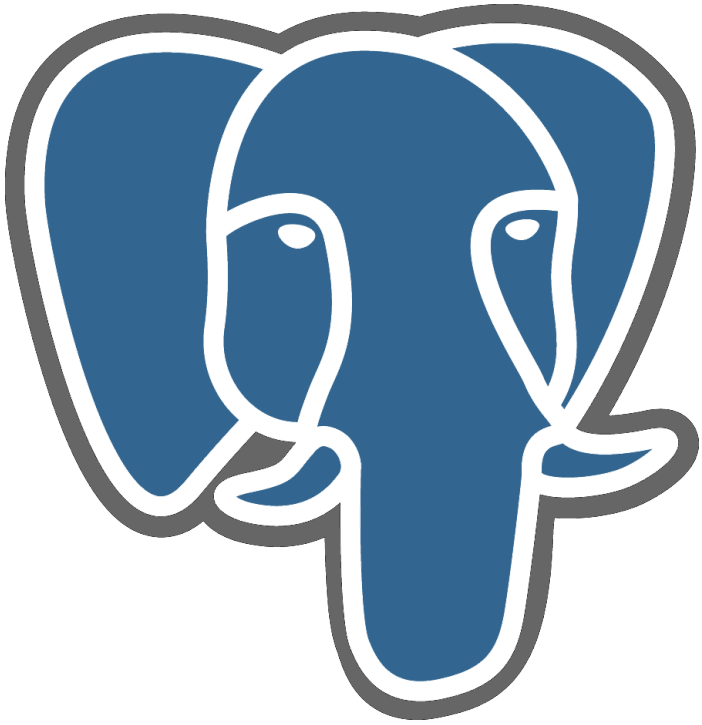

PostgreSQL

LA ALTERNATIVA A LOS SGBD PROPIETARIOS

Carlos Juan Martín Pérez
Facultad de Ciencias Exactas y Naturales
Escuela de Informática
Universidad Nacional de Costa Rica
Heredia, Costa Rica
email: cmartin@una.ac.cr

Red Costarricense de Software Libre

Índice de contenido

ABSTRACT

Nowadays, the core of almost the whole existent information systems is based on a relational Database Management System (RDBMS). There are many well known proprietary options in the software market that can fulfill this need, nevertheless in the present there also exist free options that, due to their reduced advertisement capacity, are noticeably unknown, at least in our context. This article describes the characteristics of the RDBMS PostgreSQL as an alternative to more extended proprietary RDBMS in features and strengthness to suit any kind of demand.

KEYWORDS

Relational database management systems, free software.

RESUMEN

Hoy en día, el núcleo de la práctica totalidad de los sistemas de información existentes consiste en el uso de un Sistema Gestor de Bases de Datos (SGBD) relacionales. En el mercado del software existen múltiples opciones propietarias bien conocidas para suplir esta necesidad, sin embargo en el presente también existen opciones libres que, por su escasa capacidad publicitaria, son notablemente desconocidas, al menos en nuestro entorno. Este artículo describe las características del SGBD PostgreSQL como alternativa en prestaciones y robustez para cualquier tipo de demanda a los SGBD propietarios más extendidos.

PALABRAS CLAVE

Bases de datos relacionales, software libre.

1. INTRODUCCIÓN

Aunque se aplica a muchos campos, pero principalmente en el mundo del software de gestión, la elección del Sistema Gestor de Bases de Datos (SGBD) es una circunstancia que tiene una trascendencia crucial para el éxito sostenido de un sistema informático, cualquiera que sea su tamaño.

Durante nuestra formación, tanto académica como informal o autodidacta, nos suelen introducir conceptos profundamente erróneos al respecto de qué SGBD es el idóneo según la magnitud del proyecto, habitualmente inducidos por la excelente publicidad que los SGBD propietarios más afamados proyectan en nuestro entorno, es decir, los productos de ORACLE® y Microsoft® SQL Server®... sinceramente, ¿cuántas de las maravillosas prestaciones, las cuales se pagan centavo a centavo de dólar de los EEUU (país donde residen las casas matrices de ambos productos), realmente son explotadas como para que la inversión valga la pena?: en el 99% de los casos, apenas utilizamos los servicios básicos de un gestor de bases de datos relacionales estándar.

En efecto el el precio es un obstáculo en la adquisición y mantenimiento de nuestro SGBD, y desafortunadamente la publicidad de las casas de software propietario es una cortina que se nos extiende para tapar otras opciones perfectamente viables desde todo punto de vista, más aún cuando su costo es TOTALMENTE CERO, hoy y siempre.

Tenemos también que mencionar otros impedimentos indirectos son las plataformas de hardware y el sistema operativo para soportar el SGBD, en los que al fin y al cabo también debemos invertir.

La Universidad de El Salvador (UES) tiene una experiencia acumulada de más de 5 años en el uso de PostgreSQL[1] como SGBD principal, esperamos que lo que les vamos a mostrar a continuación les sea de utilidad para poder tomar una futura decisión al respecto.

1 http://www.postgresql.org

2. Descripción General

PostgreSQL es un SGBD objeto-relacional libre, su licencia es de tipo "BSD" y por tanto la sostenibilidad del esfuerzo está asegurada. Comparándolo con otros SGBD's libres como MySQL, Ingres o Firebird, dada la amplitud de posibilidades que PostgreSQL ofrece, es sin duda el SGBD que más se asemeja en versatilidad a los SGBD's propietarios como Oracle®, Sybase®, DB2® o SQLServer®.

Al igual que otros proyectos de software libre, como GNU/Linux, Apache, etc. el desarrollo de PostgreSQL no es controlado por una compañía, sino que descansa en una comunidad de desarrolladoras y varias compañías (Sun, Fujitsu, Skype o RedHat, por citar las más conocidas).

3. Estandard SQL

PostgreSQL cumple en general (y mucho mejor que otros SGBDs) con los estándares de SQL (los ANSI-SQL 92 y 99, incluso las últimas versiones ya contemplan parte del SQL 2003), sin embargo se debe mencionar que en algunas sentencias extiende las capacidades de dicho estandard y/o carece de alguna de las funcionalidades definidas en el mismo, como cualquier otro SGBD propietario. Es sumamente importante revisar con detenimiento las exigencias en la programación del lado del SGBD, pues aunque toda dificultad es superable, estos detalles deben tenerse en cuenta para la realización de una migración exitosa así como la explotación de alguna característica ventajosa.

4. Límites

Los límites de PostgreSQL pueden ser resumidos en los siguientes puntos:

- Tamaño máximo de la base de datos: Ilimitado
- Tamaño máximo de una tabla: 32 Tb
- Tamaño máximo de una fila: 1.6 Tb

- Tamaño máximo de un campo: 1 Gb (2 Gb si es un BLOB)
- Número máximo de filas por tabla: Ilimitado
- Número máximo de columnas por tabla: de 250 a 1600, dependiendo de los tipos de las columnas.
- Número máximo de tablas o vistas en una base de datos: Ilimitado
- Número máximo de índices sobre una tabla: Ilimitado

5. Tipos de Datos

De forma nativa PostgreSQL soporta una amplia variedad de tipos de datos comunes, por ejemplo tipos numéricos de precisión arbitraria o limitada, secuencias, textos de longitud definida e ilimitados, fechas y horas de formato configurable según definición de localidad, tipos binarios de gran tamaño, que son almacenados y comprimidos aparte de los datos de un modo transparente al usuario, etcétera.

Además se dispone de algunos tipos de datos para aplicaciones específicas que resuelven excelentemente algunos problemas concretos como primitivas geométricas, direcciones IP, bloques CIDR, direcciones MAC y el más interesante, matrices cualquier tipo de dimensiones ilimitadas.

Y más aún: en PostgreSQL se pueden definir nuevos tipos de datos, totalmente indexables y con el conjunto de operadores y conversores que sea preciso, incluso sobrecargando los ya existentes. Un ejemplo de ésto es el proyecto PostGIS que implementa la infraestructura de base de datos necesaria para el soporte de un Sistema de Información Geográfica.

6. Acceso concurrente a los datos

PostgreSQL gestiona la concurrencia en el acceso a los datos mediante un sistema conocido como Control Multi-Versión de Concurrencia (MVCC), que proporciona a cada usuario una imagen (conocida en el argot como snapshot) de la base de datos, permitiendo realizar cambios a la base de datos sin que éstos sean visibles a otros usuarios hasta que la transacción es confirmada (en argot,

committed). Esto elimina radicalmente los bloqueos de lectura y asegura el mantenimiento de los principios conocidos como ACID[2] de una forma eficiente. Dicho sea de paso, este es el mismo mecanismo que usa Oracle® , en contraposición con mucho peor sistema de SQL Server® (hasta la versión 2005) o DB2® que usan bloqueo de filas (row locking). No obstante lo antedicho, está a disposición del usuario el uso de todo tipo de bloqueo de filas o tablas si es que el sistema MVCC no proporciona los resultados deseados en algún tipo de aplicación específica.

Por supuesto, dado que PostgresSQL es un SGBD ACID, soporta perfectamente el uso de transacciones, a fin de garantizar la atomicidad de las operaciones realizadas en varias consultas encadenadas. Dentro de las transacciones puede guardarse el estado temporal de los datos mediante el uso de "savepoints".

7. Mantenimiento de la Integridad de los datos

Una de las características más importantes de PostgreSQL es su completitud y fiabilidad al respecto de la manutención de la integridad de los datos, soportando claves primarias (simples y compuestas), restricciones de integridad referencial en forma de claves externas, con definición de comportamiento sobre actualizaciones o borrados (en cascada, restringida o asignación de nulo), restricciones de unicidad y todo tipo de comprobaciones a nivel de columna (no nulos y de contenido o "checks") y de fila.

8. Consultas

Algunos SGBD's, a pesar de ser más rápidos en cuanto a los tiempos de respuesta en consultas simples, carecen de mecanismos que son imprescindibles cuando se trata de explotar toda la potencia del lenguaje de consultas SQL. Estamos hablando de composiciones (joins) internos y externos, a la derecha o a la izquierda y, por supuesto, el manejo de subconsultas. En la versión 8.4 se añadió el soporte de la recursividad en las consultas[3], lo cual permite consultar de un modo muy elegante estructuras con relaciones de integridad referencial definidas sobre la propia tabla.

2 ACID: Atomicidad, Consistencia, aIslamiento, Durabilidad (ISO/IEC 10026-1:1992,secc.4)
3 http://www.postgresql.org/docs/8.4/static/queries-with.html

Otra de las mejoras notables es el manejo de funciones de ventanas sobre los resultados de una consulta (window functions[4]), ésto es similar a la funcionalidad de agrupación típica, pero respeta las características individuales de los resultados en una consulta en vez de ofrecer una fila por resultado agrupado.

En cuanto a la capacidad del motor de búsqueda, la implementación actual del optimizador de consultas utiliza un algoritmo que prácticamente alcanza un resultado cuasi-óptimo en el reordenamiento de las composiciones en las consultas. Sin embargo este algoritmo, en casos donde el número de composiciones es grande, puede tardar una cantidad excesiva de tiempo, lo que lo hace muy inapropiado para consultas que vinculan un gran número de tablas. Para resolver este inconveniente está a disposición el Optimizador Genético de consultas (GEQO), el cual resuelve este problema utilizando este tipo de paradigma.

Además de soportar búsquedas de texto basada en expresiones regulares, existen módulos especiales para la búsqueda y extracción de texto que cumpla ciertas condiciones, puntuándolo en función de su relevancia. Asimismo también existen extensiones que habilitan la búsqueda sobre XML.

9. Seguridad de Acceso

PostgreSQL tiene la capacidad de controlar el acceso al SGBD según usuario/grupo de usuarios, base de datos a la que se conecta e IP (tanto IPv4 como IPv6) y la autenticación así como la transferencia de datos es encriptada con SSL.

En lo relativo a la definición de permisos sobre los objetos de la base de datos, PostgreSQL permite definir permisos sobre los objetos para usuarios y grupos de usuarios (roles). Los objetos que pueden ser controlados son tablas completas, bases de datos, espacios de almacenamiento físico (tablespaces), funciones, lenguajes, esquemas y manejo de otros usuarios. En la versión 8.4 se añadió el soporte de seguridad a nivel de columna.

4 http://www.postgresql.org/docs/8.4/static/tutorial-window.html

10. Triggers, Reglas y procedimientos almacenados

Como en los restantes SGBDs profesionales, en PostgreSQL se pueden introducir procedimientos y funciones almacenadas para ser ejecutadas en el propio SGBD. El lenguaje nativo para la definición de estos objetos el PL/pgSQL, un lenguaje ciertamente similar al PL/SQL de Oracle®, sin embargo la aridez de este tipo de lenguajes así como la falta de flexibilidad para el control del flujo del procedimiento (sentencias de iteración y ramificación) ha potenciado la adopción de otros lenguajes dentro la definición de funciones, y hay para todos los gustos:

- Lenguajes interpretados: PL/Perl, pl/PHP, PL/Python, PL/Ruby, PL/sh, y PL/Tcl
- Lenguajes compilados: C, C++ y Java (PL/Java)
- Lenguajes de procesamiento estadístico: R (PL/R)

Las funciones pueden ser ejecutadas con los privilegios del usuario que las llama o bien con los del usuario que definió la función. Los triggers que soporta PostgreSQL realizan una función que pueden ser definida en cualquiera de los lenguajes antes mencionados, y por supuesto pueden ser ejecutados antes o después de cualquier tipo de actividad sobre una tabla o vista, objetos los cuales también pueden tener varios triggers definidos sobre ellos. El uso de sentencias de SQL dinámico dentro de un procedimiento definido en PL/pgSQL está totalmente soportado y no hay ningún problema para la realización de consultas sobre la tabla donde los datos están actualizándose (datos mutantes).

Una característica de PostgreSQL muy interesante son las denominadas “reglas”, que permiten literalmente transformar una consulta o bien añadirle algún tipo de funcionalidad adicional. Por último cabe destacar que PostgreSQL implementa un sistema de eventos que permite la intercomunicación de mensajes desde un cliente, procedimiento o trigger a cualquier otro cliente conectado a la base de datos que haya definido estar a la escucha de tal mensaje. De esta forma pueden programarse muy fácilmente aplicaciones de monitoreo de datos sin tener que estar consultando repetidamente la base de datos hasta cumplir la condición deseada. También es perfectamente factible la transmisión de datos de una base de datos a otra, tanto en el mismo servidor como en otro remoto, en una consulta en SQL.

11. ÍNDICES

Uno de los mecanismos más importantes en las bases de datos relacionales son los índices, los cuales aceleran infinitamente las consultas de mayor relevancia sobre los datos almacenados. En PostgreSQL el usuario puede definir los índices que necesite, bien usando los tipos habituales (B-trees y hashs) o bien utilizando la tecnología GiST[5] para tipos de datos o mecanismos de búsqueda específicos. Los índices de PostgreSQL tienen las siguientes características:

- **Escaneo adelante/atrás**, no es necesario definir un índice adicional para acelerar búsquedas ordenadas descendentemente.
- **Índices basados en expresiones.** Un índice puede ser creado con el resultado de una expresión o función, en vez de usar simplemente el valor de una columna.
- **Índices parciales**, que sólo indexan parte de los datos una tabla según una expresión tipo "WHERE ..." , esto permite definir índices más pequeños y por tanto más rápidos sobre una parte de los datos.
- **Índices de estrategia adaptable**: GiST (Árboles de búsqueda generalizados) y GIN (índice generalizado invertido). Éste último además de poder adaptar la algoritmia de búsqueda mantiene una estructura de clave-lista de ocurrencias, lo que lo hace idóneo para el uso sobre tipos de datos complejos como arrays.

12. HERENCIA

Habíamos mencionado que PostgreSQL es un SGBD **objeto**-relacional. En efecto, se incorpora el concepto de herencia propio de la Orientación a Objetos, permitiendo que una tabla pueda heredar sus características de una o varias tablas padre (en efecto, soporta herencia múltiple). Automáticamente PostgreSQL se encargará de "compartir" los datos entre las tablas emparentadas, insertando y borrando tuplas en la tabla padre cuando se inserten o borren en la tabla hija. También, la adición de una columna en la tabla padre implicará su aparición en la tabla hija.

5 GiST, Generalized Search Tree, o Árbol de Búsqueda Generalizado, es una estructura de datos y librería de funciones que permite construir prácticamente cualquier tipo de árbol de búsqueda sobre practicamente cualquier tipo de datos

Aunque este concepto tiene mucho desarrollo por delante, la implementación actual ya facilita en buena medida el trabajo con tablas que estén modelando una traducción de las especializaciones y generalizaciones que se definen en los esquemas de modelado conceptual Entidad Relación Extendido (EER). En la práctica el concepto de herencia permite implementar el particionamiento de tablas, pudiendo hacer más eficiente la búsqueda sobre cualquier tabla hija o directamente sobre la tabla padre.

13. Respaldo, recuperación y replicación

Aparte de los respaldos simples, PostgreSQL implementa la técnica de escritura adelantada en bitácoras sucesivas (WAL o Write Ahead Log) para reflejar cualquier cambio en la base de datos. Esto permite poder recuperar la base de datos completamente ejecutando los cambios de la bitácora sobre una imagen de los datos. Este procedimiento tiene los siguientes efectos beneficiosos:

- No necesitamos una imagen de la base de datos perfectamente consistente como punto de inicio, pues cualquier inconsistencia se corregirá al ejecutar los cambios de las bitácoras, de este modo no necesitamos más que una herramienta de compresión simple para archivar el backup.
- Dado que podemos juntar todas las bitácoras de cambios indefinidamente, se puede respaldar de un modo continuo archivando las bitácoras, lo cual es especialmente valioso en bases de datos masivas, donde es prácticamente inviable en tiempo y en inversión en medios de almacenamiento para realizar respaldos completos de la base de datos con mucha frecuencia.
- No es necesario ejecutar todas las bitácoras hasta la última guardada, se puede detener el proceso de ejecución de las bitácoras en cualquier punto y así tener una imagen consistente de la base de datos como si estuviera en ese momento. Implícitamente aseveramos el soporte de una recuperación de tipo “point-in-time”, dado que es posible restaurar la base de datos a cualquier momento en el tiempo desde que el respaldo base fue tomado.
- Si alimentamos de un modo continuo la serie de bitácoras a otro servidor que haya sido cargado con el mismo backup base, tendremos un sistema de respaldo “en caliente”: en cualquier momento podemos levantar el segundo servidor y tendremos una copia prácticamente gemela de la base de datos.

Además de los antedichos mecanismos de respaldo y restauración, están a disposición libre el uso de herramientas de replicación que habilitan el balanceo de carga y estructuras de bases datos de alta disponibilidad tanto síncrona como asíncrona, en forma multi-maestro (PgCluster, Bucardo respectivamente) o bien maestro/esclavo (Slony-I)

14. Universidad de El Salvador, El Salvador

En la Universidad de El Salvador existen en la actualidad 15 servidores que soportan los sistemas de gestión de la Administración Académica y otros sistemas administrativos. Ninguno de ellos supera los $2,000 y su configuración básica es Debian GNU/Linux como sistema operativo, Apache2, PHP y PostgreSQL como base de datos.

Con esa configuración se ha podido dar respuesta al máximo nivel posible de la demanda de servicios informáticos al estudiante, teniendo bases de datos de más de 2Gb de datos y tablas de más de un millón de filas, obteniendo siempre un optimo desempeño en un hardware ciertamente modesto.

Hasta el momento presente PostgreSQL ha satisfecho ampliamente sus necesidades, estando en todo caso de nuestra parte el uso de todas las prestaciones que como SGBD nos ofrece, es por esto que aconsejamos vehementemente considerar este SGBD como una opción sostenible para el almacenamiento de datos en cualquier institución o empresa. Es pertinente reseñar que existen grandes instituciones y empresas usando PostgreSQL, http://www.postgresql.org/about/users. Si bien no se indican en la anterior página, el core de negocio de Skype[6], la conocida red social Hi5[7], así como la que Yahoo proclama como mayor base de datos del mundo (2 petabytes) está basada en PostgreSQL[8].

6 https://developer.skype.com/SkypeGarage/DbProjects/SkypePostgresqlWhitepaper
7 http://www.computerworld.com.au/index.php/id;233796544
8 http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyId=18&articleId=9087918&intsrc=hm_topic

15. Sociedad de Seguros de Vida del Magisterio Nacional, Costa Rica

En esta institución se tiene una única base de datos segmentada en 13 esquemas. En dichos esquemas se tienen los objetos típicos de un SGBD relacional: tablas, restricciones, índices, secuencias, vistas y objetos programados en el lenguaje PL/SQL. Para tener una idea de la envergadura de la base de datos podemos señalar que tiene 635 tablas y un tamaño total de 11.3 Gb de datos en texto plano.

Es destacable la existencia de una tabla histórica de casi 23 millones de registros (3 Gb de datos). Todas las tablas hacen uso de clave primaria y, en algunos casos, se definen adicionalmente restricciones de unicidad, aparte de múltiples índices definidos sobre uno o varios campos de cada tabla y numerosas restricciones de integridad referencial entre las diferentes tablas, configurando así un esquema relacional de complejidad bastante alta.

Sobre tal base de datos se hicieron pruebas de desempeño en comparación con el SGBD Oracle® actual, obteniéndose resultados altamente positivos en el siguiente entorno:

- **Plataforma actual en producción en servidor real:** servidor Xeon doble Quad Core 64 bits 3.40 Ghz, 16 Gb RAM, almacenamiento por canal de fibra óptica en un HP EVA 4400 (max. 400 MB/s). Sistema Operativo Windows 2003 Server R2.
- **Plataforma de pruebas:** servidor virtual Xeon Quad Core 64 bits 3.40 Ghz, 8 Gb RAM, por canal de fibra óptica en un HP EVA 4400 (max. 400 MB/s). Sistema Operativo GNU/Linux.

Todas las pruebas de rendimiento en la plataforma de producción fueron realizadas en horario no laboral para garantizar la ausencia de carga sobre el SGBD y, en el caso de las plataformas propuestas, son servidores dedicados ex-profeso para estas pruebas, por lo que no existe ninguna otra actividad que pudiera alterar los resultados. Las consultas fueron lanzadas directamente en las respectivas terminales de comandos de los SGBD para minimizar cualquier tipo de interferencia por el procesamiento en el lado cliente.

Consideramos realizar una consulta que pusiera a prueba la capacidad de los SGBD para resolver la operación más costosa (una composición o join) donde estuviera involucrada la tabla más voluminosa de la BD (clie.pagos_y_desembolsos_cl, que tiene más de 22,7 millones de filas.

SELECT	
Plataforma	**Tiempo**
Oracle®	00:03:02
PostgreSQL	00:00:47

La consulta se repitió varias veces en cada plataforma, obteniendo tiempos similares. El resultado obtenido fue excelente a favor de la plataformas con PostgreSQL y GNU/Linux.

Para comprobar las prestaciones de los SGBDs en operaciones de inserción, modificación y eliminación de datos se escogió nuevamente la tabla anteriormente mencionada por ser la más densamente poblada para realizar las pruebas. A fin de que las pruebas fueran comparables se garantizó que todas las estructuras de indexado y disparadores (triggers) fueran equivalentes y estuvieran convenientemente activadas en cualquiera de las plataformas, para lo cual se realizó la oportuna adaptación del único disparador sobre dicha tabla, el cual sin embargo realiza llamadas a diferentes funciones y procedimientos que realizan consultas sobre la misma y otras tablas.

En el caso de inserción se utilizó un conjunto de 21 sentencias *insert*, a sabiendas de que la primera sentencia tendría un mayor costo (por el acceso al diccionario de datos) y las demás reducirían su tiempo de ejecución.

INSERT			
Plataforma	**Tiempo 1ero**	**Mejor Tiempo siguientes**	**Peor Tiempo siguientes**
Oracle®[9]	70 ms	10 ms	40 ms
PostgreSQL	320 ms	2 ms	77 ms

9 La precisión del timing de Oracle es sólo hasta las décimas de segundo.

En el caso de sentencias *update*, se realizaron dos operaciones para comparar el rendimiento entre ellas. Dichas sentencias alteraban un total de 107 filas. Para evitar cualquier tipo de interferencia por el envío de mensajes a las respectivas terminales se comentó, tanto en el trigger de Oracle como en la función asociada al trigger en PostgreSQL, la elevación del mensaje de notificación que sustituía a la excepción dispuesta en el trigger original, pues dicho trigger no permitía actualizaciones, limitación contraproducente para nuestras pruebas.

UPDATE		
Plataforma	**Primera**	**Segunda**
Oracle®	250 ms	20 ms
PostgreSQL	167 ms	10 ms

En este caso apreciamos nuevamente ventaja de PostgreSQL frente a Oracle, sin embargo es mínima considerando la escala de tiempo utilizada (milisegundos).

Por último, para sentencias *delete*, se consideró que la ejecución de una operación era más que suficiente testimonio del rendimiento en general de los SGBDs a la hora de realizar los cambios. Dicha operación consistía en eliminar precisamente las 21 filas insertadas previamente, para lo que se hacía la correspondiente consulta de selección en la misma sentencia.

DELETE	
Plataforma	**Tiempo**
Oracle®	00:01:39
PostgreSQL	00:00:10

En este caso, con toda certeza influido por la realización de la consulta (a la vista de las diferencias entre las dos plataformas PostgreSQL), Oracle se desempeñó realmente mal en comparación con PostgreSQL.

En suma, podemos afirmar con total certeza que en el aspecto de rendimiento del Sistema Gestor de Base de Datos no existirá problema alguno en caso de realizar la migración a PostgreSQL, por el contrario, muy probablemente se percibirá una mejoría notable, más profunda aún cuanto más grandes sean los conjuntos de datos consultados/actualizados.

Una de las posibilidades que no es posible realizar con la versión disponible del SGBD Oracle (Standard) es particionar la tabla que hemos utilizado como prueba para acelerar en lo posible las consultas. PostgreSQL soporta el particionamiento de forma nativa mediante el uso de sus características orientadas a objetos (cada partición hereda las características de la tabla padre), por lo que consideramos apropiado al menos hacer la prueba de particionar la tabla y comparar el rendimiento con la consulta de selección antes descrita.

Los resultados finalmente fueron bastante apropiados, pues la misma consulta realizada sobre la tabla particionada se calculó en un tiempo similar al de la consulta sin particionar, a sabiendas de que las características de la consulta no utilizarían los índices definidos sobre el campo particionado, lo que representa el peor caso. Por tanto un particionamiento más atinado (realizado según alguna característica de sus clave primaria por ejemplo) podría sin duda acelerar notablemente las consultas más frecuentes.

16. Optimización

Por defecto, PostgreSQL tiene unos parámetros muy conservadores en el uso de los recursos del sistema (principalmente memoria RAM) a fin de coexistir razonablemente con otros servicios que pudieran tener bastante demanda de este recurso.

La literatura consultada[10] y según la experiencia de uso, recomendamos utilizar los siguientes valores de configuración para que PostgreSQL aproveche al máximo los recursos existentes:

- Editar /etc/sysctl.conf y añadir kernel.shmmax=valor[11]. El valor debe expresarse en bytes. Este parámetro controla el tamaño máximo de un segmento de memoria compartida.
- Ajustar los parámetros en el fichero de configuración de PostgreSQL (postgresql.conf) shared_buffers, work_mem, maintenance_work_mem y effective_cache_size.

10 http://www.revsys.com/writings/postgresql-performance.html
http://www.postgresql.org/docs/8.3/interactive/runtime-config-resource.html
http://www.ca.postgresql.org/docs/momjian/hw_performance
11 En caliente puede cambiarse con sysctl -w kernel.shmmax=valor

Parámetro	Valor
kernel.shmmax	1/3 de la RAM disponible (en bytes)
shared_buffers	1/4 de la RAM disponible (en Mb)
work_mem	64 Mb (empírico)
maintenance_work_mem	128 Mb (empírico)
effective_cache_size	1/2 de la RAM disponible (en Mb)

17. Costos y Beneficios

PostgreSQL, además de libre, es gratis, y el tiempo de adaptación de cualquier persona administradora de otro tipo de base de datos es realmente mínimo por la gran compatibilidad con otros SGBD's, así que para hacer un análisis costo-beneficio de este software no hay más remedio que responder a la pregunta ¿satisface mis necesidades REALES? (enfatizamos reales para distinguir éstas de las necesidades creadas por la publicidad o los ejecutivos de ventas). Si es así, el costo es menor que 0, es decir, supone un ahorro, puesto que usted va a invertir considerablemente menos en hardware que si lo hiciera con un SGBD propietario. Si además usted decide instalarlo sobre GNU/Linux, súmese al ahorro el costo del sistema operativo, que en el caso de servidores, es un gasto ciertamente importante.

18. Agradecimientos

Instituciones: Universidad de El Salvador, la Universidad Nacional de Costa Rica, la Sociedad de Seguros de Vida del Magisterio Nacional de Costa Rica.

Organizaciones: Grupo de Usuarios de GNU/Linux de la Universidad de El Salvador, Asociación Salvadoreña de Usuarios de GNU/Linux, Red Costarricense de Software Libre.

Personas: a todas y cada una de las personas que he tenido la suerte de conocer, pues gracias a ellas he comprendido lo poco que sé y lo mucho que aún me falta.

19. Referencias

- http://www.postgresql.org
  - http://www.postgresql.org/about/press/features84
  - http://www.postgresql.org/docs/8.4/interactive/index.html
  - http://www.postgresql.org/about/featurematrix
  - http://www.postgresql.org/about/advantages
  - http://pgfoundry.org/
- http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems
- http://troels.arvin.dk/db/rdbms/