Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

Análisis de la optimización de sentencias SQL del Lenguaje de Consulta Estructurado usando grandes volúmenes de datos

Ariosto Eugenio Vicuña Pino[1], Jessica Alexandra Ponce Ordoñez[2], Orlando Ramiro Erazo Moreta[3]

1 Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador, avicuna@uteq.edu.ec

2 Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador, jessica_ponceord@hotmail.es

3 Universidad Técnica Estatal de Quevedo, Quevedo, Ecuador, oerazo@uteq.edu.ec

RESUMEN

Este trabajo examina la importancia de la optimización de sentencias del Lenguaje de Consulta Estructurado (SQL) bajo la condición de operar con millones de datos. Se pretende dilucidar si una sola etapa de optimización es suficiente para mejorar el tiempo de respuesta ante el cambio del volumen de datos. Para el efecto, se utiliza la estructura de una base de datos real y en servicio, que se pobló con uno y cinco millones de registros. Después, se plantearon consultas complejas a ser resueltas por el script que contenía sentencias SELECT y medir el tiempo de respuesta de su ejecución. Luego de optimizar dos veces las sentencias se encontró que es posible reducir el tiempo de respuesta en t/10 segundos. Estos resultados confirman que las sentencias SQL complejas deben ser optimizadas de acuerdo al gestor de bases de datos donde se ejecutan para los diferentes rangos de millones de registros.

Palabras claves: Optimización, Base de Datos, SQL, tiempo de respuesta.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

Analysis of the optimization of SQL statements of the Structured Query Language using large volumes of data

ABSTRACT

This work examines the importance of the optimization of statements of the Structured Query Language (SQL) under the condition of operating with millions of data records. It is intended to elucidate whether a single optimization stage is enough when changing the volume of data. For this purpose, the structure of a real and in-service database was used, which was populated with one and five million records. Then, complex queries were raised to be resolved by the script that contained SELECT statements and measure the response time of their execution. After optimizing the sentences twice, it was found that it is possible to reduce the response time in t/10 seconds. These results confirm that complex SQL statements must be optimized according to the database manager where they are run for the different ranges of millions of records.

Keywords: Optimization, Databases, SQL, response time.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

INTRODUCCIÓN

Las tecnologías de la información han desarrollado capacidades de conservación y obtención de información que facilitan las operaciones transaccionales y de toma de decisiones. Éstas contienen gestores de bases de datos que implementen un lenguaje de consultas de alto nivel proporcionando un fácil y rápido acceso a la recuperación o actualización de los datos almacenándolos en los principales "buffers" de memorias especiales (Dhande & Bamnote, 2014).

El Lenguaje de Consultas Estructurado (SQL), cuando las sentencias que provee son utilizas de manera eficiente, permite la creación de sistemas altamente escalables, flexibles y manejables. Por eso, la utilización de las consultas especialmente, debe ser hecha con eficiencia, ya que se pueden clasificar 100.000 filas de cuatro bytes en aproximadamente un segundo o sólo 10.000 filas de cincuenta bytes en el mismo tiempo (Larsen, 2005). La correcta operación de las sentencias SELECT permite que el acceso a los datos para consultar sea eficiente. Estas búsquedas son realizadas por medio de algoritmos que tienen en cuenta las consultas que se realizan a los registros, que permiten crear un vector expresión para cada consulta (Singh & Varshney, 2013).

La eficiencia tiene que ver preferentemente con el problema de los tiempos de respuesta en un Sistema Gestor de Base de Datos (DBMS, por sus siglas en inglés) que se da principalmente porque el uso de la sentencia SELECT podría estar consumiendo recursos de manera considerable, lo que conlleva percibir una lentitud en el sistema. Cabe entonces intuir que la incorrecta utilización de las sentencias SQL, y la sentencia SELECT en particular, es "la razón principal por la cual una consulta no puede resolverse en máximo 3 segundos" (Salazar, 2015).

Teniendo en cuenta estos aspectos, este trabajo presenta los problemas de rendimiento en la ejecución de sentencias SQL cuando éstas no se someten a un proceso de optimización adecuado. Para el efecto se utilizaron varios DBMS, evidenciando algunos aspectos importantes tales como la dependencia de la depuración de la sentencia SQL y el DBMS donde se ejecuta. Estos resultados deberían servir de referencia para que los administradores de bases de datos optimicen de mejor manera las respectivas bases de datos, de acuerdo los DBMS empleados.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

MATERIALES Y MÉTODOS

Datos

El estudio se llevó a cabo basándose en la estructura de una base de datos empleada dentro de un contexto real. El modelo lógico de la base de datos seleccionada (Figura 1) corresponde a la utilizada en el catastro urbano y rural de una municipalidad de Ecuador. Teniendo en cuenta que esta base de datos ha sido cuidadosamente diseñada y evaluada previamente como parte del sistema de catastros municipal, no se realizó ningún análisis previo de dicho diseño antes de cargar los datos experimentales. Los registros contenidos en esta base de datos fueron generados de forma automática mediante procedimientos almacenados. De hecho, la base de datos se pobló con un millón de registros por tabla en primera instancia y posteriormente con cinco millones.

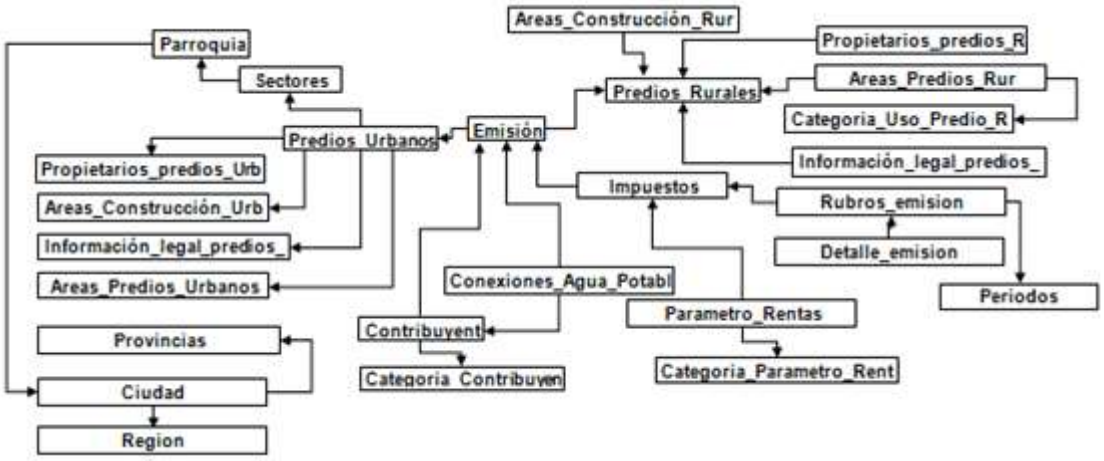

Figura 1. Modelo lógico de la base de datos utilizada

Equipos informáticos y software

La base de datos de catastro, los datos generados y las sentencias SQL para el estudio se cargaron en un computador con procesador AMD Athlon XP con 512 MB de memoria RAM. Se utilizó este equipo de bajas prestaciones con la finalidad de evidenciar con mayor detalle cuál es el efecto en el tiempo de respuesta de las optimizaciones de las sentencias SQL. El equipo trabajó en forma dedicada mientras ejecutaba las sentencias SQL para lo cual se desactivó cualquier otro software que pudiera consumir recursos. Finalmente, después de ejecutar todo el conjunto de sentencias SQL con un DBMS, y previo a la instalación del siguiente DBMS, se procedió a resetear el equipo dejándolo en el mismo estado inicial. Los DBMS que se usaron fueron PostgreSQL 9.0, Oracle

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

10G y MS-SQL Server 2012, los mismos que se escogieron por estar entre los más usados a nivel mundial.

Procedimiento y diseño

Con el fin de analizar el efecto de la optimización de las sentencias SQL, se definieron diez sentencias complejas para cada una de las cuatro operaciones básicas (insertar, modificar, eliminar y seleccionar) sobre la base de datos de Catastro. Estas sentencias se ejecutaron diez veces en cada uno de los gestores de base de datos. Los tiempos de respuesta fueron obtenidos desde la consola del analizador de consultas en cada ejecución. Posteriormente se calculó el promedio del tiempo de respuesta de las diez ejecuciones de cada sentencia y este valor promedio fue utilizado para la prueba de hipótesis. Este procedimiento se aplicó tanto para sentencias sin optimizar (SOPT) como para sentencias con una (1OPT) y dos (2OPT) etapas de optimización. Para optimizar las sentencias SQL se trabajó sobre la base de datos en PostgreSQL. Las sentencias optimizadas, bajo el mismo esquema experimental, también fueron ejecutas en las bases de datos de Oracle y SQL Server. Las optimizaciones fueron realizadas fundamentalmente en base a lo planteado por (Rahman, Feroz, Kamruzzaman, & Faruque, 2010; Geng & Zhang, 2009; Al Mamun & Kabir, 2008).

Se utilizó el análisis de varianza de un factor (ANOVA) para determinar la existencia de diferencias significativas entre las medias de los tiempos de respuesta. Posteriormente se aplicó la prueba de rangos múltiples de diferencia honestamente significativa (HSD) de Tukey con la cual se determinó el mejor tiempo de respuesta de acuerdo a la optimización realizada.

RESULTADOS

El tiempo *t*(s) de ejecución de las sentencias SQL sin optimizar (SOPT) se logró disminuir a *t*/10 (s) mediante las dos etapas de optimización. No se evidenció una mejora significativa del tiempo de respuesta en las pruebas realizadas entre la primera optimización (1OPT) y segunda optimización (2OPT). La figura 2 muestra los resultados de la ejecución de las sentencias SQL sin optimizar y con dos etapas de optimización para uno y cinco millones de registros.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

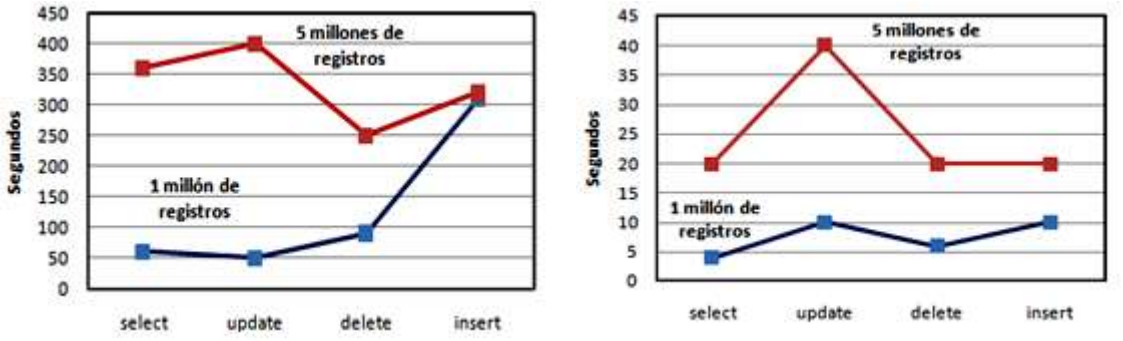

Figura 2. Umbrales de los tiempos de respuesta de las sentencias SQL sin optimizar y con dos etapas de optimización

Con los DBMS poblados con un millón de registros se ejecutaron las sentencias SQL. En PostgreSQL 9.0 las sentencias SQL sin optimizar (SOPT) presentaron una media de 300 segundos. Éstas al pasar por 1OPT disminuyeron sus los tiempos de respuesta a una media de 10 segundos y en la 2OPT a 5 segundos. En Oracle 10G se alcanzó una media de 40 segundos para SOPT. Esto se redujo significativamente con 1OPT donde se observa una media de 8 segundos y 6 segundos para 2OPT. En SQL Server 2012 para las sentencias SQL SOPT se obtuvo una media de 4 segundos. Por otro lado, 1OPT alcanzó una media de 3.5 segundos y 12 segundos para 2OPT. Esto se debe a que las optimizaciones de las sentencias SQL se hicieron en el DBMS PostgreSQL 9.0 y fueron ejecutadas tanto en Oracle 10G como en SQL Server 2012 tuvo un comportamiento diferente. Estos resultados pueden observarse en la Figura 3.

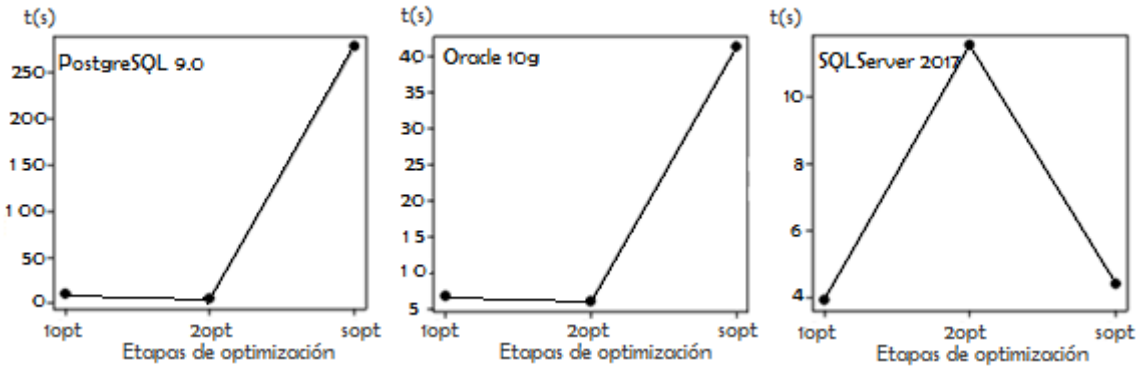

Figura 3. Gráfico de medias en DBMS con 1 millón de datos

Las sentencias SQL se ejecutaron en cada DBMS con cinco millones de registros. Para PostgreSQL 9.0 las SOPT mostraron una media de 225 segundos; con la 1OPT presentó una media de 35 segundos y, mediante la 2OPT, la media disminuyó a 20 segundos. En Oracle 10G las sentencias

Received 17/05/2018
Approved 05/07/2018

SQL SOPT retornaron un tiempo de respuesta de 175 segundos, mientras que para 1OPT se tuvo una media de 90 segundos y la segunda optimización resultó con una media de 70 segundos.

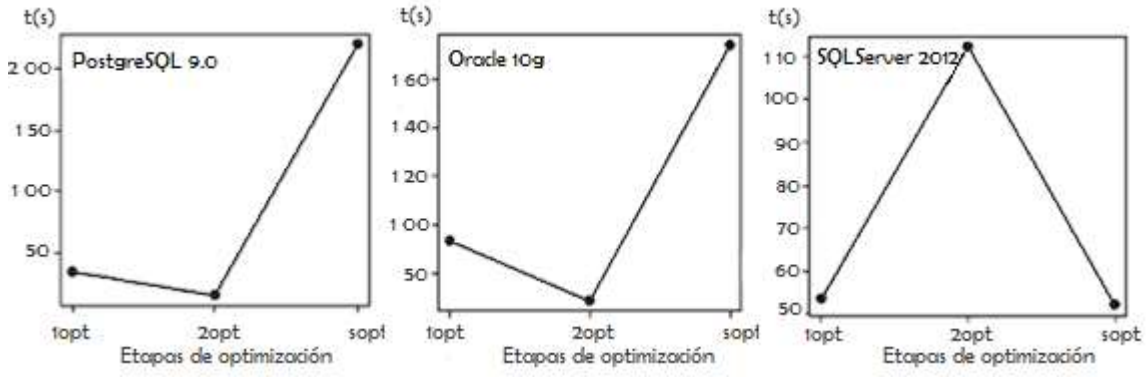

Figura 4. Gráfico de medias en DBMS con 5 millones de datos

La ejecución de las pruebas en SQL Server 2012 mostraron un tiempo similar en las sentencias SOPT Y 1OPT con una media de 55 segundos, mientras que en 2OPT la media es de 120 segundos. La figura 4 muestra estos resultados, los mismos que además evidencian la influencia del DBMS y como éstos implementan los algoritmos de ejecución de las estructuras de sentencias.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

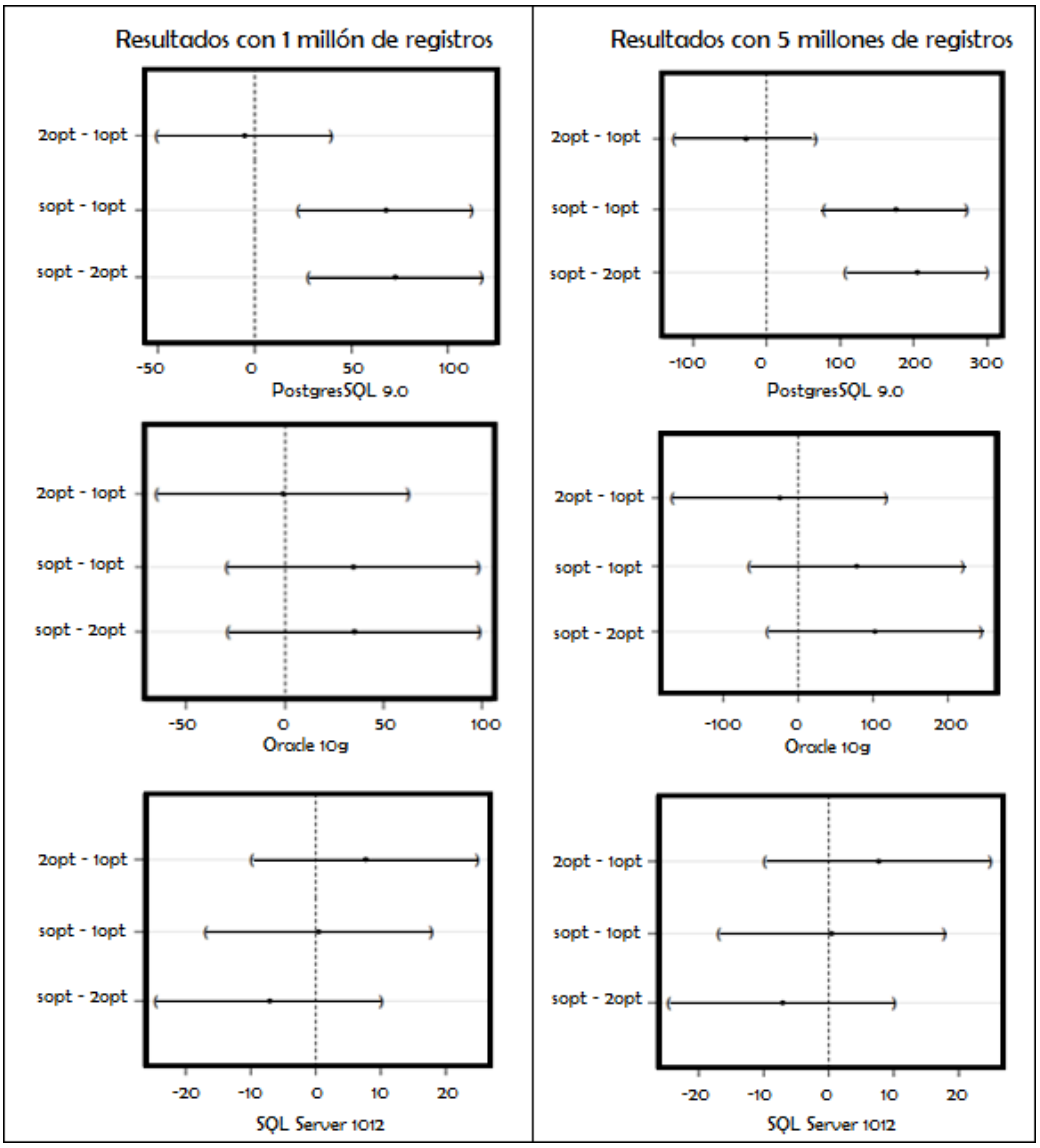

Figura 5. Comparaciones de medias de Tukey para los tiempos de respuesta de las sentencias SQL ejecutadas en gestores de base de datos PostgreSQL, Oracle y MS-SQL Server

En la figura 5 se muestra la aplicación del método de la diferencia honestamente significativa de Tukey, el cual mostró que existen diferencias significativas entre las sentencias no optimizadas y las optimizadas, tanto con uno como con cinco millones de registros. Cabe resaltar que entre las sentencias SQL optimizadas no existe mayor diferencia.

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

CONCLUSIONES Y TRABAJO FUTURO

Este artículo describe los resultados obtenidos al analizar el comportamiento de varios DBMS cargados con millones de datos y a la vez demuestra la importancia de realizar optimizaciones de las sentencias SQL. En efecto, utilizando la estructura de una base de datos real, se ha confirmado que las sentencias SQL complejas necesitan al menos una etapa de optimización para reducir los tiempos de respuesta. Se precisa que dichas optimizaciones sean realizadas para el gestor de bases de datos en el cual será ejecutada la sentencia para obtener mejores resultados. Esto es necesario porque cada gestor de bases de datos implementa de manera diferente el plan de ejecución de las sentencias SQL.

Por otro lado, las sentencias SQL diseñadas para PostgreSQL reaccionaron de forma diferente al ejecutarse en otros SGBD como Oracle y MS-SQL Server. Esto se debe a que sus algoritmos para resolver el árbol de planificación de ejecución de las sentencias SQL no son iguales, los procesos de optimización definen los planes de ejecución, la estrategia de ejecución de la consulta y elige el mejor plan de ejecución (Sukheja & Singh, 2010). En Oracle se mejoró en el tiempo de respuesta, pero estos fueron el doble de los alcanzados en PostgreSQL. Para MS-SQL Server los tiempos de respuesta se incrementaron; es decir, se obtuvo un resultado inverso al esperado.

Resumiendo lo antes señalado, el estudio aquí descrito proporciona, por un lado, evidencia de la importancia de optimizar las sentencias SQL, y por otro lado, la necesidad de hacerlo directamente con el gestor de base de datos usados y de ser posible debería revisarse dichas optimizaciones a medida que varía en el rango de millones el número de registros. Realizar tales optimizaciones es un aspecto importante al trabajar con grandes volúmenes de datos.

Finalmente, aunque existen otros aspectos que se espera abordar a futuro, uno de ellos se refiere a una de las sentencias SQL en particular y que requiere especial atención, la sentencia SELECT. Como es conocido, esta sentencia es muy utilizada en el contexto de las bases de datos, pero algunos métodos pueden ser ineficientes, en especial aquellos usados con sentencias SELECT anidadas (Ganski & Wong, 1987). Además, sería interesante analizar el comportamiento de dichas sentencias bajo configuraciones diferentes, como, por ejemplo, una mayor cantidad de datos de la aquí usada.

AGRADECIMIENTOS

Received 17/05/2018
Approved 05/07/2018

Revista Publicando, 5 No 16. (1). 2018, 70-79. ISSN 1390-9304

Los autores agradecen el aporte de Carlos Hidalgo para la elaboración de este trabajo.

REFERENCIAS

Al Mamun, M., & Kabir, M. H. (2008). Performance Improvement Techniques for Customized Data Warehouse. *Transactions, 6*(4), 332-338.

Dhande, S. S., & Bamnote, G. R. (2014). Quqry Optimitazion in OODBMS:Identifying subquer for query management. *International Journal of Database Management Systems*, 49-63.

Ganski, R. A., & Wong, H. K. (1987). Optimization of nested SQL queries revisited. *ACM SIGMOD Record, 16*(3), 23-33.

Geng, Y., & Zhang, C. (2009). Study on the Optimization Method of Select Query Sentence in Standard SQL Language. *Computer and Information Science, 2*(1), 121-125.

Larsen, S. M. (2005). Top Ten SQL Performance Tips. *Quest software*.

Rahman, S., Feroz, A. A., Kamruzzaman, M. N., & Faruque, M. N. (2010). Analyze Database Optimization Techniques. *IJCSNS, 10*(8), 275-279.

Salazar, E. (2015). Optimización de los tiempos de respuestas en una base de datos. *Gaceta Sana*, 56.

Singh, N., & Varshney, M. (2013). Query Recommendation employing Query Logs in Search Optimization. *Advanced Networking and Applications*, 1917-1921.

Sukheja, D., & Singh, U. K. (2010). Query Optimizer Model for Performance Enhancement of Data Mining Based Query. *International journal of computer science & communication, 1*(1).

Received 17/05/2018
Approved 05/07/2018