Revista Tecnológica ESPOL, Vol. 18, N. 1, 25-29, (Octubre, 2005), ISSN : 0257-1749

Sistema Optimizador de Almacenamiento Físico de una Base de Datos con Múltiples Dispositivos

Alex Paúl Tapia Chichande
Luis Miguel Escobar Larenas
Pedro Fabricio Echeverría Briones, Ing Comp
Facultad de Ingeniería En Electricidad Y Computación
ESCUELA SUPERIOR POLITECNICA DEL LITORAL
Campus La Prosperina, Guayaquil, Ecuador
aptapia@armadaecuador.com
luismiescobar@hotmail.com
pechever@espol.edu.ec

Resumen

El uso de las bases de datos cada día se populariza mas y más, razón por la cual se debe considerar que todos aquellos procesos a ser realizados en cada una de las mismas deberían estar encaminados al manejo ágil de la información. En el diario trabajo de los Administradores de Base de Datos (DBA), una de sus principales obligaciones es la de mejorar los tiempos de respuesta y respaldar las bases que se encuentran bajo su cargo, es el momento en donde surgen problemas tales como: ¿Tengo la capacidad suficiente? ¿Los dispositivos son los idóneos para esta función? ¿Puedo mejorar los tiempos de respuesta?¿Puedo mejorar el tiempo de respaldo?. Esta es la razón por la cual una vez detectado el problema, se planteó un CASE capaz de optimizar estos procesos, que sirvan para cualquier base de datos con cualquier dispositivo de almacenamiento físico, para lo cual se implemento un algoritmo genético que permita combinar la naturaleza de las tablas de las bases de datos mas las características propias de los dispositivos de almacenamiento, para que la respuesta recomiende soluciones que mejoren donde almacenar los objetos del modelo lógico en el modelo físico de almacenamiento.

Palabras Claves: *Administrador de Base de Datos, CASE, Algoritmo Genético, Optimización de procesos.*

Abstract

In the last years the use of databases has been popularized, that is the reason for consider that all those processes should be solve by an agile information management. In the hard work daily of the database administrators (DBA), one of the main tasks is supporting the bases, immediately appear in their mind the following doubts: Do I have enough capacity storage? Are the devices suitable for this function? Should I improve that time? This is the reason because we have created a CASE for any database and any storage device thanks the implementation of Genetic Algorithm capable of combining the nature of the tables (of the databases), with the own features devices, and recommend one of the better places to store the information.

Key words: database administrators, CASE, Genetic Algorithm, improve processes.

1. Introducción

Una base de datos es una fuente central de almacenamiento que esta diseñada para que sea compartida por muchos usuarios con una diversidad de aplicaciones. Esta se encuentra organizada y almacenada en diferentes dispositivos físicos, la cual puede ser manipulada por medio de programas que nos permiten un acceso directo a los datos de la misma, para de esta forma obtener: la información requerida en mucho menos tiempo, manejo de mayor cantidad de información, manejo de varios usuarios simultáneamente (concurrentes), la independencia de su uso, coherencia de resultados y el hecho de evitar la redundancia entre muchas otras virtudes. Sin embargo el manejo, uso, mantenimiento y respaldos de las mismas son un problema complejo para los administradores.

Recibido: Junio, 2005
Aceptado: Agosto, 2005

2. Justificativo, Materiales y métodos

Justificativo

Los algoritmos genéticos han sido desarrollados basados en el principio de la evolución es decir, el de mejor estado o condición es el que se impone sobre los demás (biológicamente) y sobre la base de él se genera el resto de su especie. Estos algoritmos han sido utilizados en resolución de problemas de optimización, tales como: optimización de funciones lineales y no lineales, problemas del viajero (la ruta más corta), problemas de itinerarios, partición, control entre otros. Es así como aprovechando técnicas de programación, tomando como base a estos algoritmos y considerando las necesidades evidenciadas por los DBA, creemos que existe la necesidad de crear un programa capaz de decidir cual es el lugar más adecuado para los respaldos realizados en las bases de datos razón por la cual era necesario la construcción de un CASE capaz de brindar asesoría para la mencionada labor.

Materiales

El sistema ha sido desarrollado en lenguaje java utilizando como compilador el Forte for Java 4 Community Edition, hemos utilizado este lenguaje en virtud de su orientación a objetos, multiplataforma y Open Source, lo que nos ha permitido un acceso fácil y barato para la implementación del mismo. En el caso de la base de datos se ha adquirido la licencia de SQL 7.0 por las bondades y facilidades que esta brinda para el desarrollo anteriormente mencionado.

Método de funcionamiento

Los autores del presente han considerado la creación de un algoritmo genético basados en una parametrización, que usa matrices para optimizar el lugar de almacenamiento en una base de datos. El objetivo consiste en minimizar las mejores soluciones de una o varias posibilidades. Se implementó una matriz la cual relaciona las tablas (de la base de datos) Tabla 1, con los discos utilizados en el almacenamiento, (Tabla 2), fundamentados en la naturaleza de la tabla y las características de los discos.

Tabla 1. Características discos

D1	D2	D3	D4	D5
Rápido	Lento	Rápido	Medianamente Rápido	Medianamente Rápido

Tabla 2. Naturaleza tablas

T1	Altamente Accesada
T2	Altamente Accesada
T3	Poco acceso
T4	Altamente Accesada
T5	Poco acceso
T6	Medianamente accesada
T7	Medianamente accesada
T8	Altamente Accesada
T9	Poco acceso
T10	Altamente Accesada

De la combinación de estas dos matrices se obtiene, la Tabla 3:

Tabla 3. Matriz pesos tabla disco

	D1	D2	D3	D4	D5
T1	10	0	10	5	5
T2	10	0	10	5	5
T3	0	10	0	5	5
T4	10	0	10	5	5
T5	0	10	0	5	5
T6	5	0	5	10	10
T7	5	0	5	10	10
T8	10	0	10	5	5
T9	0	10	0	5	5
T10	10	0	10	5	5

El algoritmo diseñado llena una matriz randomicamente con 0 y 1, para posteriormente convertirlas en las posibilidades ideales (10) en 1, a las regulares (5) en 1 con la opción de poder cambiarla a 0, o mantenerse como tal y finalmente a las posibilidades no deseadas las convierte en 0, de esta manera la matriz convertida quedará como en la Tabla 4.

Tabla 4. Matriz tabla vs. Disco convertida

	D1	D2	D3	D4	D5
T1	1	0	1	1/0	1/0
T2	1	0	1	1/0	1/0
T3	0	1	0	1/0	1/0
T4	1	0	1	1/0	1/0
T5	0	1	0	1/0	1/0
T6	1/0	0	1/0	1	1
T7	1/0	0	1/0	1	1
T8	1	0	1	1/0	1/0
T9	0	1	0	1/0	1/0
T10	1	0	1	1/0	1/0

Se ha clasificado inicialmente la distribución de tablas en los discos, sin embargo observamos que para las tablas: T1, T2, T4, T8 y T10 las posiciones ideales serian que se guarden en los discos: D1 o D3, recomendable en los discos D4 o D5 y no deseado en el disco D2, de esta manera si empezamos a colocar todas las tablas en los primeros discos saturaremos en poco tiempo los mismos, mientras que los siguientes estarían poco utilizados siendo esto ineficiente.

El algoritmo genético por medio de una función conocida como fitness (adaptatividad), la cual compara el cromosoma actual con uno promedio, de ser mejor

lo aprueba y recalcula el nuevo promedio caso contrario desecha al inicial, de esta manera evalúa todas las posibilidades de los cromosomas y recalcula cuales serían las mejores posiciones para las diferentes tablas es así como escoge en segunda instancia a las elecciones de la Tabla 5.

Tabla 5. Matriz tabla vs. Disco depurada por una corrida de algoritmo genético

	D1	D2	D3	D4	D5
T1	1	0	0	0	0
T2	0	0	1	0	0
T3	0	1	0	0	0
T4	1	0	0	0	0
T5	0	1	0	0	0
T6	0	0	0	1	0
T7	0	0	0	1	0
T8	1	0	0	0	0
T9	0	1	0	0	0
T10	0	0	1	0	0

En la tabla anterior podemos observar claramente como el problema de distribución se ha resuelto puesto que las tablas: T1, T4, T8 han sido colocadas en el disco ideal D1. De la misma manera para las tablas T2 y T10 se encuentran ubicadas en le disco D3 siendo este disco uno de los ideales para las mismas. La misma situación ocurre para el resto de tablas las cuales han sido colocadas en los lugares ideales para su almacenamiento.

Sin embargo el sistema ahora es poco eficaz puesto que no existe ninguna tabla que haya sido guardada en el disco D5, es decir los recursos han sido mal utilizados, por lo cual el sistema no ha sido explotado en su mayor capacidad, razón por la cual se ha creado la siguiente matriz, (Tabla 6) la cual nos ha permitido fijarnos en este error:

Tabla 6. Matriz sumatoria tabla vs. Disco.

	D1	D2	D3	D4	D5
Sumatoria	3	3	2	2	0

Entonces al algoritmo genético inicia un proceso de cruzamiento (cross over), el cual consiste en tomar los 2 cromosomas predecesores y de esta manera realizar un proceso de cruzamiento de genes en los mismos, los cuales se verán reflejados en uno nuevo, de características diferentes respecto a los dos anteriores, para visualizar con un ejemplo disponemos de la Figura 1.

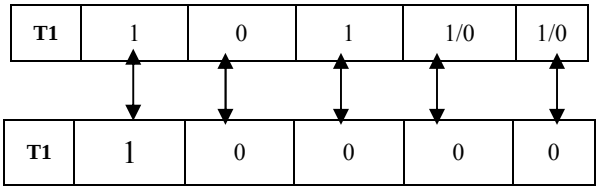

Figura 1. Cruzamiento (cross over)

Al obtener nuevos cromosomas los cuales son validados por la aplicación de la función fitness, en caso de no existir un cruzamiento aceptable es decir que no cumpla con la función, el algoritmo esta en capacidad de devolver randomicamente un descendiente.

Una vez realizado el cruzamiento, el algoritmo esta diseñado para realizar un proceso conocido como mutación el cual arbitrariamente altera uno o más genes de un cromosoma seleccionado; para de esta manera cubrir con posibilidades que no han sido consideradas por el cruzamiento, como lo apreciamos en la Figura 2.

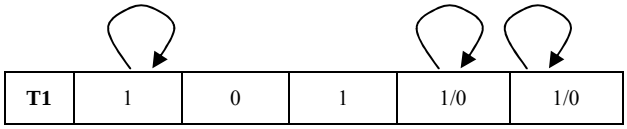

Figura 2. Mutación

De esta manera se puede apreciar que existen posibilidades que no han sido consideradas hasta el momento para su aplicación, de la misma manera al igual que en el caso anterior se calcula la probabilidad de mutación y la creación de nuevos cromosomas a los cuales nuevamente se aplica la función de adaptabilidad, para determinar si estos son ideales, regulares o no deseados.

El algoritmo genético trabaja con una aproximación igual a 1 y una vez obtenida este la convierte en posibilidades de almacenamiento indicando con el numero 1 como el ideal y al numero 0 como el no deseado. Adicionalmente nos indica el número de transacciones que realizó para llegar a este resultado, el cual después de varias pruebas realizadas lo estimamos en 650000 instancias (se debe considerar que mientras mayores sean las ocasiones de aplicación mas preciso será el resultado).). Como es obvio suponer no existe una respuesta única ideal y de la misma manera al realizar varias corridas sobre un mismo problema los resultados son diferentes, los mismos que han sido guardados en una base de datos.

3. Resultados obtenidos

Todos los valores mencionados aquí han sido tomados después de varias pruebas y de esta manera hemos obtenido un promedio de estos, tanto en los servidores de prueba como en los originales.

Tabla 7. Matriz sumatoria tabla vs disco

Base de datos / Servidor	DIPERGYE	SUELDOS
	5117Mb	4313 Mb
Servidores de prueba		
STORAGETECH Configuración ovalo No 1	145	6 intentos fallidos
Servidor IBM x SERIES 235 Configuración No 2	96	52
Servidores originales con nueva disposición de tablas		
Intel SBT2 (1)	112	No aplicable
Intel SBT2 (2)	No aplicable	55
Servidores originales disposición original de tablas		
Intel SBT2 (1) Configuración No 3	122	No aplicable
Intel SBT2 (2) Configuración No. 3	No aplicable	60
desventajas en tiempo de respuesta STORAGETECH	118.85%	No aplicable
beneficios en tiempo de respuesta IBM x SERIES 235	78.69%	86.67%
beneficios en tiempo de respuesta Intel SBT2 (1).	91.80%	No aplicable
beneficios en tiempo de respuesta Intel SBT2 (2)	No aplicable	91.66%

Analizando el cuadro anterior concluimos que:

Se obtienen muy malos resultados con el servidor STORAGETECH (Configuración No 1), dando una disminución de rendimiento equivalente al 18.85%, siendo la posible causa los problemas mencionados en la descripción del equipo, la degradación del mismo o la incompatibilidad de hardware y software.

En el mencionado servidor se produjo excesiva cantidad de errores por lo cual creemos que en versiones posteriores de este CASE se debería considerar la posibilidad de tomar en cuenta el uso y degradación de los discos, para de esta manera tratar de colocar un parámetro más de medición y lograr corregir situaciones como esta.

Se tiene buenos resultados con el servidor IBM x SERIES 235 (Configuración No.2), dando una mejora en rendimiento medido en tiempo a un equivalente de 21.31% (para la base de datos DIGPERGYE) y del 13.33% (para la base de datos de SUELDOS). Sin embargo se debe considerar que los servidores disponen de RAID, lo cual no permite verificar la eficiencia auténtica del CASE, en vista de que los servidores originales no disponen de RAID.

En los servidores originales, con la nueva disposición de las tablas (Configuración No.3), se obtuvieron mejoras del 8.196% (para la base de datos DIGPERGYE) y del 8.33% (para la base de datos de SUELDOS)

Para la utilización del CASE se considera la velocidad standard de los discos, la cual viene como característica propia dada por el fabricante, en ningún momento se considera la velocidad optimizada del RAID.

En el CASE elaborado se ha considerado únicamente las tablas en las cuales se guardan datos, motivo por el cual no se ha manipulado ninguna tabla perteneciente al sistema. Esta es la razón por la que los índices se han guardado de acuerdo a la disposición recomendada por el SGBD.

3. Conclusiones

Se optimizó la capacidad de almacenamiento en los discos utilizados, tanto en los sistemas originales como en los que disponen de RAID.

Se mejoró la velocidad en los procesos de respaldo realizado en las diferentes transacciones ordinarias en las bases de datos.

Actualmente se dispone de varias alternativas óptimas para la elección del sitio de almacenamiento de las diferentes tablas de las bases de datos, y no simplemente estar a la disposición de lo que nos indica la base de datos.

Se puede crear un modelo de comportamiento, Histórico del CASE, basados en las acciones realizadas sobre la misma; una vez que se obtenga la suficiente cantidad de datos.

4. Recomendaciones

Para versiones futuras se podrá considerar el uso y la degradación de los discos duros y dispositivos, para de esta manera obtener un parámetro más a ser medido y mejorar los valores obtenidos para equipos con problemas.

Creación de históricos de bases de datos ya utilizadas.

Creación de modelos aplicables a situaciones parecidas de diferentes bases de datos.

5. Agradecimiento

Al Ing Fabricio Echeverría quien con su destreza y conocimientos nos supo orientar en la elaboración del presente

6. Referencias

[1] ZBIGNIEW, Michalewicz, "*Genetic Algorithms + Data Structures = Evolution Programs*", third extended revised edition, Springer Verlag heidenberg New York, 1999.

[2] KENDAL & KENDAL, "*Análisis Y Diseño De Sistemas*", tercera edición, México, 1997.

[3] GILL &RAO, "*Data Ware Housing"*, edición en español, Prentice Hall, Hispanoamericana, Naucaplan de Juárez, Edo México, México, 1996.

[4] LATHROP, WEBSTER, TEMPLE, "*Ariadne: Pattern Directed Inference And Hierarchical Abstraction In Structure Recognition*", paper, Communications of the ACM, Massachusetts, USA, 1987.

[5] GRIMSON, Eric, " The combinatorics of local constraints in model-based recognition and localization from sparse data.", paper, Journal of the ACM, California, USA, 1986.

[6] LIBERATORE, Paolo, "*Compability And Compact Representation Of Revision Of Horn Knowledge Bases*", paper, ACM Transactions on Computational Logic, 2000.

[7] COSTA, Carlos, "*Algoritmos Genéticos*", 2003, http://ccp.servidores.net/genetico.html

[8] Larraña, "*Algoritmos Genneticos*", Departamento de Ciencias de Computación e inteligencia artificial, Universidad del país vasco, 2003, http://geocities.com/CapeCanaveral/9802/3d5ca000.htm

[9] CONTRERAS, J, "*Tutorial sobre algoritmos geneeticos* ", Maracaibo, 2003, http://www.ianet.com/users/jcontre/genetic

[10] COELLO, Carlos, "*Introducción a los algoritmos genéticos"* , 2003, http://redcientifica.com/doc/doc199904260011.html