

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Implementación de un algoritmo genético para optimizar el checkpoint en una base de datos oltp

Modalidad: Proyecto Tecnológico

Trimestre 2017 Primavera

Oscar Alberto Jiménez Huerta

205202887

al205202887@alumnos.azc.uam.mx

M. en C. Hugo Pablo Leyva

Profesor Titular

Departamento de Sistemas

hpl@correo.azc.uam.mx

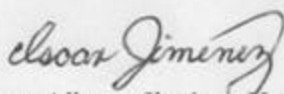
4 de septiembre de 2017

Yo, Hugo Pablo Leyva, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



M. en C. Hugo Pablo Leyva

Yo, Oscar Alberto Jiménez Huerta, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Oscar Alberto Jiménez Huerta

Resumen

En el presente proyecto se crea un programa con el objetivo de emplear un algoritmo genético para optimizar el checkpoint en una base de datos. El checkpoint o punto de control se encarga de bajar la información presente en el caché hacia el disco duro, para evitar la pérdida de datos en caso de una caída. El uso de un algoritmo genético, para encontrar los valores óptimos de los parámetros que afectan el checkpoint, es una forma más eficiente para reducir el tiempo de duración del checkpoint. Una base de datos OLTP del inglés *On-Line Transaction Processing* o dinámica es aquella en la cual se insertan, modifican y eliminan datos en línea durante la operación del sistema.

El programa hace uso de un Sistema Gestor de Base de Datos, SGBD, o manejador de base de datos y está integrado por los siguientes módulos: el módulo Ambiente de pruebas, el módulo Métricas, el módulo Cambios en el manejador y el módulo Algoritmo genético. Dentro de este último, se hace uso del Algoritmo genético simple de Goldberg. Para la realización del programa se hace uso de software libre como lo son: el sistema operativo Linux y el Sistema Gestor de Base de Datos PostgreSQL.

Tabla de contenido

Resumen	3
Introducción	5
Antecedentes y Justificación	6
Antecedentes	6
Justificación	6
Objetivos.....	7
Objetivo general.....	7
Objetivos específicos	7
Marco teórico	8
Algoritmo genético.....	8
Checkpoint.....	12
Base de datos OLTP	13
Desarrollo del proyecto	14
Elementos generales	14
Metodología	15
Resultados	20
Conclusiones.....	21
Referencias bibliográficas.....	22
Apéndice	23

Introducción

Cuando se trabaja con una base de datos OLTP, del inglés *On-Line Transaction Processing*, o de procesamiento de transacciones en línea, en la que la información se modifica en tiempo real, o durante la operación del sistema, el conjunto de inserciones, eliminaciones y modificaciones de datos se almacena en el caché de la memoria, posteriormente estas operaciones o transacciones son bajadas al disco duro, para evitar la pérdida de información en caso de que ocurra una caída del sistema; el evento encargado de realizar esta acción se denomina checkpoint. “El checkpoint es un punto de sincronización entre la base de datos y el log de transacciones [1]”.

Encontrar la duración óptima del checkpoint es un proceso matemático complejo, que involucra el estudio de los valores de los parámetros que afectan el checkpoint, para facilitar este proceso se puede hacer uso de un algoritmo genético, el cual es un algoritmo de búsqueda basado en los mecanismos de selección natural y genética natural [2].

El empleo del algoritmo genético va a servir para encontrar, de forma más rápida y eficiente, los valores óptimos o aquellos que están muy cercanos al óptimo, de los parámetros que determinan el checkpoint. Posteriormente estos valores se utilizarán para modificar aquellos que se encuentran en el Sistema Gestor de Base de Datos, SGBD, o manejador de base de datos, programa encargado de administrar y almacenar información en una base de datos, para cuando se presente el checkpoint.

Antecedentes y Justificación

Antecedentes

Existen programas en el ámbito educativo que hacen uso de los algoritmos genéticos para resolver diferentes problemas, por ejemplo:

- 1 Programa para la categorización automática de documentos [3],
- 2 Aplicación para la distribución óptima de horarios de clases [4] y
- 3 Software para la asignación de carga académica en instituciones de educación superior [5].

Para mejorar el desempeño de una base de datos existe, por ejemplo, un programa que mediante el empleo de un algoritmo genético optimiza el operador unión del SGBD PostgreSQL [6].

Justificación

Determinar mediante el empleo de un algoritmo genético, los valores óptimos que deben tener los parámetros que ejecutan el checkpoint en un manejador de base de datos de software libre, es algo que puede ser muy útil para cualquier administrador de una base de datos, ya que en la actualidad encontrar estos valores se hace mediante un procedimiento empírico.

Objetivos

Objetivo general

Disminuir la duración del evento del checkpoint en una base de datos OLTP mediante la implementación de un algoritmo genético que permita mejorar los parámetros de un manejador de base de datos.

Objetivos específicos

- 1 Encontrar una representación adecuada de los parámetros de un manejador de base de datos relacionales para emplear el algoritmo genético.
- 2 Implementar el módulo Ambiente de pruebas en una base de datos OLTP en la que se implemente un checkpoint.
- 3 Desarrollar una aplicación que implemente el módulo Algoritmo genético y permita modificar, vía el módulo Cambios en el manejador, los parámetros del manejador de base de datos para optimizar el checkpoint.
- 4 Desarrollar el módulo Métricas que valide que ya se logró el objetivo de minimizar la duración del checkpoint.

Marco teórico

Algoritmo genético

Los algoritmos genéticos son una variante de los algoritmos evolutivos y fueron propuestos por John Holland en 1975. Tienen la particularidad de que para un problema dado la solución obtenida se da mediante una cadena de bits.

Elementos de un algoritmo genético:

Representación de los individuos

Para identificar los datos de un individuo, en ocasiones se utiliza el vocabulario particular de la Biología. Así en un algoritmo genético los individuos se representan mediante cadenas binarias, que se indican como b , genotipo, y que representan a puntos x , fenotipo, del espacio de búsqueda de un problema. La codificación de una determinada característica del individuo es el gen, el cual a menudo se identifica con cada posición de la cadena binaria. A una determinada posición de la cadena binaria se le llama locus y a los distintos valores que puede tomar un gen se le denomina alelo.

Aún cuando la representación binaria que utilizan los algoritmos genéticos los hace eficientes, es necesario tener un método para pasar esta representación binaria al área de búsqueda natural al problema, lo cual permite evaluar la adecuación de un individuo como solución al problema.

El método de codificación seleccionado debe considerar lo siguiente:

- 1 En la codificación los n puntos del espacio de búsqueda deben estar representados por n cadenas binarias.
- 2 Cada posición en la cadena debe tener un significado para el problema, esto permitirá que los genes que le proporcionan alta calidad a un individuo generen características de calidad en un nuevo individuo.
- 3 La codificación debe permitir alcanzar de forma eficiente al fenotipo.

Generación de la población inicial

La población inicial está integrada por individuos, cadenas de ceros y unos generadas de forma aleatoria, en donde cada gen se genera con una función que devuelve cero o uno. Cada individuo puede ser solución al problema o la base para encontrar la solución a éste. El proveer a la población de variedad permite explorar todos los puntos del espacio de búsqueda, y más importante, sirve para que el algoritmo genético funcione correctamente.

Grado de adaptación de los individuos

La calidad relativa de los individuos, los cuales compiten por incrementar su número en la población y por reproducirse, determina la evolución de la población. Cuando se trata de un problema de optimización, la adaptación de un individuo a ser solución al problema, es lo que determina su calidad. Si el objetivo es optimizar una función matemática entonces la función de adaptación coincidirá con la función a optimizar.

Condiciones de terminación

Se debe establecer una condición de terminación o paro, con el fin de determinar el punto en el cual el algoritmo deja de evolucionar y proporciona la mejor solución. La condición de terminación más simple es cuando se llega a un determinado número de generaciones de evolución; también lo son: cuando debido a la poca diversidad, la mayoría de la población tiene ya una forma similar, y cuando la solución obtenida tiene una calidad específica.

Proceso de selección

La población de individuos se mete a un proceso de selección el cual elimina a ciertos individuos y favorece las copias de los individuos más adaptados. Los mecanismos de muestreo para realizar el proceso de selección son los siguientes:

- Selección por ruleta o proporcional

La probabilidad de selección p_i de un individuo i es proporcional a su adaptación relativa:

$$p_i = \frac{f(i)}{\bar{f}}$$

donde \bar{f} es la adaptación promedio de la población.

Para poder emplear el siguiente método, antes que nada, se requiere que de forma uniformemente distribuida se generen números aleatorios en un intervalo cerrado como puede ser $[0,1]$, posteriormente se deben seguir los siguientes pasos:

1 Definir las puntuaciones acumuladas como se muestra a continuación:

$$q_0 := 0 \quad \text{y} \\ q_i := p_1 + \dots + p_i \quad (\forall i = 1, \dots, n)$$

2 Generar un número aleatorio en el intervalo cerrado y

3 Seleccionar al individuo i que cumpla la siguiente condición:

$$q_{i-1} < a < q_i$$

- Muestreo estocástico universal

En este mecanismo se debe generar un número aleatorio, el cual será la base para generar los m números o individuos necesarios. Después de que se han calculado los m números se deben aplicar los pasos del método anterior.

La fórmula para realizar el cálculo de los m números es:

$$a_i := \frac{a + i - 1}{m} \quad (\forall_i = 1, \dots, m)$$

- Selección por torneo

En este procedimiento se selecciona de manera aleatoria una pequeña muestra de la población a partir de la cual se elige al individuo con el mejor valor de adaptación; lo anterior se repite tantas veces como individuos se deseé tener. El procedimiento de selección puede ser de dos formas: determinista y probabilístico.

En la forma determinista se seleccionan n individuos de forma aleatoria, donde n generalmente es igual a 2, de los cuales se selecciona el de mayor valor de adaptación.

En la forma probabilística después de seleccionar a los n individuos, se aplica determinada probabilidad en la selección de los individuos: si un número generado de forma aleatoria, entre 0 y 1, es mayor a cierto límite entonces, en ese caso, se elegirá al mejor, pero en caso de que el número no sea mayor al límite determinado entonces se elegirá al peor.

- Muestreo por restos

En este mecanismo se hace una selección proporcional a la adaptación de los individuos: de cada individuo x_i se seleccionan $p_i k$ copias para la muestra, donde p_i es la probabilidad de selección del individuo i y k es el número de individuos por seleccionar. La selección de los individuos que faltan para completar la muestra se puede hacer mediante alguno de los mecanismos de muestreo anteriores.

Proceso de reproducción

En esta etapa se crean nuevos individuos al aplicar operadores genéticos a los individuos de la población de la generación anterior; esto permite al algoritmo genético llegar a nuevas áreas del espacio de búsqueda. Los operadores genéticos más utilizados son: el operador de cruce y el operador de mutación. Cualquiera de estos operadores se puede emplear, solamente si el valor generado de forma aleatoria rebasa una tasa determinada.

- Operador de cruce

Este operador produce nuevos individuos a partir de la combinación de las propiedades de dos individuos de la población anterior. Los nuevos individuos pueden presentar una mejor adaptación que los individuos de la generación precedente.

- Operador de mutación

Este operador crea un nuevo individuo al efectuar un pequeño cambio a un individuo de la población anterior.

Aún cuando el aplicar este operador a los individuos de la población puede romper las correlaciones existentes entre genes, derivadas de la evolución de la población, lo que genera individuos con peor adaptación que la que tenían los individuos que formaban parte de la población anterior; el emplear este operador permite que el algoritmo trabaje correctamente, como resultado de la existencia una población diversa.

Proceso de reemplazo

Para que el tamaño de la población permanezca constante los individuos de la población anterior deben ser reemplazados por los individuos creados en el proceso de reproducción.

Los algoritmos genéticos se dividen, dependiendo del número de individuos reemplazados en:

- Algoritmos con estado estacionario

En este tipo de algoritmos una parte de la población se mantiene de una generación a otra: algunos de los individuos de la población anterior son reemplazados por nuevos individuos. Los tipos de reemplazo más comunes son:

1 Reemplazo de los padres

En este tipo de reemplazo los padres son reemplazados por los hijos.

2 Reemplazo aleatorio

En esta clase de reemplazo el tamaño de la población y las tasas de cruce y mutación determinan el tamaño de la población de la siguiente generación, este tamaño determina la cantidad de individuos que de manera aleatoria se van a eliminar.

3 Reemplazo de los individuos con peor adaptación

En este caso, entre los individuos con los valores de adaptación más bajos se eligen, de forma aleatoria, a aquellos que se van a eliminar.

4 Reemplazo de los individuos con valor de adaptación similar

En este tipo de reemplazo un individuo de la generación anterior es reemplazado por otro que tenga un valor semejante de adaptación.

- Algoritmos genéticos generacionales

En esta clase de algoritmos toda la población de una generación es sustituida por otra al pasar a otra generación.

Checkpoint

El manejador de base de datos PostgreSQL conserva un registro de transacciones o WAL del inglés *Write Ahead Log*, o *transaction log* en el subdirectorio `pg_xlog` del directorio de datos del sistema de archivos o *cluster*. El *log* describe cada cambio realizado a los archivos de datos de la base de datos. En caso de que el sistema se caiga la base de datos puede ser restablecida con consistencia al volver a ejecutar los registros del *log* realizados desde el último checkpoint.

“Los checkpoints son puntos en la secuencia de transacciones en los cuales está garantizado que los archivos de datos han sido actualizados con toda la información escrita antes del checkpoint [8].” Los checkpoints son puntos en el WAL a partir de los cuales inicia la recuperación en caso de una caída.

Antes de que se realice un checkpoint los cambios en los archivos de datos ya han sido vaciados a los archivos WAL. Cuando llega el checkpoint todas las páginas de datos sucias, páginas con cambios que no han sido escritas en el disco, son vaciadas al disco y en el archivo *log* se escribe un registro especial del checkpoint. “En caso de una caída, el procedimiento de recuperación de caídas examina el último registro del checkpoint para determinar el punto en el log o registro redo a partir del cual debe iniciar la operación REDO [8].” La operación REDO permite rehacer una transacción, para lo cual es necesario que el *log* conserve, después de que se haya realizado un cambio en la base de datos, una copia de cada registro de la base de datos.

Un checkpoint puede volverse un problema para el desempeño cuando se trata de servidores de bases de datos concurrencios, debido al número de escrituras necesarias es decir: vaciar todas las páginas de datos sucias a disco puede causar una carga de entrada y/o salida importante; para evitar esto, la actividad del checkpoint es estrangulada para que la carga de entrada y/o salida, comience al inicio del checkpoint y termine antes de que inicie el siguiente checkpoint.

Existen dos parámetros que determinan cuando es llamado un checkpoint: el primero es el `checkpoint_segments`, el cual controla los segmentos *log* que serán escritos antes de que un checkpoint sea accionado, su valor inicial es de 3 segmentos en donde cada archivo de segmento es normalmente de 16 megabytes, el segundo parámetro es el `checkpoint_timeout`, el cual es el número de segundos hasta el siguiente checkpoint, el valor preestablecido es de 5 minutos. Cuando se llega al límite de alguno de los dos parámetros es cuando se presenta un checkpoint.

Se puede disminuir o aumentar el valor o los valores de alguno o de ambos parámetros. El disminuir algún valor causa que los checkpoints se realicen más frecuentemente, aunque esto permite una recuperación más rápida después de una caída: dado que requerirá menos trabajo por rehacer, una disminución incrementará el costo de vaciar las páginas de datos

sucias más a menudo; por otro lado, aumentar el valor de alguno de estos parámetros puede incrementar el tiempo necesario para la recuperación en caso de una caída.

Base de datos OLTP

Las bases de datos se pueden dividir en: bases de datos OLTP del inglés *On-Line Transaction Processing*, o dinámicas y bases de datos OLAP del inglés *On-Line Analytical Processing* o estáticas.

Un sistema o base de datos OLTP, provee a las bases de datos empresariales de datos que aún no han sido procesados. Este tipo de sistema es principalmente usado en el procesamiento de un gran número de transacciones pequeñas en línea: operaciones de inserción, actualización y borrado de datos. Entre sus características se encuentra la de proporcionar un rápido procesamiento de consultas, basado en la eficiencia así como en el mantenimiento de la integridad de la base de datos en un ambiente multiusuario.

Las bases de datos OLTP son empleadas por ejemplo: en las instituciones bancarias, en el comercio electrónico, en las tiendas de autoservicio y en la manufactura.

Elementos generales

El proyecto está formado por cuatro módulos: el módulo Ambiente de pruebas, el módulo Algoritmo genético, el módulo Cambios en el manejador, el módulo Métricas y se hace uso del manejador de base de datos PostgreSQL. La relación que tienen los módulos y el manejador de base de datos se muestra en la Figura 1.

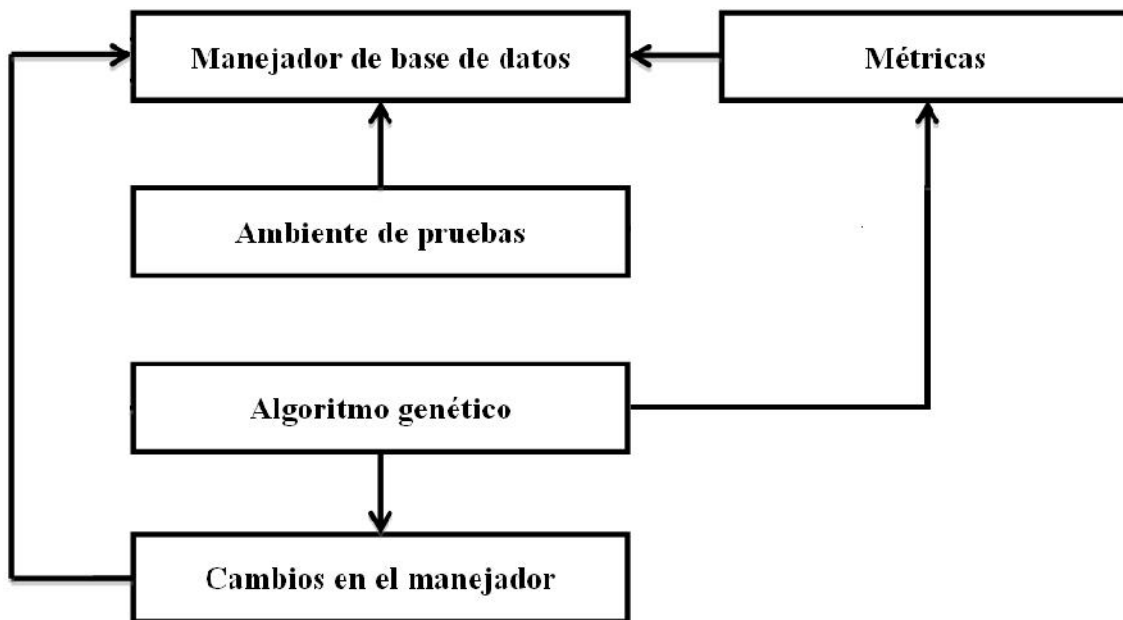


Figura 1. Relación entre los módulos y el manejador de base de datos

Manejador de base de datos

El manejador recibirá del módulo Ambiente de pruebas las operaciones de insert, delete, update y select. Del módulo Métricas recibirá las instrucciones para monitorear la duración de checkpoint. Del módulo Cambios en el manejador recibirá los valores a usar de sus parámetros. Del módulo Algoritmo genético vía el módulo Cambios en el manejador recibirá los parámetros con los que trabajará y del módulo Métricas se tendrán las evaluaciones que requiere el algoritmo genético para optimizar.

Ambiente de pruebas

En esta parte mediante un programa se simularán los accesos a la base de datos mediante selects, inserts, etcétera.

Algoritmo genético

Este módulo se encargará de implantar el algoritmo.

Cambios en el manejador

Este módulo se encargará de hacer los cambios en los parámetros del manejador según se lo indique el módulo del Algoritmo genético.

Métricas

Este módulo se encargará de tomar los valores que permitan saber si ya se cumplieron los objetivos al afinar el checkpoint en el manejador de base de datos.

Metodología

Para el desarrollo del proyecto se empleó la versión 16.04 LTS de la distribución Ubuntu del sistema operativo Linux, como manejador de base de datos se instaló PostgreSQL en su versión 9.5.7, así también, se instaló la biblioteca libpqxx, la cual permite que un programa escrito en el lenguaje de programación C++ pueda acceder a la base de datos en PostgreSQL.

Para simular la actividad en el manejador de base de datos se creó la base de datos Profesores.

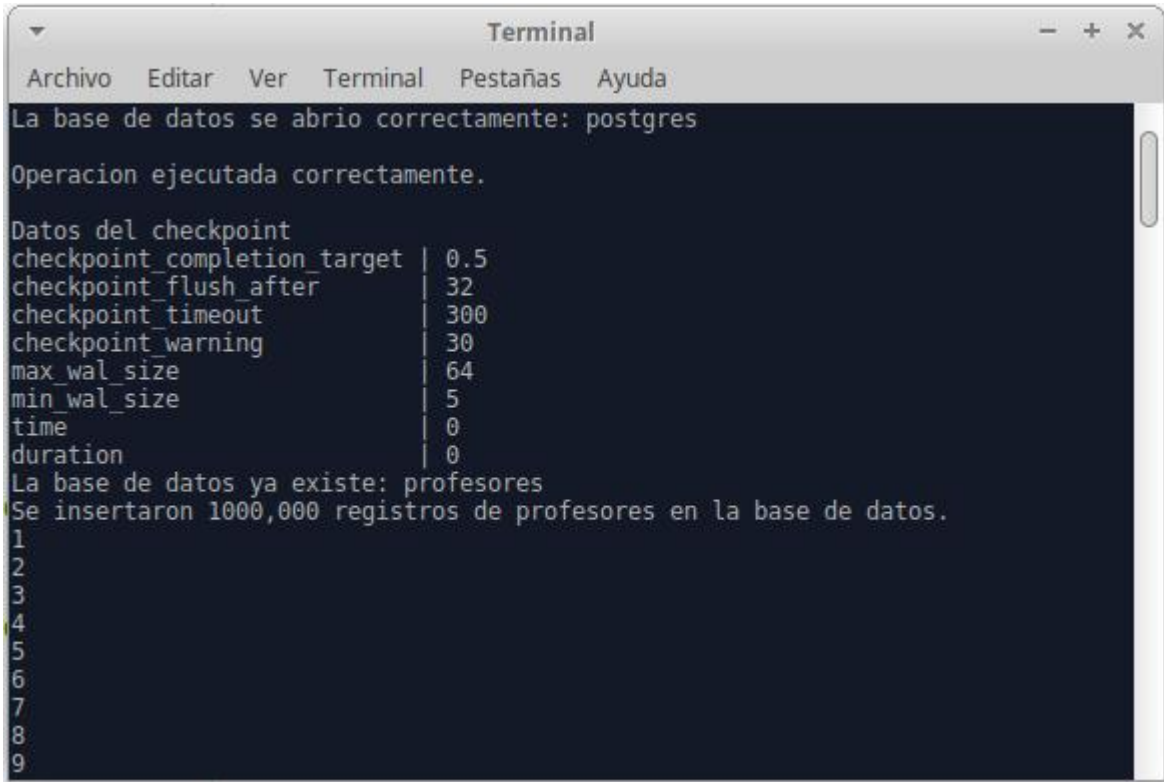
El módulo Ambiente de pruebas, fue el módulo que se creó a continuación para simular la actividad en la base de datos mediante operaciones de inserción masiva de datos.

En el módulo Métricas se recopilaban los valores obtenidos, como resultado de la actividad realizada por el módulo Ambiente de pruebas sobre la base de datos instalada en el manejador.

El módulo Algoritmo genético se realizó empleando el Algoritmo genético simple de Goldberg, el cual trabaja con los valores provistos por el módulo Métricas.

El módulo Cambios en el manejador se creó con el objetivo de emplear los valores proporcionados por el algoritmo genético, y realizar los cambios que fuesen necesarios en el manejador para optimizar el checkpoint.

Como parte de la ejecución del programa se obtienen los valores de algunos de los parámetros del checkpoint. Posteriormente, se insertan en la base de datos 1, 000,000 de registros para simular accesos a la base de datos. Y a continuación, se hacen consultas de prueba. Lo anterior se muestra en la Figura 2.



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
La base de datos se abrio correctamente: postgres
Operacion ejecutada correctamente.
Datos del checkpoint
checkpoint_completion_target | 0.5
checkpoint_flush_after       | 32
checkpoint_timeout           | 300
checkpoint_warning            | 30
max_wal_size                  | 64
min_wal_size                  | 5
time                          | 0
duration                       | 0
La base de datos ya existe: profesores
Se insertaron 1000,000 registros de profesores en la base de datos.
1
2
3
4
5
6
7
8
9
```

Figura 2. Información sobre el checkpoint, y la ejecución de inserciones y consultas como parte una primera prueba.

En la Figura 3 se muestra una parte de las salidas generadas por consultas realizadas a la base de datos como parte de una segunda prueba.

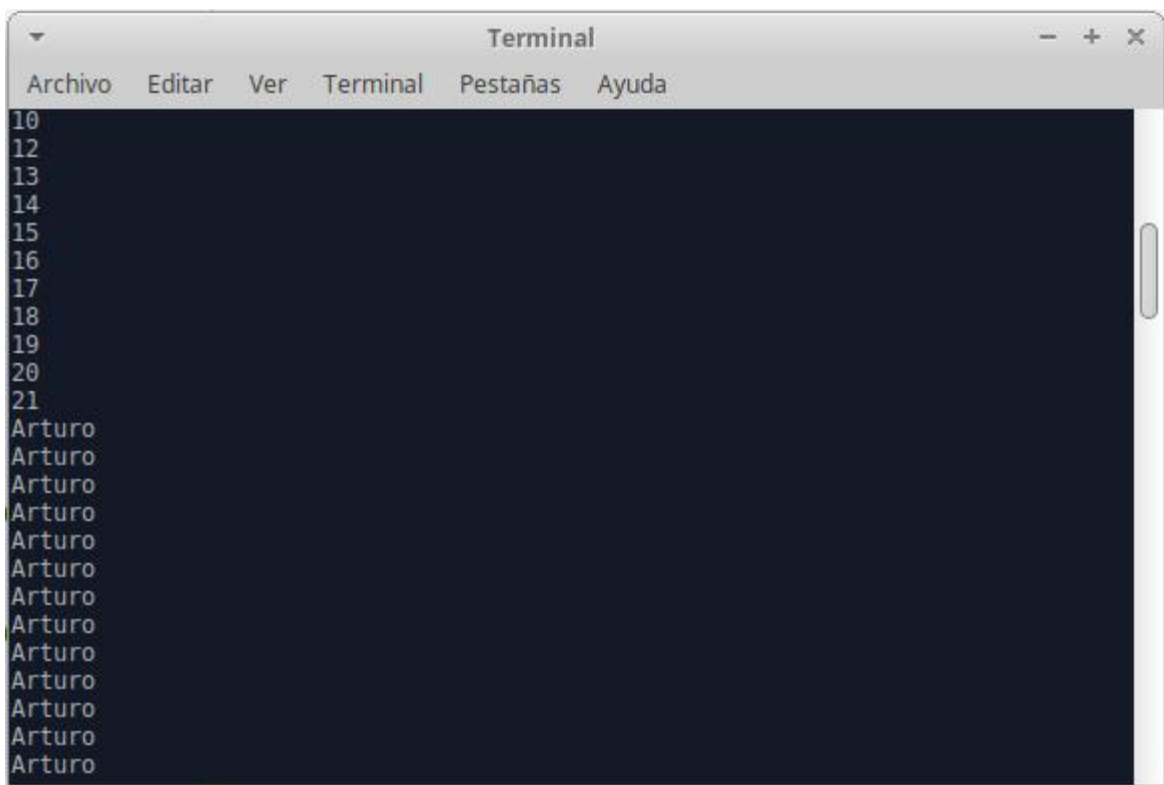


Figura 3. Salidas proporcionadas como resultado a consultas a la base de datos, como parte de una segunda prueba.

Las consultas realizadas como parte de una tercera prueba, así como la consulta al intervalo del checkpoint se muestran en la Figura 4.

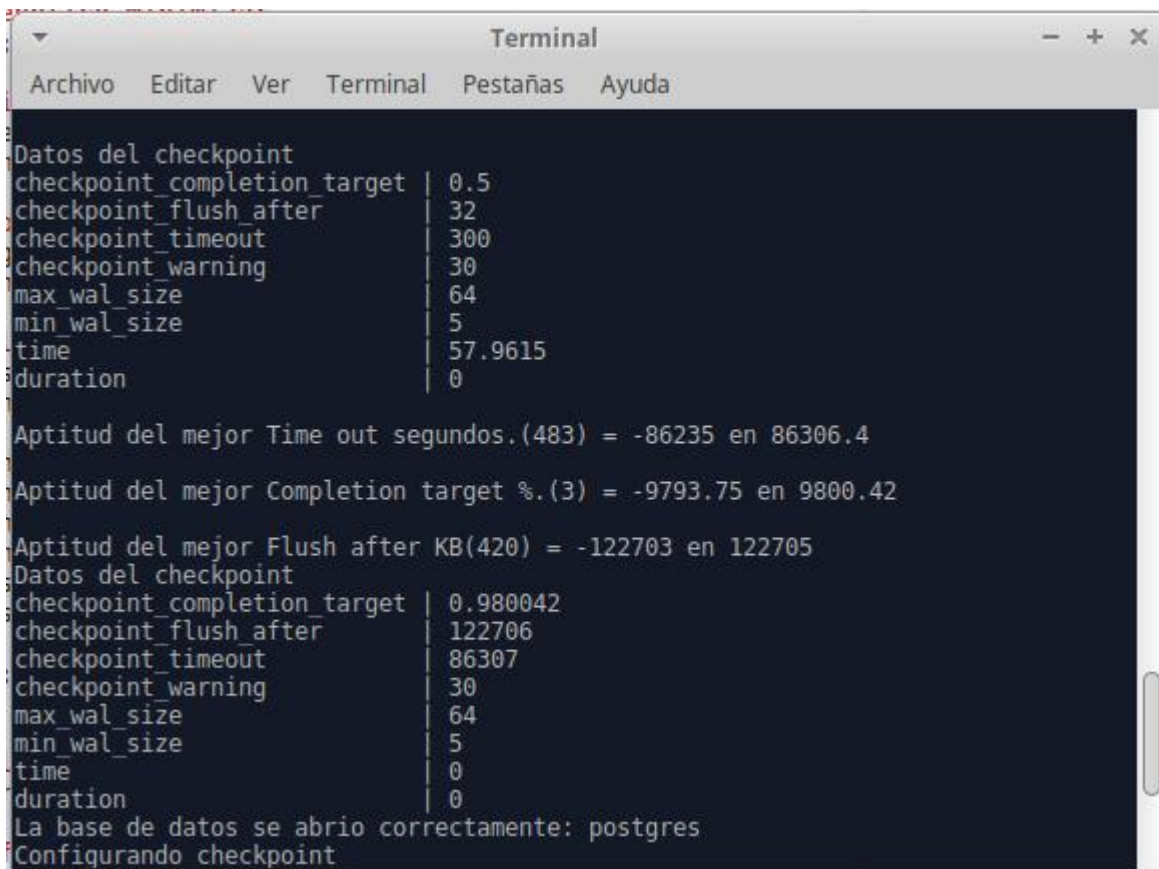
```

Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
Arturo
1 | Juan | Perez | Lopez | FM | ELT | 108 | 108 | Temporal | 2017-12-15 | 10000 |
2 | Jose | Hernandez | Flores | AD | PDR | 205 | 205 | Permanente | | 15000 |
3 | Laura | Chavez | Martinez | EA | ECN | 209 | 209 | Permanente | | 20000 |
4 | Carlos | Peña | Ramirez | MB | AAS | 220 | 220 | Permanente | | 17500 |
5 | Maria | Cardenas | Sanchez | AD | MAD | 162 | 162 | Temporal | 2017-12-15 | 12500 |
6 | Claudia | Cordero | Quesada | EA | AMN | 207 | 207 | Permanente | | 20000 |
7 | Elena | Rojo | Salgado | MB | AAS | 106 | 106 | Temporal | 2017-12-15 | 7500 |
8 | Lilia | Carrasco | Carrillo | AD | IPD | 256 | 256 | Permanente | | 7500 |
9 | Luis | Sarmiento | Diaz | FM | STM | 118 | 118 | Permanente | | 15000 |
10 | Arturo | Ruiz | Oropeza | FM | CEC | 172 | 172 | Permanente | | 17000 |
12 | Juan | Perez | Lopez | FM | ELT | 108 | 108 | Temporal | 2017-12-15 | 10000 |
13 | Jose | Hernandez | Flores | AD | PDR | 205 | 205 | Permanente | | 15000 |
14 | Laura | Chavez | Martinez | EA | ECN | 209 | 209 | Permanente | | 20000 |
15 | Carlos | Peña | Ramirez | MB | AAS | 220 | 220 | Permanente | | 17500 |
16 | Maria | Cardenas | Sanchez | AD | MAD | 162 | 162 | Temporal | 2017-12-15 | 12500 |
17 | Claudia | Cordero | Quesada | EA | AMN | 207 | 207 | Permanente | | 20000 |
18 | Elena | Rojo | Salgado | MB | AAS | 106 | 106 | Temporal | 2017-12-15 | 7500 |
19 | Lilia | Carrasco | Carrillo | AD | IPD | 256 | 256 | Permanente | | 7500 |
20 | Luis | Sarmiento | Diaz | FM | STM | 118 | 118 | Permanente | | 15000 |
21 | Arturo | Ruiz | Oropeza | FM | CEC | 172 | 172 | Permanente | | 17000 |
10 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
21 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
32 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
43 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
54 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
65 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
76 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
87 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
98 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
109 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
120 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
131 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
142 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
153 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
164 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
175 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
186 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
197 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
208 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
219 | Arturo | Ruiz | Oropeza | FM | CEC | 172 |
1 | Juan
Arturo
La base de datos se abrió correctamente: postgres
total_checkpoints | 845
minutes_between_checkpoints | 58.9613133717949

```

Figura 4. Consulta a la base de datos y del intervalo del checkpoint

Los valores generados por el algoritmo genético para los parámetros `checkpoint_timeout`, `checkpoint_completion_target` y `checkpoint_flush_after` se muestran en la Figura 5.



```
Terminal
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda

Datos del checkpoint
checkpoint_completion_target | 0.5
checkpoint_flush_after      | 32
checkpoint_timeout          | 300
checkpoint_warning          | 30
max_wal_size                | 64
min_wal_size                | 5
time                        | 57.9615
duration                    | 0

Aptitud del mejor Time out segundos.(483) = -86235 en 86306.4
Aptitud del mejor Completion target %.(3) = -9793.75 en 9800.42
Aptitud del mejor Flush after KB(420) = -122703 en 122705
Datos del checkpoint
checkpoint_completion_target | 0.980042
checkpoint_flush_after      | 122706
checkpoint_timeout          | 86307
checkpoint_warning          | 30
max_wal_size                | 64
min_wal_size                | 5
time                        | 0
duration                    | 0
La base de datos se abrio correctamente: postgres
Configurando checkpoint
```

Figura 5. Valores generados por el algoritmo genético para los parámetros y `checkpoint_timeout`, `checkpoint_completion_target` y `checkpoint_flush_after`.

Resultados

Después de realizar pruebas sobre la base de datos de 100,000, 200,000 hasta 1, 000,000 de registros, la disminución en la duración del evento del checkpoint es muy pequeña; un elemento que pudiera servir para confirmar que si se presenta una disminución, es el indicador de actividad sobre el disco duro, el cual muestra una reducción de funcionamiento después de que el algoritmo genético ha trabajado con los parámetros del checkpoint.

Conclusiones

El algoritmo genético si proporciona una forma de mejorar el desempeño de una base de datos, como lo marca la teoría sobre el uso y funcionamiento de los algoritmos genéticos; para que la disminución en la duración del checkpoint sea más perceptible, se debe generar mucha mayor actividad sobre la base de datos, ya sea incrementando el número de usuarios y/o aumentando el flujo de datos.

Referencias bibliográficas

- [1] D. M. Kroenke, (25-03-2015), *Procesamiento de bases de datos: fundamentos, diseño e implementación* (8.ª ed.) [En línea]. Disponible: https://books.google.com.mx/books?id=7ORUWItwcNEC&pg=PA321&lpg=PA321&dq=checkpoint+base+de+datos&source=bl&ots=KvQOSqCLpU&sig=_8WQhXLCpfoNcZYmkHjzsJYHu18&hl=es-419&sa=X&ved=0ahUKEwiK6ZrewdzLAhWDsYMKHcTbC844ChDoAQgaMAA#v=onepage&q=checkpoint%20base%20de%20datos&f=false
- [2] P. Tolmos Rodríguez-Piñero, (25-03-2015), *Introducción a los algoritmos genéticos y sus aplicaciones* [En línea]. Disponible: <http://www.uv.es/asepuma/X/J24C.pdf>
- [3] E. Yolis, (26-08-2015), *Algoritmos genéticos aplicados a la categorización automática de documentos* [En línea]. Disponible: <http://laboratorios.fi.uba.ar/lsi/yolis-tesisingenieraiinformatica.pdf>
- [4] C. L. Pacheco Agüero, (26-08-2015), *Distribución Óptima de Horarios de Clases utilizando la técnica de Algoritmos Genéticos* [En línea]. Disponible: http://jupiter.utm.mx/~tesis_dig/6557.pdf
- [5] J. D. Johnston Barrientos, (26-08-2015), *Aplicación de algoritmos genéticos para la asignación de carga académica en instituciones de educación superior* [En línea]. Disponible: <http://cdigital.dgb.uanl.mx/te/1020130069.PDF>
- [6] M. Utesch, (27-05-2016), *Optimización Genética de Consulta en Sistemas de Base de Datos* [En línea]. Disponible: <http://es.tldp.org/Postgresqls/web/navegable/todopostgresql/geqo.html>
- [7] L. Araujo y C. Cervigón, “Algoritmos Genéticos,” en *Algoritmos evolutivos Un enfoque práctico*, 1.ª ed. Distrito Federal, México: Alfaomega, 2009, cap. 2, sec. 2.9, pág. 38.
- [8] The PostgreSQL Global Development Group, *PostgreSQL 8.4 Server Administration*. Fultus Corporation, 2009

Diagrama entidad-relación de la base de datos

