algoritmos finalmente son dependientes de las personas que diseñan las bases de datos y las que diseñan las consultas. Es importante por lo tanto utilizar buenas prácticas para el diseño físico de base de datos, así como la correcta utilización de los operadores de una consulta explotando de esta manera las características propias de cada servidor [19].

III. METODOLOGÍA

Con el objetivo de mostrar las mejoras en el rendimiento de las operaciones de consulta se han creado dos bases de datos de ventas; una que actuará como base de datos experimental a quien se le aplicarán las técnicas de optimización de base de datos y otra que actuará como base de datos control. La data utilizada para realizar el llenado de las bases de datos es simulada y se ha generado mediante script. Ambas bases de datos tienen la misma estructura, contienen doce tablas que se indican a continuación:

Tabla 1. Tablas de la Base de Datos de Venta

Nº	Tabla	Cantidad de filas	Método de inserción
1	Categoría	5	Manual
2	Marca	27	Manual
3	Producto	31	Manual
4	Modelo	119	Manual
5	Cliente	4211	Manual
6	Trabajador	122	Manual
7	Venta	10.077.395	Script
8	Detalle Venta	26.434.060	Script
9	Cuota	15.635.369	Script
10	Pago	2.748.547	Script
11	Cuota Pago	2.748.547	Script
12	Usuario	70	Script

Nota: Cantidad de filas (registros) de cada tabla de la base de datos de Venta, las tablas que tienen millones de registros fueron insertados por medio de script

Con el objetivo de obtener diferentes mediciones de las consultas, éstas se ejecutaron en computadores con diferentes configuraciones, las cuales se detallan a continuación:

Equipo 01, Tipo: Laptop, Procesador: Core i5 6200U 2.30 GHz, RAM: 8GB, Disco duro: 1TB 5400 rpm, Sistema operativo: Windows 10 Pro x64.

Equipo 02, Tipo: Laptop, Procesador: Core i5 2.20 GHz, RAM: 6GB, Disco duro: 1TB 5400 rpm, Sistema operativo: Windows 8.1 Pro.

Equipo 03, Tipo: Desktop, Procesador: Core i5 2320 3.30 GHz, RAM: 8GB, Disco duro: 1TB 5400 rpm, Sistema Operativo: Windows 10 Enterprise x64.

En las tres computadoras se han creado las dos bases de datos: base de datos con valores de configuración por defecto, y la base de datos optimizada con buenas prácticas.

Luego se diseñaron 10 consultas que permitieran acceder a las tablas que tenían mayor cantidad de registros, las cuales se detallan a continuación:

•Listado de los clientes morosos
•Listado de los modelos de productos más vendidos durante el año 2016
•Listado de los 10 trabajadores que más han vendido en el último trimestre del año 2016
•Estado de cuenta de un cliente
•Listado de los modelos de productos que no se han vendido durante el año 2017
•Listado de ingresos por trabajador en el año 2015
•Indicar los clientes que deben pagar detallando el monto para un mes y año determinado
•Listado de los trabajadores y el número de documentos anulados por mes para el año 2016.
•Cuáles son los trabajadores que no han realizado ventas en el primer trimestre del año 2017.
•Evolución de ingresos por medio en los años 2011 hasta el 2016

Estas consultas fueron ejecutadas con la finalidad de medir el tiempo de respuesta de la base de datos por defecto (BD control) y base de datos optimizada (BD experimental) y posteriormente poder compararlos.

En cada computadora se han ejecutado estas diez consultas a cada base de datos: experimental y control, realizándose la ejecución 4 veces simultáneas y obteniendo un promedio de estas ejecuciones simultáneas por computadora. Posteriormente se han promediado los resultados de las tres computadoras, obteniéndose un solo valor (promedio) por cada consulta.

IV. Resultados

Para poder corroborar la optimización de una base de datos, se han aplicado buenas prácticas solo a la Base de Datos optimizada (experimental), que se detallan a continuación:

Índices en la creación de tablas
En la base de datos optimizada (base de datos experimental) se han creado índices agrupados en las claves foráneas de las tablas: Cuota, CuotaPago, Pago, DetalleVenta y Modelo.

También se ha creado índices agrupados en columnas que son frecuentemente buscadas pero que no son claves foráneas ni claves primarias, en las tablas de: Categoría, Cliente, Producto, Trabajador. En la tabla Categoría se ha

colocado índice agrupado a la columna nombre, en la tabla Cliente un índice compuesto por apellidos y nombres (en ese orden), en la tabla Producto a la columna nombre y en la tabla Trabajador a la columna compuesta por apellidos y nombres.
Por último, se ha creado índice agrupado a la clave primaria de la columna código en la tabla Venta, porque cuando se quiere realizar una búsqueda de un pago o de las cuotas de un pago, el criterio de búsqueda principal es el código de venta, siendo un caso excepcional que obedece a que se debe declarar como índice agrupado aquellas columnas que se usen frecuentemente en la búsqueda y ordenamiento de datos.

Con respecto, a los índices no agrupados, se han asignado a las claves primarias de las siguientes tablas: Categoría, Cliente, Cuota, CuotaPago, DetalleVenta, Modelo, Pago, Producto, Trabajador. Adicionalmente se han creado más de un índice no agrupado en una misma tabla en: DetalleVenta, Modelo, Pago, Producto y Venta.

La tabla Venta por tener campos que se consultan frecuentemente, tiene una configuración diferente a las demás tablas, asignándosele índice agrupado a la clave primaria (código) y asignándose dos índices no agrupados a las claves foráneas (Codigo de Cliente y Codigo de Trabajador).

Columnas calculadas
Se ha creado una columna denominada Total en la tabla Venta. Esta columna contiene el pago total del documento de venta; es decir la sumatoria de la multiplicación del precio de venta y la cantidad vendida de cada producto.

Operadores adecuados y listado de columnas en consultas realizadas
Con el objetivo de mejorar el rendimiento de las consultas a la base de datos se ha utilizado el operador EXISTS/NOT EXISTS en vez del operador IN/NOT IN. En la cláusula ORDER BY se ha optado por utilizar columnas en lugar de operaciones (concatenaciones). Este cambio favorece la utilización de los índices. En la cláusula SELECT se ha listado las columnas de una determinada tabla en reemplazo del operador *.

Posteriormente se han medido los tiempos de respuesta de la ejecución de cada una de las diez consultas a la BD control y a la BD experimental, conforme se ha detallado en la metodología.

Para poder hacer las comparaciones respectivas se ha creado una tabla donde se agrupan las consultas en: Categoría 1 (Consultas con mejores tiempos en BD Optimizada) y Categoría 2 (Consultas con mejores tiempos en BD por defecto).

Tabla 2. Promedios de tiempos de ejecución en milisegundos de las diez consultas agrupados en dos categorías, en la BD con configuración por defecto (control) y la BD optimizada (experimental)

Categoría	Nº de Consulta	BD con configuración por defecto (control)	BD optimizado (experimental)
Categoría I	1	116.109,75 ms.	71.818,00 ms.
	2	129.897,67 ms.	28.297,67 ms.
	4	2.834,75 ms.	2.605,50 ms.
	5	1.929.282,50 ms.	42.381,33 ms.
	6	2.528,25 ms.	650,58 ms.
	9	79.717,42 ms.	76.300,33 ms.
	10	361,92 ms.	348,00 ms.
Promedio		**322.961,75 ms.**	**31.771,63 ms.**
Categoría II	3	1.466,25 ms.	1.946.17 ms.
	7	4.819,58 ms.	8.348,58 ms.
	8	300,75 ms.	1.483,33 ms.
Promedio		**2.195,53 ms.**	**3.926,03 ms.**

Nota: En la Categoría I se ubican las consultas que tuvieron menor tiempo de respuesta en la base de datos experimental y en la categoría II se ubican las consultas que tuvieron mejor tiempo de respuesta en la base de datos control.

En la tabla anterior se puede apreciar que en la Categoría II, se ha obtenido una situación desfavorable para la investigación ya que la base de datos con configuración por defecto es la que ha obtenido mejor promedio que la versión optimizada y esto se debe a qué en las consultas de la Categoría II (consultas 03, 07 y 08) se realiza procesamiento previo en las columnas de tipo fecha, por ejemplo, obtener el trimestre en un año específico, y las columnas de fechas no contienen ningún índice.

Posteriormente, se realizó el cálculo de las medidas de dispersión (x), la desviación estándar (s) y del coeficiente de variación (CV) de las dos bases en cada una de las categorías.

Tabla 3. Medidas de dispersión de tiempos de ejecución de las diez consultas agrupados en dos categorías, en la BD con configuración por defecto (control) y la BD optimizada (experimental)

		BD con configuración por defecto (control)	BD optimizado (experimental)
Categoría I	x	322.961,75 ms.	31.771,63 ms.
	s	710.484,02 ms.	32.959,01 ms.
	CV	2,1999	1,0374
Categoría II	x	2.195,53 ms.	3.926,03 ms.
	s	2.346,03 ms.	3.837,03 ms.
	CV	1,0686	0,9773

Nota: Medidas estadísticas promedio, desviación estándar y coeficiente de variación de los tiempos obtenidos en la base de datos control y en la experimental.

El coeficiente de variación es menor en la base de datos optimizada que en la base de datos con configuración por defecto, en tal sentido podemos establecer que los tiempos de ejecución en la base de datos optimizada tuvieron valores más homogéneos que los de la base de datos con la configuración por defecto.

Conclusiones

La creación de la base de datos debe implicar no sólo la satisfacción de los requerimientos funcionales que darán soporte al sistema sino también la utilización de buenas prácticas que permitan afrontar los desafíos de tiempo de respuesta cuando la base de datos crezca y llegue a tener millones de registros.

Se llega a la conclusión que las buenas prácticas para optimizar una base de datos basados en software se centran en: crear índices agrupados en columnas que se utilicen con mayor frecuencia para realizar ordenamientos, búsquedas o muchas comparaciones; crear índices no agrupados en columnas que se utilicen para hacer comparaciones y que no se les haya asignado anteriormente índices agrupados, en tal sentido; en tablas que tengan claves foráneas a las que alguna de ellas se les ha asignado índice agrupado, la clave primaria será asignada como índice no agrupado; en la tabla que tenga claves foráneas pero que alguna otra columna haya sido asignado como índice agrupado, tanto clave foránea como clave primaria serán índices no agrupados; en las tablas que no tengan clave foránea la clave primaria será asignada como índice agrupado; utilizar columnas calculadas que tengan registros resultantes de operaciones de otros campos y que son frecuentemente buscados y utilizar operadores y listado de columnas adecuadas en una consulta, todo ello, ayuda a disminuir los tiempos de respuestas de las consultas, sin embargo, también se debe hacer uso restringido de la creación de índices porque afectan a las operaciones de inserción, actualización y eliminación en la base de datos, ya que de las 10 consultas ejecutadas tanto, en la base de datos experimental optimizada con las buenas prácticas antes mencionadas, como en la base de datos con configuración por defecto, se obtuvo que siete

consultas demostraron resultados favorables en cuanto a menor tiempo de respuesta y tres consultas mostraron resultados desfavorables porque obtuvieron mayor tiempo de respuesta, esto se puede atribuir a que las consultas que obtuvieron resultados desfavorables realizan procesamiento previo en las columnas, por ejemplo un trimestre en un año específico, y las columnas de fechas no contienen ningún índice. Por último, se debe realizar uso restringido en la creación de índices tanto agrupados como no agrupados, ya que afectan el rendimiento de las operaciones de inserción, actualización y eliminación dentro de una tabla de la base de datos.

Agradecimientos

El presente proyecto fue realizado con el apoyo del Bach. Walter Ventura Chozo, Responsable del Laboratorio de Cómputo de la Escuela Profesional de Ingeniería en Computación e Informática; quién nos facilitó el acceso a los laboratorios para ejecutar las pruebas diseñadas a las bases de datos utilizadas.

Limitaciones

Se ha realizado la ejecución de consultas en tres computadoras con diferentes características lo cual puede sesgar el resultado final, atenuando este sesgo con la ejecución consecutiva de hasta en 04 (cuatro) oportunidades, obteniendo un promedio de los tiempos de respuesta por cada consulta.

Referencias

[1] C. M. Ricardo, Bases de datos, Segunda ed., Mexico: McGraw-Hill, 2009.
[2] R. Elmasri y S. B. Navathe, Sistemas de Bases de Datos: Conceptos Fundamentales, Segunda ed., España: Addison-Wesley Iberoamericana, 1997.
[3]Microsoft, "SQL Server Database Engine," 2012. [Online]. Available: https://msdn.microsoft.com/es-es/library/ms187875(v=sql.110).aspx. [Accessed Agosto 2015].
[4] A. Zulfikar, H. Spits Warnas, F. Gaol, E. Abdurachman y B. Soewito, «La optimización de consultas para bases de datos distribuidas utiliza un enfoque basado en semiuniones (SBA) con el algoritmo SDD-1,» Conferencia internacional sobre información de 2019 and Communications Technology (ICOIACT), pp. 619-623, 2019.
[5] A. Samson y A. Aponso, «An Analysis on Automatic Performance Optimization in Database Management Systems,» 2020 World Conference on Computing and Communication Technologies (WCCCT), pp. 6-9, 2020.
[6] J. Murlewski, T. Kowalski, R. Adamus, B. Sakowicz y A. Napieralski, «Optimización de consultas en bases de datos de cuadrícula,» 14th International Conference on Mixed Design of Integrated Circuits and Systems, pp. 707-710, 2007.
[7] J. Xu, «Análisis de optimización dinámica de los resultados de consultas de palabras clave en bases de datos relacionales basadas en el algoritmo de optimización de colonias de hormigas,» Conferencia internacional sobre tecnología informática, electrónica y comunicación (ICCTEC), pp. 721-724, 2017.
[8] X. Mingyao y l. Xiongfei, «Algoritmo de optimización de consultas de bases de datos integradas basado en optimización de enjambre de partículas,» Séptima Conferencia internacional sobre tecnología de medición y automatización mecatrónica, pp. 429-432, 2015.
[9] J. L. Jorden y D. Weyn, MCTS Microsoft SQL Server 2005 Implementation and Maintenance Study Guide, EEUU: Wiley Publishing, 2006.
[10] P. De Betta, Introducing SQL Server 2008, EEUU: Microsoft Press, 2008.
[11] R. Mistry y S. Misner, Introducing Microsoft SQL Server 2012, EEUU: Microsoft Press, 2012.
[12] O. Thomas y I. McLean, Training Kit Optimización y Mantenimiento de una Solución de Administración de Base de Datos Microsoft SQL Server 2005, Madrid: Anaya Multimedia, 2006.
[13] M. G. Varas Beltrán, «Creación de un modelo de optimización para los Query utilizando la sentencia SELECT de SQL,» Quevedo, 2014.
[14] S. Deepak, S. Kumar, M. Durgesh y PBK, «Procesamiento de consultas y optimización del sistema de bases de datos paralelas en entornos de múltiples procesadores,» Sexto Simposio de Modelado de Asia de 2012, pp. 191-194, 2012.

[15] X. Sun, B. Jiang y X. He, «Optimización de consulta de base de datos basada en generación de energía fotovoltaica distribuida,» 2nd IEEE Advanced Information Management, Communications, Electronic and Automation Control Conference (IMCEC), pp. 2382-2386, 2018.
[16] J. Ruiz Rangel, «Procesamiento y optimización de consultas,» Journal of Engineering and Technology, vol. 3, nº 2, pp. 36-43, 2014.
[17] R. García Frutos, «Optimización de consultas en bases de datos relacionales,» Madrid, 2016.
[18] E. Ramas Ferrández, «Optimización de consultas a bases de datos relacionales,» Zaragoza, 2017.
[19] P. Quiñonez Villa, «Implementacion de una base de datos SQL Server,» 2013. [En línea]. Available: http://es.slideshare.net/paulquinonez3/implementacion-de-una-base-de-datos. [Último acceso: Agosto 2016].

Los Autores

Gisella Luisa Elena Maquen Niño
Doctora en Ciencias de la Educación con mención en Administración de la Educación. Estudios de doctorado en Ciencias de la Computación y Sistemas, Universidad Señor de Sipán. Ingeniero en Computación e Informática, Universidad Pedro Ruiz Gallo. Filiación: Universidad Pedro Ruiz Gallo – Perú. E-mail: gmaquenn@unprg.du.pe, ORCID: https://orcid.org/0000-0002-9224-5456 .

Franklin Edinson Terán Santa Cruz
Maestro en Ingeniería de Sistemas con mención en Gerencia de tecnologías de la información y gestión del software. Ingeniero en Computación e Informática, Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú. Filiación: Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú. E-mail: fteran@unprg.edu.pe , ORCID: https://orcid.org/0000-0002-3197-7979.

Consuelo Ivonne Del Castillo Castro
Doctora en Educación, Magister en Ingeniería de Sistemas con mención en Gerencia de Tecnologías de la Información, Ingeniera en Computación e Informática. Microsoft Certified Professional. Docente adscrito al Departamento Académico de Computación y Electrónica de la Universidad Nacional Pedro Ruiz Gallo de Lambayeque - Perú.. E-mail: cdelcastilloc@unprg.edu.pe, ORCID: https://orcid.org/0000-0002-1512-006X

Rafael Damián Villón Prieto
Ingeniero de Sistemas, Licenciada en Educación Secundaria: Matemática, Computación E Informática, Maestro en Gestión Pública, Doctor en Gestión Pública y Gobernabilidad. Docente contratado en la Universidad Cesar Vallejo SAC en la escuela de Posgrado de Lambayeque – Perú. E-mail: villonpr@ucvvirtual.edu.pe, ORCID: https://orcid.org/0000-0002-5248-4858