REVISTA Universidad EAFIT
Vol. 43. No. 148. 2007. pp. 78-90

Evaluación del rendimiento
de los motores de bases de datos Mysql y Firebird

Mauro Callejas Cuervo
Ingeniero de Sistemas, Especialista en Ingeniería de Software, Tesista de Maestría en Ciencias Computacionales. Profesor, Facultad de Ingeniería, Escuela Sistemas y Computación, Universidad Pedagógica y Tecnológica de Colombia.
Director del Grupo de Investigación en Software (GIS).
Investigador principal de Proyecto Software Libre.
mauro.callejas@uptc.edu.co

Diego Alberto Rodríguez Vela
Ingeniero de sistemas y computación. Investigador, grupo GIS, Proyecto Software Libre, Universidad Pedagógica y Tecnológica de Colombia.
darvela@gmail.com

Recepción: 19 de agosto de 2007 | Aceptación: 25 de octubre de 2007

Resumen

En el presente artículo se hace una descripción de una serie de métodos y herramientas usados para la construcción de un programa que evalúa el rendimiento de los sistemas de gestión de bases de datos Mysql y Firebird, a través de la implementación de una aplicación genérica para el manejo de inventarios. Describe el proceso para crear el algoritmo llamado robot, cuya función principal es poblar las bases de datos; luego se explican los casos de prueba y, finalmente, se muestran los resultados obtenidos.

Palabras Clave

Rendimiento
Mysql
Firebird
Software libre
Sistemas de gestión de bases de datos SGBD

Performance assessment of Mysql and Firebird Database Management Systems

Abstract

The article describes a number of methods and tools used to design software to evaluate the performance of Mysql and Firebird database management systems, through a generic application of inventories management. The process to create a *robot* algorithm, aimed at populating databases, is described and obtained results are shown.

Key words

Performance
Mysql
Firebird
Free Software
Databases Management's Systems
DBMS

Introducción

La investigación que se expone a continuación se llevó a cabo a partir de la motivación de conocer con anterioridad a su utilización, el rendimiento de un sistema de gestión de bases de datos, DBMS (Pons *et al*., 2005, 6). El trabajo se implementó en su totalidad con herramientas de *software* libre (Stallman, 2002, 41) y bajo el paradigma de desarrollo de prototipos incrementales (Pressman, 2005, 51).

Una vez explicadas las condiciones técnicas preliminares desde las cuales partió el proyecto, se describe el programa en cuanto a la base de datos que se modeló y el algoritmo que se desarrolló para la carga masiva de información a las bases de datos; este último aspecto es el elemento esencial de aplicación.

Posteriormente se describen las pruebas realizadas y el entorno, tanto de *hardware* como de *software*, sobre el cual se ejecutaron; con base en esta información, finalmente, se muestra un consolidado de los resultados en cifras y gráficos.

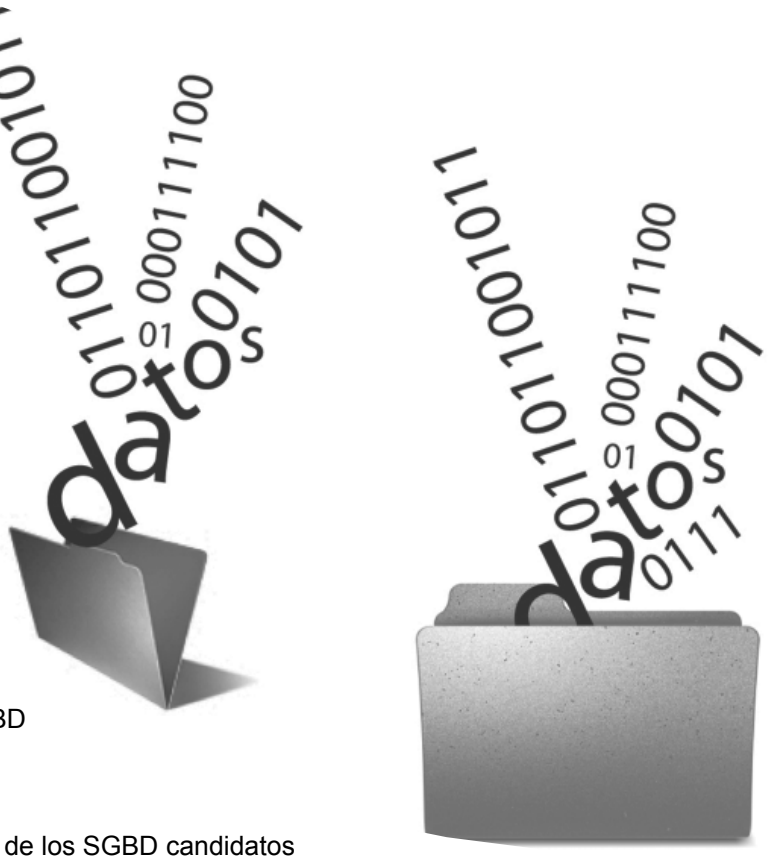
1. Métodos y herramientas usados para la construcción del *software* de evaluación

El tiempo de respuesta en las transacciones más comunes de una base de datos es un factor de productividad determinante en las empresas; por esta razón cobra especial importancia la posibilidad de medir dicha variable. En tal sentido se desarrolló un *software* capaz de medir el tiempo de respuesta en las transacciones propias del lenguaje de manipulación de datos LMD (Roman, 2002, 254), para los sistemas de gestión de bases de datos Mysql y Firebird. En particular, el *software* permite evaluar las sentencias *select* básico, *select* con funciones de grupo, *select* con condiciones de grupo, *insert, update, delete* y procedimientos almacenados; esta última depende del lenguaje procedimental que implemente cada herramienta. Asimismo, el *software* da la posibilidad de seleccionar los disparadores que se desea activar, y si se ejecuta sobre una única máquina local permite definir el número de conexiones concurrentes que se quieran simular. Si se ejecuta en un entorno de red real no es necesario efectuar dicha simulación. El *software*, que se desarrolló en Java J2SE (Melton & Eisenberg, 2002, 209; Trottier, 2004, 19), se conecta vía JDBC (Reese, 2000, 25; Liang, 2006, 1106) a una base de datos modelada para la gestión básica de inventarios e implementada en cada uno de los DBMS (Mysql, 2006, 163) analizados.

En la aplicación se desarrolló un algoritmo heurístico denominado Robot, que, con base en los principios fundamentales de la simulación y atendiendo las restricciones de integridad referencial de la base de datos (Rob y Coronel, 2004, 68), se encarga de poblar las tablas que conforman el modelo físico.

El *software* se ejecutó sobre plataforma Linux, pero, dado que las herramientas utilizadas tanto para la programación como para la base de datos son multiplataforma, es viable desplegar la aplicación en un entorno Windows.

Las características que la mayoría de motores de bases de datos incorporan típicamente, se constituyeron en el punto de partida para definir qué tipo de transacciones evaluar. En la Tabla 1 se muestra el resumen de las características de algunos SGBD.



La investigación se orientó hacia la evaluación de los motores Mysql y Firebird; el primero tiene gran reconocimiento, e incluso se integra como parte de algunas distribuciones de Linux, pero el segundo es poco conocido. En la Tabla 2 se observa un resumen de las características de cada uno de los SGBD analizados.

Tabla 1. Características fundamentales de los SGBD candidatos

SGBD	ACID	Integridad referencial	Transacciones	Unicode
DB2	Sí	Sí	Sí	Sí
Firebird	Sí	Sí	Sí	Sí
Informix	Sí	Sí	Sí	Sí
Ingres	Sí	Sí	Sí	Sí
InterBase	Sí	Sí	Sí	Sí
SapDB	Sí	Sí	Sí	Sí
MaxDB	Sí	Sí	Sí	Sí
MySQL	Depende	Depende	Depende	Sí
Oracle	Sí	Sí	Sí	Sí
PostgreSQL	Sí	Sí	Sí	Sí

Fuente: WIKIPEDIA. "Comparación de sistemas de bases de datos relacionales". http://es.wikipedia.org/wiki/Comparaci%C3%B3n_de_sistemas_administradores_de_bases_de_datos_relacionales

Tabla 2. Características de los SGBD seleccionados

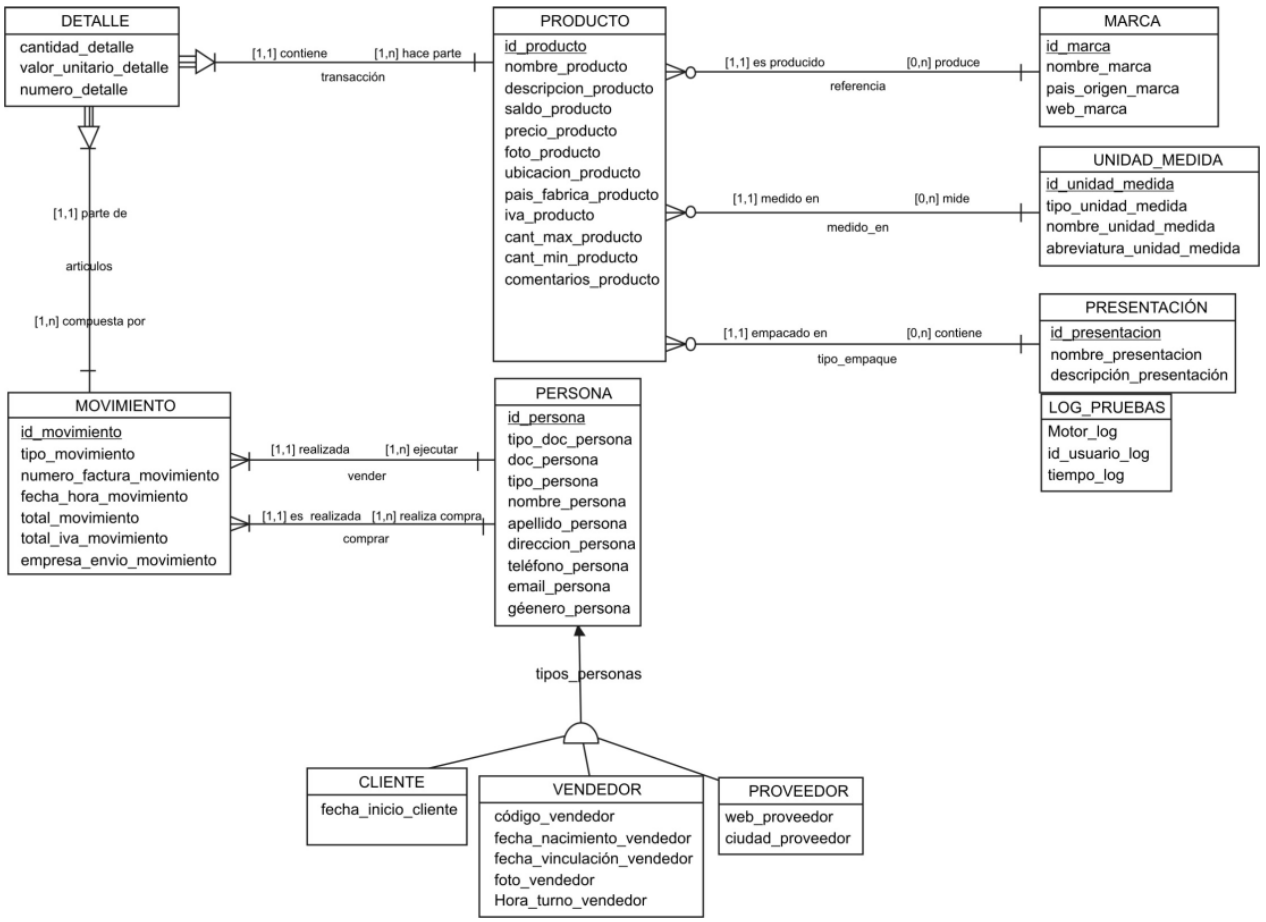
Característica	Interbase	Mysql
Soporte multiplataforma	X	X
Soporta transacciones	X	X
Generación automática de claves secuenciadles	X	X
Funciones definidas por el usuario	X	X
Herramientas de monitoreo del rendimiento	X	Limitadas
Procedimientos almacenados	X	X
Triggers	X	X
Vistas	X	X
Campos de tipo Check	X	X
Eventos	X	X
Integridad referencial declarativa	X	X
TIPOS DE DATOS		
Integer 16	X	X
Integer 32	X	X
Flota	X	X
Double	X	X
Numeric	X	X
Decimal	X	X
Char	X	X
Varchar	X	X
Text Blob	X	X
Bolob	X	X
Date	X	X
Time	X	X
TimeStamp	X	X
Bolean	X	X
PROTOCOLOS DE RED		
TCP/IP	X	X
NetBEUI	X	X
IPX/SPX	X	X

Fuente: WIKIPEDIA. “Comparación de sistemas de bases de datos relacionales”. http://es.wikipedia.org/wiki/Comparaci%C3%B3n_de_sistemas_administradores_de_bases_de_datos_relacionales

Se concibió la idea de desarrollar un programa capaz de evaluar el rendimiento de un par de SGBD en un ambiente cercano al entorno de producción real de una base de datos. Fiel a esta filosofía, se modeló un sistema genérico para la gestión de inventarios, el cual implementa los aspectos básicos que se deben

tener en cuenta en un sistema de este tipo, tales como gestión de productos, ventas, gestión de clientes y facturación, entre otros. En la Figura 1 se muestra en detalle el modelo Entidad-Relación (Barker, 1994, 49) de la aplicación. Una vez implementadas las bases de datos, tanto en Mysql como en Firebird, se dio comienzo a la etapa de desarrollo del *software* de pruebas.

Figura 1. Modelo entidad-relación del sistema de inventarios genérico



En las tablas 3 y 4 se muestra el diccionario de datos para las tablas producto y detalle, que son las de mayor peso en el algoritmo Robot, de carga automática de datos.

Tabla 3. Diccionario de datos para la tabla producto

Nombre tabla: PRODUCTO

Descripción: Cada una de las columnas de esta tabla describe las características de un objeto susceptible de comercializar.

Nombre	Codificación	Descripción
id_producto	Int.	Clave que identifica de forma única un producto.
id_marca_producto	Int.	Clave que identifica a qué marca corresponde el producto.
id_unidad_medida_producto	Int.	Clave que identifica a qué unidad de medida corresponde el producto.
id_presentacion_producto	Int.	Clave que identifica a qué presentación corresponde el producto.
nombre_producto	varchar(50)	Es la denominación del producto.
descripcion_producto	varchar(50)	Características más relevantes.
saldo_producto	Int.	Número de unidades en existencia.
precio_producto	Flota	Valor para la venta del producto.
foto_producto	Blob.	Imagen del producto.
ubicacion_producto	Varchar(100)	Lugar del negocio en el cual está almacenado el producto
pais_fabrica_producto	Varchar(25)	País en el cual se elaboró el producto.
iva_producto	Int.	Porcentaje de impuesto al valor agregado (IVA) que el producto debe pagar.
cant_max_producto	Int.	Máximo de unidades que puede haber en existencia.
cant_min_producto	Int.	Mínimo de unidades que puede haber en existencia.
comentarios_producto	Blob.	Observaciones importantes acerca del producto.

Fuente: Elaboración propia.

Tabla 4. Diccionario de datos para la tabla detalle

Nombre tabla: DETALLE

Descripción: Cada una de las columnas de esta tabla describe la venta de un único producto.

Nombre	Codificación	Descripción
id_producto_detalle	Int.	Número que identifica el producto vendido.
id_movimiento_detalle	Int.	Número que identifica el movimiento al que pertenece este detalle.
cantidad_detalle	Int.	Número de unidades que se venden del producto.
valor_unitario_detalle	float	Valor de un único producto.
numero_detalle	Int	Orden en el cual se registran los detalles.

Fuente: Elaboración propia.

Siguiendo recomendaciones y estándares plasmados en Sommerville (2005, 331) y Shneiderman (1992, 36), el programa se diseñó con una interfaz gráfica de usuario sencilla e intuitiva, que facilita el trabajo del usuario y la comprensión de los resultados que arrojan las pruebas ejecutadas. En la Figura 2 se muestra la consola de administración en la que el usuario administrador puede elegir el volumen de carga de información y los *triggers* que desea activar sobre la base de datos.

Figura 2. Interfaz de la consola de administración

Una vez configurados los parámetros iniciales mediante la consola de administración, es posible configurar y ejecutar un conjunto de pruebas que se definen mediante el formulario que muestra la Figura 3.

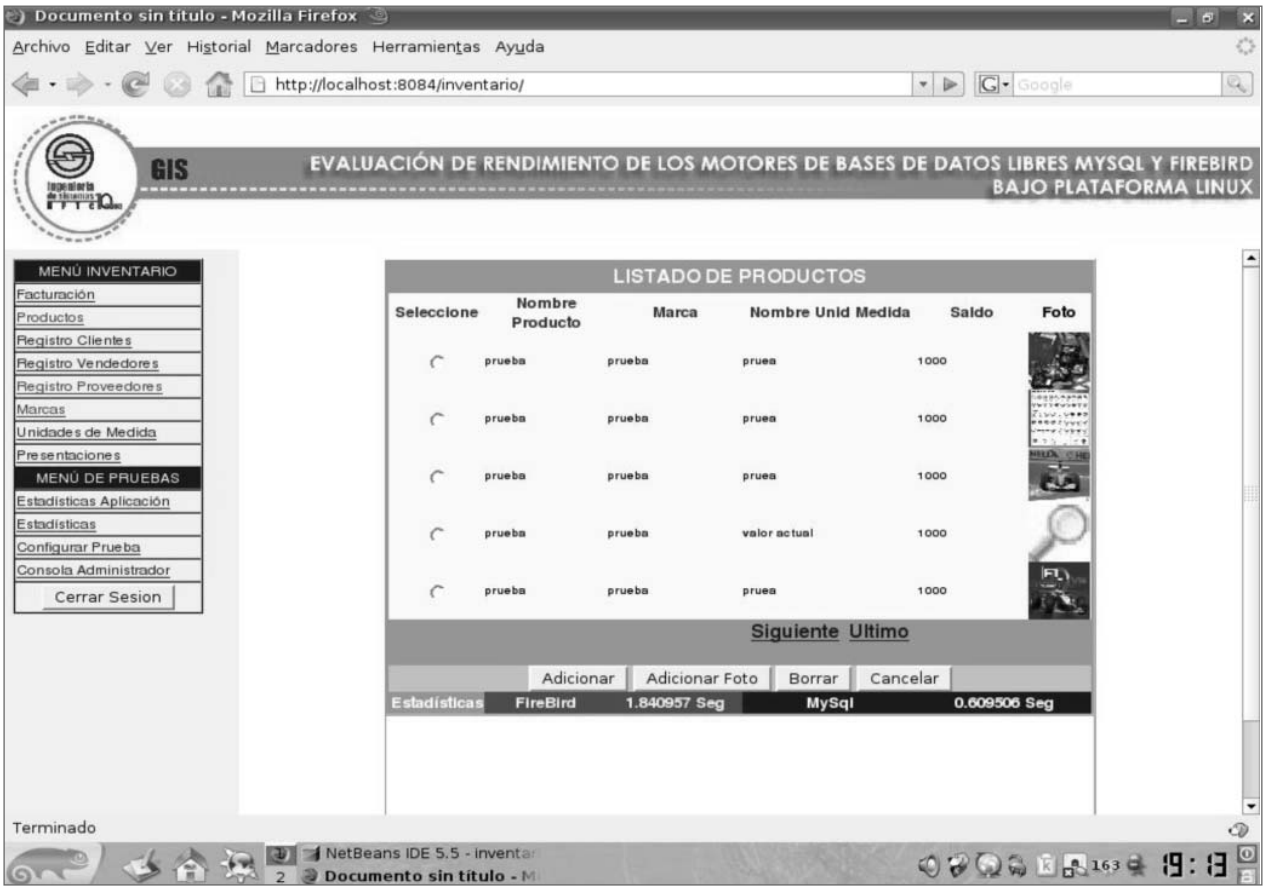
Figura 3. Formulario para configuración de pruebas

Además de las pruebas personalizadas, es decir, las que el usuario configura según su criterio, el programa brinda la posibilidad de ejecutar pruebas en tiempo de ejecución, que consisten en medir los tiempos que tarda en efectuarse cada una de las transacciones propias de un sistema de inventarios, tales como actualización, inserción o borrado de productos, marcas y clientes, entre otras, como se puede observar en la Figura 4. Al consultar el listado de productos, en su parte inferior se encuentra la barra de estadísticas, que presenta el tiempo acumulado para los dos motores en cada una de las transacciones ejecutadas.

El núcleo del programa contiene un algoritmo, aquí denominado Robot, cuya función principal es poblar de forma inteligente la base de datos con un volumen de carga de información definido por el usuario. A continuación se describe su funcionamiento.

Cada una de las tablas del modelo físico es poblada de acuerdo a un peso porcentual asignado según la importancia de la tabla; según esto, las tablas de mayor tránsito, como detalle, producto y movimiento, serán pobladas con mayor cantidad de registros, mientras que tablas como marca, por ejemplo, serán cargadas con un número menor de datos. Cada vez que se defina el número de registros por cada tabla, el algoritmo genera datos seudoaleatorios para las tablas que no precisan de registros relacionados. Posteriormente, las que sí los tienen son cargadas con registros que el Robot arrastra de las anteriores, con el fin de no cometer violaciones de integridad referencial dentro de la base de datos.

Figura 4. Listado de productos



El robot implementa un vigía que se encarga de controlar que la cantidad de registros insertados no exceda la inicialmente seleccionada por el usuario y que cada tabla sea poblada por la cantidad exacta de datos asignada de acuerdo al peso.

El tiempo de ejecución del programa crece proporcionalmente al número de registros con que se desee poblar el modelo; del mismo modo, se ve afectado por el *hardware* utilizado, en particular por la velocidad del disco duro. En las pruebas de campo, para un volumen de carga de un millón de registros el programa tardó seis horas y 32 segundos, trabajando con un disco duro de 5400 revoluciones por minuto.

Con el fin de simular varias conexiones simultáneas a los motores, se genera un hilo de ejecución por cada una de ellas. Esta programación multihilo tiene un minucioso control de los procesos, aplicado para evitar que los tiempos finales de las transacciones se vean afectados por la ejecución anticipada o errónea de un hilo.

2. Pruebas y resultados obtenidos

El programa inicialmente se ejecutó en un entorno local, es decir, tanto cliente como servidor se ejecutaron sobre una misma máquina; en este caso, un equipo con plataforma Linux Suse y un procesador Turion de 64 bits a 1,6 Ghz, con 1 Gb de memoria DDR, un disco duro de 80 Gb a 5400 revoluciones por minuto.

Las pruebas en entorno de red se realizaron utilizando como servidor la misma máquina de las pruebas locales; por su parte, los clientes corrieron desde máquinas Pentium 4 de 2,4 Ghz con una memoria de 256 Mb y disco duro de 40 Gb a 7200 revoluciones por minuto.

El programa requiere para su ejecución un navegador Web con soporte para Java, que la mayoría de los navegadores de la actualidad brindan, y aunque el desarrollo del programa se orientó a Firefox, también fue probado de forma exitosa en Opera, Mozilla e Internet Explorer. Igualmente, requiere Java Runtime Edition J2RE 1,5 o superior; no es viable el uso de una versión anterior, puesto que Java ofrece solo a partir de la versión mencionada soporte para trabajar con nanosegundos, que es la unidad de medida con la cual se efectúa el cálculo del tiempo de ejecución de transacciones en los dos motores para esta investigación. Del lado del servidor, se requiere Tomcat para desplegar la aplicación Web.

Luego de ejecutar varias veces el mismo conjunto de pruebas, como principal resultado en cuanto a la velocidad de los motores comparados, se obtuvo mejor rendimiento en Mysql, ya que en todas las pruebas demostró ser más veloz que Firebird, tal y como se aprecia en la Tabla 5 y en la Figura 5. En las pruebas en tiempo real se llegó a la misma conclusión (Figura 6). De cualquier forma, el estudio merece ser ampliado para evaluar más posibilidades, al mismo tiempo que se puedan implementar las características más avanzadas de cada uno de los motores.

El programa está abierto a ser ejecutado con cualquier par de SGBD que dispongan de un controlador JDBC (Harrison, 2006, 310) y que implementen el SQL estándar (Groff, 2002, 9).

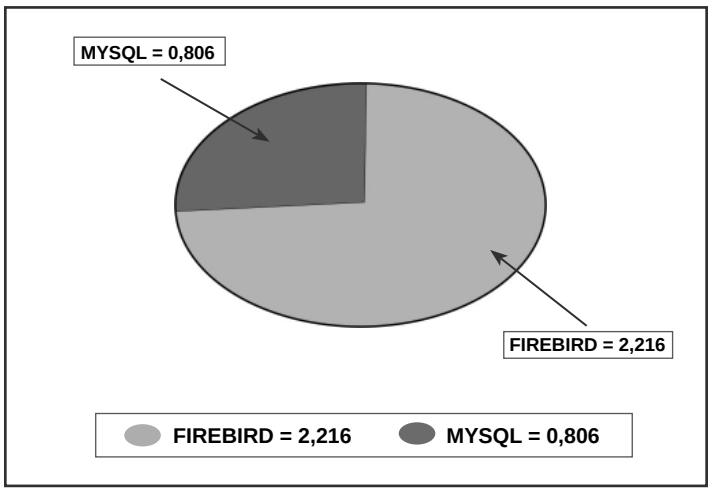
Tabla 5. Consolidado de pruebas, con 15 conexiones concurrentes

Nombre Prueba	No. Registros	Firebird Tiempo (ms)	Mysql Tiempo (ms)
funcion_avg	30	1,823	0,130
funcion_count	30	1,508	0,270
funcion_max	30	1,739	0,152
funcion_min	30	2,134	0,134
funcion_sum	30	1,597	0,172
grupo_groupby	87	1,412	0,080
grupo_having	30	2,082	0,146
grupo_orderby	33600	13,851	0,682
grupo_union	29	3,573	0,289
lmd_delete	30	0,508	0,153
lmd_insert	530	1,282	0,038
lmd_update	126	3,039	2,066
procedimiento_standar	26	0,911	0,092
select_anidado	30	0,567	0,332
select_basico	36000	8,633	6,181
select_distinct	90	3,179	0,083
select_join	90000	33,280	14,859

Fuente: Elaboración propia.

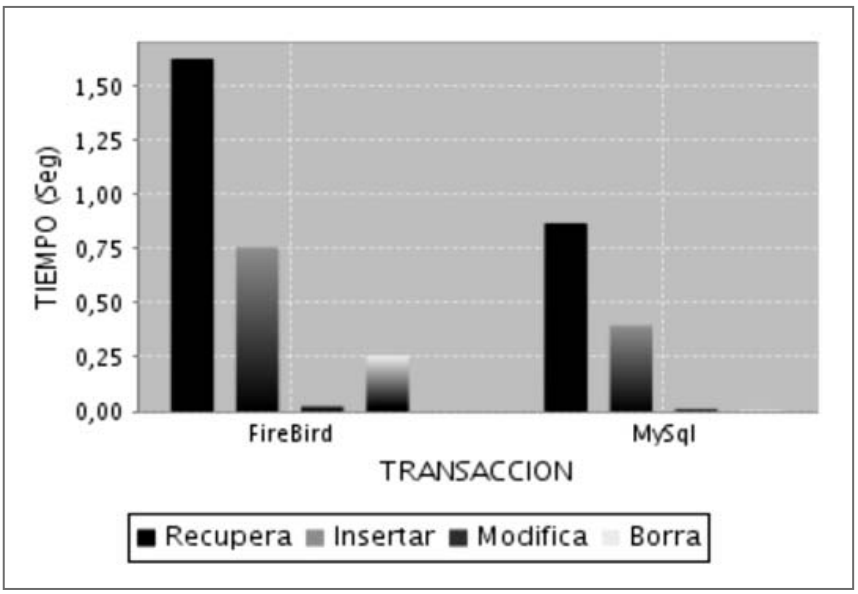
Los tiempos obtenidos en la tabla anterior se convirtieron a milisegundos para conseguir una cifra más informativa.

Figura 5. Consolidado de todas las pruebas



En la Figura 6 se aprecian los resultados logrados en tiempo real, discriminados por transacción: Recupera (*Select*), Inserta (*Insert*), Modifica (*Update*), Borra (*Delete*).

Figura 6. Resultados tiempo real



La arquitectura del programa desarrollado permite evaluar, con escasos ajustes, cualquier par de motores de bases de datos que dispongan de un conector JDBC. Sin embargo el alcance del proyecto se enfoca en Mysql y Firebird.

Al margen de los resultados alcanzados, se debe mencionar que durante el desarrollo de la investigación quedó de manifiesto que Firebird, a pesar de no ser tan veloz como Mysql, presenta otras características en las cuales supera al

segundo, como la seguridad, la gestión de usuarios, la riqueza del lenguaje procedimental que incorpora y la claridad en el concepto del manejo de disparadores y procedimientos almacenados; sin embargo, estas características no hacen parte del contexto del estudio y merecerían un análisis independiente más profundo.

Por otra parte, el desarrollo del Robot corresponde más que a un algoritmo elaborado para dar solución a un problema concreto, a una metodología que podría ser adaptada a procesos de migración de datos y a casos en los que se requiera poblar dos bases de datos simultáneamente.

Conclusiones

Dar un juicio absoluto acerca de si un motor de bases de datos es mejor que otro no es viable. Realmente, cada herramienta tiene fortalezas y debilidades que deben analizarse en el momento de decidir qué SGBD se va escoger, partiendo, eso sí, del criterio de potencializar las fortalezas que pueda tener cada motor de acuerdo al tipo de proyecto que se esté desarrollando; verificando, al mismo tiempo, que las posibles debilidades no sean elementos críticos en la aplicación que se va a desarrollar.

El motor Mysql resultó ser el mas rápido al ejecutar en menor tiempo todas las transacciones realizadas, superando a Firebird, el cual tardó en promedio un milisegundo (1 ms) más que su contraparte.

En la implementación de una aplicación de *software* de pruebas, es de vital importancia ejecutar algoritmos de generación automática de datos seudoaleatorios, si bien no resultó eficiente el trabajo con datos estáticos y cuyo comportamiento era perfectamente predecible; por tal razón, la aplicación de esta técnica esencial en la simulación fue el pilar fundamental en la culminación de esta investigación.

Desarrollar un estudio de este tipo a través de *software* libre trae ventajas no solamente académicas, sino económicas, ya que este tipo de *software* está a disposición del público en general con bajos costos.

Bibliografía

Barker, R. (1994). *El modelo entidad-relación Case*meted*. Wilmington: Addison Wesley/ Díaz de Santos, 231 p.

Groff, J. R. & Weinberg P. N. (2002). *SQL: The Complete Reference*. Berkeley: McGraw-Hill, 1025 p.

Harrison, G. & Feuerstein, S. (2006). *MySQL Stored Procedure Programming*. Sebastopol: O'Reilly, 636 p.

Liang, Y.D. (2006). *Introduction to Java Programming: Comprehensive.* E.U.A.: Pearson Prentice Hall, 1301 p.

Melton, J. & A. Eisenberg. (2002). *Understanding SQL and java together: a guide to SQLJ, JDBC, and related technologies*. San Diego: Academic, 470 p.

Mysql AB. (2006). *MYSQL Administrador's Guide and Languaje*. E.U.A.: Sams publishing, 860 p.

Pons, O. *et. al.* (2005). *Introducción a las bases de datos*. México: Thomson Learning Ibero, 286 p.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. New York: McGraw-Hill, 880 p.

Reese, G. (2000). *Database Programming with JDBC and Java*. Sebastopol: O'Reilly, 345 p.

Rob, P. y C. Coronel. (2004). *Sistemas de bases de datos: Diseño, implementación y administración*. México: Thomson Learning Ibero, 829 p.

Roman S. (2002). *Access Database Design and Programming*. Sebastopol: O'Reilly, 448 p.

Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human--computer Interaction*. E.U.A.: Addison-Wesley, 537 p.

Sommerville, I. (2005). *Ingeniería de Software*. Madrid: Addison Wesley, 687 p.

Stallman, R. (2002). *Free Software, Free Society: Selected Eassays of Richard M. Stallman*. Boston: Free Software Foundation, 224 p.

Trottier, A. (2004). *Java 2 Developer*. E.U.A.: Que Publishing, 399 p.

WIKIPEDIA. "Comparación de sistemas de bases de datos relacionales". http://es.wikipedia.org/wiki/Comparaci%C3%B3n_de_sistemas_administradores_de_bases_de_datos_relacionales.