Universidad de las Ciencias Informáticas

Facultad 6

Título: Propuesta de solución para la transferencia de información en la plataforma alasGRATO.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Bárbaro Leduar Sierra Romero.
Angel Daniel Bravo Consuegra.

Tutores: Ing. José Rolando Lafaurié Olivares.
Ing. Julio Omar Prieto Entenza.
Ing. Julio Antonio Villaverde Martínez

Ciudad de La Habana, junio, 2010

"La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia."

John Ruskin

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Bárbaro Leduar Romero Sierra

_____

Firma del Autor

Angel Daniel Bravo Consuegra

_____

Firma del Autor

Ing. Julio Omar Prieto Entenza

_____

Firma del Tutor

Ing. José Rolando Lafaurié Olivares

_____

Firma del Tutor

Ing. Julio Antonio Villaverde Martínez

_____

Firma del Tutor

DATOS DE CONTACTO

Tutores:

Julio Omar Prieto Entenza: Ingeniero en Ciencias Informáticas, Profesor con 4 años de experiencia.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Email: jprieto@uci.cu

José Rolando Lafaurie Olivares: Ingeniero en Ciencias Informáticas, Profesor Instructor con 1 año de experiencia.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Email: jrlafaurie@uci.cu

Ing. Julio Antonio Villaverde Martínez: Ingeniero en Ciencias Informáticas, Profesor Instructor con 1 año de experiencia.
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Email: jvillaverde@uci.cu

AGRADECIMIENTOS

A todos los profesores que a lo largo de la carrera nos han brindado su apoyo y ayuda para lograr este sueño de convertirnos en ingenieros.

A nuestros tutores José Rolando y Villaverde por su ayuda en el desarrollo de este trabajo.

A nuestro tutor Julio Omar que fue el que nos guió todo el tiempo y sin él sería imposible la realización de nuestro trabajo.

A todos nuestros familiares y amigos por la ayuda incondicional y por el apoyo brindado.

A nuestros padres por su apoyo incondicional.

A todos los que de una forma u otra contribuyeron y nos apoyaron en el desarrollo de este trabajo.

A todos muchas gracias.

DEDICATORIA

A mis padres, que han sido mi faro guía, mi ejemplo a seguir, mi fortaleza, mi fuente de inspiración gracias por tener tanta confianza en mí, apoyarme en mis decisiones y por ese amor sin límites.

A mi hermano Yusney por su apoyo y preocupación.

A toda mi familia en general, por todo su apoyo.

A todos mis amigos y compañeros que han compartido junto conmigo estos 5 años que han hecho de verdad mi vida más interesante... a todos muchas gracias...

Bárbaro Leduar Sierra Romero

A mi mamá y mi papá por todo su amor y comprensión, por ser lo más importante de mi vida. A mi abuela por una niñez tan maravillosa. A mi hermana de la vida por todos los momentos que hemos pasado juntos. A toda mi familia en general, por todo su apoyo. A todos mis amigos y compañeros que han compartido junto conmigo estos 5 años que han hecho de verdad mi vida más interesante... a todos muchas gracias...

Angel Daniel Bravo Consuegra

RESUMEN

Está investigación surge en el trabajo del proyecto de investigación Predicción de Actividad Biológica de Compuestos Orgánicos (GraphTool) desarrollado en la Universidad de las Ciencias Informáticas (UCI). Debido a la delicada situación en la que se encuentra nuestro país a la hora de realizar transferencias de datos en la red y, con el objetivo de apoyar a numerosos proyectos y de esta manera contribuir al desarrollo tanto económico como investigativo utilizando bases de datos empotradas en los diferentes proyectos. En el cual hizo necesario realizar una fundamentación teórica acerca de las diferentes características de las bases de datos embebidas que existen en el mundo para su posterior estudio; donde se realizó un plan de pruebas que aborda la metodología a seguir tanto en el diseño como en la aplicación de los diferentes casos de pruebas aplicados a las bases de datos seleccionadas. Posteriormente se decidió incorporar la base de datos seleccionada al módulo Cálculo de Descriptores.

Palabras claves

Bases de datos empotradas.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La bioinformática es una nueva disciplina que surge como respuesta al creciente aumento de los volúmenes de datos biológicos y la facilidad de compartir esta información a través de Internet. Por su esencia es multi e interdisciplinaria pues se nutre de la biología, la bioquímica, la química, la informática y las tecnologías de la información con vista al análisis, organización y distribución de toda esa información relacionada. Esta confluencia ha permitido grandes avances en las investigaciones, como por ejemplo, en el diagnóstico, tratamiento y prevención de diversas enfermedades que puedan mejorar la calidad de vida. Es así que el desarrollo de la industria farmacéutica guarda una estrecha relación con la bioinformática, la cual le permite la búsqueda, cada vez más eficiente, de nuevos fármacos para el tratamiento de diversas enfermedades.

Un aporte importante a estos estudios lo constituyó el desarrollo paralelo, tanto de las técnicas de la computación como del hardware lo cual, no sólo aceleró el avance de la disciplina, sino que incrementó la calidad de los resultados. En la actualidad, es posible encontrar en Internet listados de software aplicados al diseño racional de fármacos, que permiten el ahorro tanto de tiempo como de otros recursos en la fase inicial de búsqueda de nuevos agentes biológicamente activos.

Por otra parte, la Facultad 6 de la Universidad de las Ciencias Informáticas (UCI) desarrolla el proyecto CITMA de investigación-desarrollo-formación denominado: "Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos" (alasGRATO), el cual tiene entre sus objetivos, predecir la relación estructura-actividad de compuestos orgánicos con la utilización de diferentes técnicas de inteligencia artificial a partir de la descripción de la molécula mediante diferentes procedimientos.

Este proyecto se concibió, desde sus inicios, con una arquitectura cliente/servidor que fuese capaz de brindar servicios a los diferentes centros fuera del contexto UCI como por ejemplo, la Red del Ministerio de Educación Superior y centros del Polo Científico del Oeste. Sin embargo, debido a los problemas existentes en el desarrollo de la informática en el país, sobre todo en la velocidad de transmisión de datos entre centros como consecuencia del poco ancho de banda existente, se hace necesario la búsqueda de diferentes procedimientos para que dicha transmisión entre los diversos clientes y la UCI se vea afectada lo menos posible, y de esta forma garantizar el correcto desarrollo de los diferentes proyectos asociados.

Lo anteriormente planteado conduce al siguiente **problema científico** de cómo contribuir a la transferencia de datos entre la plataforma alasGRATO y centros externos al mismo. Por lo que se define como **objeto de estudio** es la transferencia de datos, definiendo como **campo de acción** transferencia de datos mediante bases de datos embebidas.

Se formula como **objetivo general** que se persigue con la realización de este trabajo transferencia de datos en la plataforma alasGRATO haciendo uso de bases de datos embebidas. Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- ✓ Seleccionar las bases de datos adecuadas.
- ✓ Definir el diseño de los casos de prueba.
- ✓ Realizar los casos de pruebas.
- ✓ Evaluar los resultados obtenidos.
- ✓ Incorporar la base de datos embebida al módulo Cálculo de descriptores.

Para darle solución al problema científico planteado y lograr el cumplimiento de los objetivos se proponen las siguientes **tareas**:

- ✓ Realización de una investigación sobre las bases de datos empotradas.
- ✓ Análisis de los tipos y métodos de pruebas que se realizan para medir el rendimiento de las bases de datos empotradas.
- ✓ Realización de los diseños de casos de pruebas.
- ✓ Análisis del entorno donde se realizaran las pruebas.
- ✓ Estudio de las herramientas de software que soportan el proceso de pruebas.
- ✓ Realización de los casos de pruebas.
- ✓ Evaluación del rendimiento de las bases de datos.
- ✓ Análisis de los resultados de las pruebas.
- ✓ Incorporación de la base de datos empotrada al módulo.

El presente trabajo se encuentra estructurado por tres capítulos y anexos, en los cuales se exponen todo el trabajo investigativo y prácticas realizadas.

Capítulo 1: Fundamentación Teórica.

En este capítulo se aborda fundamentalmente el tema de las bases de datos empotradas que existen en el mundo. Se presentan los conceptos esenciales relacionados con el tema, definiéndose una estrategia para la realización de los casos de pruebas.

Capitulo 2: Programas y procedimiento**.**

Este capítulo está dedicado fundamentalmente al estudio de los programas y metodologías seguidas para la realización de los casos de pruebas.

Capitulo 3: Resultados y discusión

Se realiza la evaluación del resultado de las pruebas realizadas, así como la comparación de los resultados de ellas y, se exponen los errores encontrados a partir de los resultados obtenidos; además se selecciona la base de datos a incorporar al sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se realizara un estudio sobre las bases de datos empotradas que existen en el mundo, realizándose así un estudio sobre las mismas. Se presentan los conceptos esenciales relacionados con el tema, definiéndose una estrategia para la realización de los casos de pruebas.

1.1 Bases de datos embebidas o empotradas

La informática es una herramienta de indudable valor para el desarrollo de investigación científica. Entre sus múltiples aplicaciones encontramos la gestión de bases de datos. Una base de datos es un conjunto de información perteneciente a un mismo contexto, que está almacenada en forma sistemática, de manera tal, que los datos que la conforman puedan ser utilizados cuando sea necesario. (1)

Las bases de datos empotradas se definen por no iniciar un servicio en nuestra máquina independiente de la aplicación, pudiéndose enlazar directamente a nuestro código fuente o bien utilizarse en forma de librería. (2)
Ellas comparten una serie de características comunes tales como: su pequeño tamaño, en términos de tamaño de código añadido a la aplicación, recursos que consumen, y que en general no están pensadas para el acceso multiusuario. (2)
Ellas en algunos casos, representan el primer paso hacia un servidor de bases de datos tradicional ,en otros casos su pequeño tamaño las hace ideales como soporte de información a los sistemas mono usuario que deban utilizar los recursos de la forma más eficiente posible. (3)

Una de las ventajas de tener una base de datos embebida es que no es obligatorio configurar la red o su administración ya que ambos cliente y servidor correr juntos en el mismo proceso. Esto reduce los gastos relacionados con la red en cuestión de llamadas, simplifica la administración de la bases de datos, y hace que sea más fácil desplegar la aplicación. (4)

En el mundo del software existen diversas bases de datos empotrables, entre ellas se pueden citar Berkeley DB, Daffodil DB, Perst, Metanotion BlockFile, Hypersonic SQL, Quadcap Embeddable Database, Gadfly, Metakit, Apache Derby, Db4o, H2, SQLite.

1.1.1 Apache Derby

Apache Derby es una base de datos de código abierto, sin soporte de administración, lo cual la hace ideal para desarrolladores que no necesitan de un sistema de bases de datos. Soporta datos de almacenamiento y de interrogación SQL, proporcionando así todas las funciones que se pueden encontrar normalmente en los sistemas grandes de base de datos. Como funcionalidad adicional, puede correr en modo cliente/servidor al igual que los grandes motores de base de datos.

Entre sus características fundamentales se encuentra que está basado en java e implementa estándares SQL y JDBC, es muy liviano (deja una huella de 2MB para el motor de bases de datos. (5)

Dentro de las ventajas de Derby tenemos sus mínimas necesidades de configuración y administración y el pequeño espacio que ocupa en disco. Otra ventaja es que Derby soporta el almacenamiento de una base de datos archivada en un archivo JAR, lo que lo hace fácil de incluirlo y distribuirlo en nuestra aplicación. (6)

1.1.2 SQLite

SQLite es un Software Libre que hace que su código fuente sea de dominio público, portando una licencia GPL (General Public License). Fue creado por D. Richard Hipp, el cual implementó una pequeña librería de aproximadamente 500kb, programado en el lenguaje C.

SQLite, permite que múltiples usuarios accedan en modo escritura a la base de datos, ya que posee un mecanismo de bloqueo muy basto. Cuenta con la utilidad que permite ejecutar comandos SQL contra una base de datos SQLite en modo consola. (7)

Tiene la capacidad de reemplazar los grandes motores de Bases de Datos y acoplarse al desarrollo de nuestros proyectos informáticos, ya sea en ambientes de prototipos de sistemas como en complejos y robustos software. (8)

Entre sus características fundamentales se encuentra que tiene una pequeña memoria y una única biblioteca para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas, realiza las operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL, es portable, es compatible con ACID (Atomicity Consistency Isolation and Durability), cuenta con diferentes

interfaces del API (Programming Interface), las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby, groovy. No posee configuración. (7)

1.1.3 H2

Es una base de datos relacional programada en Java, que se puede integrar completamente en nuestras aplicaciones y acceder a ella lanzando SQL directamente, sin tener que pasar por una conexión a través de sockets o utilizando conexiones JDBC externas. Lo cual hace posible una mayor velocidad y una mejor integración.

Cuando se utiliza dentro de una aplicación, una vez que ha sido abierta la base de datos para trabajar con ella, esta queda físicamente bloqueada, no permitiendo que otro motor H2 (en otra aplicación Java) pueda acceder a ella, lo cual permite mantener la integridad del fichero físico de la base de datos. (9)

Es una base de datos muy rápida y de código abierto, presenta un navegador de aplicación basada en consola, la huella del disco es cerca de 1 MB, el API de programación principal es SQL y JDBC, las tablas pueden ser persistentes o temporales, presenta conexiones en el modo del servidor de cliente y consola, y apoya la protección contra la inyección del SQL. (9)

1.1.4 Db4o

Es un novedoso motor de bases de datos orientados a objetos de código abierto, bajo la GPL, que no requiere de administración, nativa de java.NET, y trabaja directamente con objetos. Sus siglas corresponden con la expresión base de datos para cuatro objetos, que a su vez es el nombre de la compañía que la desarrolla, pudiendo ejecutarse sobre las plataformas de java por ser un software libre. (10)

Entre sus características principales se encuentra su alto rendimiento, doble licencia: GPL (Open Source) y Comercial (que incluye soporte), por su bajo consumo de recursos, (de 600Kb a 800Kb) es especialmente apta para dispositivos móviles y entornos Clientes/Servidor, aunque no necesariamente limitada sólo a ellos, presenta un alto nivel de respuesta y participación, su documentación es clara, amplia, ordenada y orientada a ejemplos y de fácil lectura. Presenta dos modos de trabajo embebido y Cliente/Servidor. Presenta portabilidad entre .Net y Java, y transacciones ACID. Es fácil de instalar y tiene

un único fichero de base de datos. Los objetos los almacenan tal y como son, y no hay que cambiar las clases para poder almacenarlas. (10)

1.1.5 Berkeley DB

Berkeley DB es un motor de bases de datos transaccional y escalable que puede utilizarse en cualquier aplicación, el cual provee a los desarrolladores una base de datos simple, rápida y segura, con cero administración, debido a que funciona como una biblioteca que se enlaza directamente en la aplicación eliminando la penalización en el rendimiento de los sistemas cliente-servidor y el procesamiento SQL, ideal para consultas estáticas sobre datos dinámicos. Es uno de los motores de almacenamiento soportado por MySQL, Subversión, OpenLDAP, Bogofilter y varios CMS (Content Management System). Estaba disponible con código fuente y licencia de libre distribución (free software), sin embargo fue comprado por Oracle, por lo que mantiene una licencia dual de software libre y de software privativo para implementaciones cerradas sobre Berkeley DB. (11)

Es una base de datos incrustada con API para C, C++, Java, Perl, Python, Ruby, Tcl y muchos otros lenguajes de código abierto, que permite a los desarrolladores incorporar en sus aplicaciones un motor de base de datos transaccional, rápido y escalable, con disponibilidad y confiabilidad de clase industrial.
Permite, que los clientes y usuarios finales atengan una aplicación que simplemente funcione y administre confiablemente los datos.
Berkeley DB permite miles de hilos de control manipulando bases de datos de hasta 256 terabytes en muchos sistemas, incluidos la mayoría del tipo-UNIX y Windows, e incluso sistema operativos de tiempo real. (11)

Es una base de datos de un rendimiento extraordinario, elimina los gastos de la comunicación interprocesos y SQL, se integra por completo en la aplicación y es invisible para los usuarios finales.Su recuperación de datos es en forma secuencial e indexada. Presenta procesos múltiples por aplicación e hilos múltiples por proceso. Contiene un Cifrado de datos por el algoritmo AES (Advanced Encryption Standard).Presenta registros de hasta 4GB y tablas de hasta 256TB.Su administración es automática y su código fuente es incluido. Disponible en los sistemas operativos: Linux, Windows, BSD Unix, Solaris, Mac OS X y tiene un soporte de los lenguajes C, C++, Java, Perl, Python, PHP, TCL y Ruby. (11)

1.1.6 Metakit

Es una base de datos empotrable multiplataforma y Open Source que se distribuye bajo licencia X/MIT. Es destacable la documentación que nos encontramos, aunque falta alguna documentación relativa a los puntos más oscuros de la API del lenguaje. Destaca por su pequeño tamaño y su gran funcionalidad, entre las bases de datos relacionales y las bases de datos orientadas a objeto.

Es una base de datos muy ligera, presenta pequeño tamaño , una gran funcionalidad, es muy portable con una API que enlaza con los lenguajes de programación más habituales C++, TCL y como no Python. Los archivos de datos son portátiles. La biblioteca se ha utilizado en Unix, Windows, Macintosh, VMS (Virtual Memory System), y otros, con un rango de 16 a las arquitecturas de 64-bit, de PDA's (Personal Digital Assistant) a la S390. Metakit está en uso en varios proyectos comerciales y productos a millones de computadoras de escritorio. (12)

1.1.7. Gadfly

Es un sistema pequeño de base de datos portátil escrito completamente en Python y específicamente destinado a programadores Python. Fuente de libre disposición del código, el cual permite a los programas de Python almacenar, recuperar y consultar datos en tablas sin tener que depender de cualquier motor de base de datos externa. Gadfly proporciona un modo simple, fácil y relativamente eficiente en la memoria del motor relacional de bases de datos. El aspecto atractivo de Gadfly es que funciona siempre que sea programada en Python. Las cuales puede ejecutar y apoyar el modelo cliente / servidor en cualquier plataforma que soporta la interfaz de Python; es capaz de mover los formatos de archivo utilizado por Gadfly para el almacenamiento a Win95 ó a Linux a través de un mecanismo de copia binaria y ejecutar la base de datos, debido a que no presenta en este momento un control de concurrencia. (13)

Entre sus características más importantes se encuentra su portabilidad y compatibilidad con las bases de datos persistentes. (13)

1.1.8 Daffodil DB

Es una plataforma independiente con normas ricas en funciones y cero de administración. Compactible con la mayoría de las principales marcas de Servidores de aplicaciones y servidores red. Es una base de datos integrada la cual soporta muchas conexiones concurrentes, la cual ha sido diseñada

específicamente para incrustar dentro de la aplicación java. Es fácil de instalar y desplegar, completamente segura, compatible con varios servidores de aplicaciones. (14)

1.1.9 Perst

Base de datos de código abierto, la cual presenta una licencia dual. Está disponible en una edición como una base de datos de Java embebida y otro implementado en C#. Ofrece a los desarrolladores la capacidad de clasificar, almacenar y recuperar los objetos en sus aplicaciones con la máxima velocidad y con poca memoria y gastos generales de almacenamiento, mientras aprovecha el paradigma orientado a objetos de Java y C #.Tiene una merecida reputación por su simplicidad diseño y alto rendimiento. Perst ofrece multiproceso de acceso, encriptación de datos y la replicación asincrónica. (15)

Entre sus características fundamentales se encuentra su rápida recuperación, que no tiene ningún archivo de registro, guarda grandes volúmenes de datos (tamaño máximo de la base de datos 1 terabyte), tiene un tamaño reducido, su tamaño de biblioteca de 250 KB.es orientada a objetos y quita los campos automáticamente. (15)

1.1.10 Metanotion BlockFile

Es una base de datos 100% pura de Java, ligera e integrable. Es un cruce entre SQLite y Berkeley. Desarrollada para el uso de aplicaciones móviles. Es una base de datos agradable para trabajar, por ser rápida con distribución gratuita. (16)

Sus principales características son que trabaja con único archivo, Se puede utilizar de forma gratuita en un proyecto comercial, se puede utilizar de forma gratuita en un proyecto comercial. No hay dependencia de archivo API. Es muy rápida y su tiempo de carga es un poco lento. (16)

1.1.11 Quadcap Embeddable Database (QED)

Es una base de datos relacional pura de java muy rápida. QEB es una aplicación de la norma SQL 92, con las Transacciones y la Recuperación de errores resistente. Libre y en proyectos de código abierto contiene una licencia de código abierto que permite su uso libre. Presenta un alto rendimiento con un tamaño reducido y cero de administración. Contiene integridad de datos y presenta soporte para JDBC. (17)

1.2 Selección de las bases de datos adecuadas

Se seleccionan las bases de datos H2, Apache Derby y SQLite por ser las bases de datos más utilizadas en el mundo; ya que presentan un grupo de características que las hacen ideales para ser incorporadas en una aplicación java. Son fáciles de instalar, desplegar y usar; permitiendo soportar datos de almacenamiento y de interrogación SQL, proporcionando así todas las funciones que se pueden encontrar normalmente en los sistemas grandes de base de datos, basadas en JDBC muy livianas de código abierto y presentando una gran portabilidad.

1.3 Pruebas de base de datos

Dentro del estudio para la selección de la base de datos las pruebas son fundamentales; ya que a partir de ellas es posible controlar el rendimiento.

Ellas constituyen una actividad en la cual las bases de datos son ejecutadas bajo un grupo de condiciones específicas. Las pruebas se pueden traducir como una revisión de las bases de datos con el objetivo de encontrar problemas antes que éstos sean detectados por el cliente final garantizando la calidad del sistema. (18)

Las pruebas de bases de datos son conocidas como la verificación y validación (V&V) de los resultados guardados. Resumiendo se puede decir que las pruebas de base de datos demuestran hasta qué punto estas pueden funcionar de acuerdo con las especificaciones y requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba, proporcionan una buena indicación de la fiabilidad de las bases de datos y, de alguna manera, indican la calidad como un todo. (18)

Dentro de los objetivos fundamentales que se persiguen al aplicarles las pruebas a las bases de datos se encuentran los siguientes:

- ✓ Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- ✓ Detectar fallas o errores.
- ✓ Aumentar la calidad de las bases de datos.

Existe una serie de actividades que se deben realizar para conseguir la ejecución de las pruebas, estas son:

Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar las bases de datos. Distintas técnicas de prueba ejecutan diferentes criterios como guía para realizar las pruebas.

Generación de los casos de prueba: Consiste en la confección de los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir las bases de datos al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

Definición de los procedimientos de la prueba: Conlleva a una especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar y cuándo.

Ejecución de la prueba: Es el momento de aplicar los casos de prueba generados previamente e identificar los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

1.3.1 Estrategias de prueba

Una estrategia de prueba integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por lo tanto, cualquier estrategia de pruebas debe incorporar la planificación de la prueba, el diseño de los casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes. Para aplicar pruebas al software se deben seguir un conjunto de estrategias para lograr que estas se hagan en el menor tiempo posible y con la calidad requerida, además de lograr que arrojen los resultados esperados.

1.3.2 Tipo de pruebas

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Cada una de las pruebas se realiza en determinados momentos del ciclo de vida. Teniendo en cuenta esto, las pruebas se agrupan por niveles, de acuerdo con las diferentes etapas del proceso de desarrollo:

Pruebas Funcionales

Entre todos los tipos de pruebas que se realizan en un sistema está el tipo que evalúa la funcionalidad de esté, estas son las llamadas pruebas funcionales ya que se centran en las funciones, las entradas y las salidas. Su objetivo fundamental es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados, estas pruebas tienen como meta:

- ✓ Verificar el procesamiento, recuperación e implementación adecuada.

- ✓ Verificar la apropiada aceptación de datos.

Pruebas de rendimiento

Las pruebas de rendimiento pueden servir para diferentes propósitos. Estas son capaces de demostrar que un sistema cumple los criterios de rendimiento. También se utilizan para comparar dos sistemas y encontrar cuál de ellos funciona mejor. O pueden medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico, los ingenieros de software utilizan herramientas como pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles (y umbrales) del mismo que mantengan un tiempo de respuesta aceptable. Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplíen durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el coste de la solución. Esto es cierto en el caso de las pruebas funcionales, pero mucho más en las pruebas de rendimiento, debido a que su ámbito de aplicación es de principio a fin. En las pruebas de rendimiento, a menudo es crucial (y con frecuencia difícil de conseguir) que las condiciones de prueba sean similares a las esperadas en el uso real. Esto es, sin embargo, casi imposible en la práctica. La razón es que los sistemas en producción tienen un carácter aleatorio de la carga de trabajo y aunque en las pruebas se intente dar lo mejor de sí para imitar el volumen de trabajo que pueda tener el entorno de producción, es imposible reproducir exactamente la variabilidad de ese trabajo, salvo en el sistema más simple. Dentro de los elementos que son comprobados, con las pruebas de rendimiento, se pueden encontrar:

- ✓ La determinación de los tiempos de respuesta.
- ✓ La verificación del espacio que ocupa el módulo en disco o memoria. (19)

Dentro de las pruebas de rendimiento se pueden encontrar otras pruebas más específicas

Prueba de carga

Encargada de probar el funcionamiento del software bajo condiciones extremas. Estudia la especificación de las bases de datos, las funciones que debe realizar, las entradas y las salidas analizando los valores límites.

El objetivo primordial de las pruebas de carga es validar que el comportamiento y los tiempos de respuesta experimentados, bajo ciertas condiciones (Número de usuarios, número de transacciones por unidad de tiempo) satisfacen adecuadamente los requerimientos especificados. (19)

Prueba de resistencia (Estrés)

Estas pruebas están diseñadas para enfrentar al programa a condiciones anormales. La prueba ejecuta un sistema de manera que demande recursos en cantidad, frecuencia o volúmenes anormales muy superior al esperado en su explotación real. (19)

Pueden encontrar errores como:

- ✓ Hardware Insuficiente (espacio en disco, consumo de RAM (Random Access Memory).
- ✓ Necesidad de ajuste del servidor de base de datos.
- ✓ Respuesta lenta.

Prueba de Volumen

Pruebas que se hacen para analizar el comportamiento de la base de datos, con volúmenes de datos almacenados similares a los esperados en la explotación real del sistema, poblándose cada tabla con cantidades de registros de entre un 110% y un 150% de los volúmenes esperados. Estas pruebas pueden detectar errores tanto del diseño como de la implementación del sistema como tal; errores tales como: falta de espacio en disco, falta de optimización de las consultas, mal diseño de las mismas, entre otros.

Se realizan para encontrar debilidades en el sistema al momento de manejar grandes volúmenes de datos durante prolongados períodos de tiempo, el objetivo principal es determinar si la plataforma de integración se degrada o deja de funcionar al manejar grandes volúmenes de datos. (20)

Conclusiones

En este capítulo se realizó un estudio de las bases de datos empotrables en el mundo, así como una breve caracterización de cada una de ellas profundizando en el tema. Además, se definieron los diferentes

tipos de pruebas para desarrollar en un buen procedimiento de pruebas. Se seleccionó como las bases de datos más apropiadas para una posible incorporación a la plataforma alasGRATO las bases de datos H2, Apache Derby y SQLite.

CAPÍTULO 2: PROGRAMAS Y PROCEDIMIENTO

Introducción

En este capítulo se realizará un estudio de los programas que se utilizaran en el desarrollo de las pruebas; definiéndose así el procedimiento que se empleará para la realización de las pruebas de rendimiento y de volumen a las bases de datos H2, Apache Derby y SQLite.

2.1 Programas

Existen unas series de herramientas muy útiles para permitir a los sistemas asegurar alta fiabilidad y rendimiento de manera sencilla, con productos y servicios que ayudan a los desarrolladores a emplear las mejores prácticas en el proceso de producir aplicaciones de alta calidad. Con estas herramientas se proporciona un análisis detallado de la calidad y rendimiento de las aplicaciones propiciando a los desarrolladores un experto automatizado capaz de advertirles, localizar y corregir sus problemas.

2.1.2 JMeter

Existen varias formas de generar la carga de trabajo para la realización de las pruebas de rendimiento. La más sencilla es de forma manual, sin embargo, es la menos eficiente. Otra vía es desarrollando un software que permita la realización de estas pruebas, que mirándolo desde este punto de vista representa un ahorro considerable de dinero, sin embargo estas herramientas son enormemente complejas en su desarrollo, lo que representaría un buen gasto de dinero y tiempo, a no ser que se limitaran los requisitos.

Por lo antes expuesto que se toma la decisión de usar algún software existente, que permita configurar un entorno de pruebas propio. A la hora de escoger la herramienta uno de los aspectos más importantes a tener en cuenta es su flexibilidad respecto a la configuración y la confianza en los resultados que este exponga, pues de esta forma se garantiza un rápido aprendizaje en su uso y el éxito en la aplicación de las pruebas.

El JMeter es la herramienta seleccionada para la realización de pruebas pues es una herramienta libre muy confiable que muestra los resultados de las pruebas en una amplia variedad de informes y gráficas, con una gran cantidad de variables diferentes que permiten interpretar los resultados desde diferentes puntos de vista. Además, facilita la rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo. Esta herramienta brinda gran cantidad de variantes para la obtención de

resultados, como pueden ser el reporte agregado que muestra una tabla con todos los valores que va dando como respuesta la aplicación por peticiones permitiendo un análisis mucho más profundo de los resultados.

Es un proyecto de Apache Jakarta que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, en base de datos.
Puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP (File Transfer Protocol), LDAP (Lightweight Directory Access Protocol), Servicios web, JMS (Java Message Service), HTTP (Hypertext Transfer Protocol)y conexiones TCP (Transmission Control Protocol) genéricas. Una característica importante del JMeter es que es extensible y ofrece la posibilidad de que el propio usuario desarrolle en Java un controlador ("Controller") a medida, cumpliendo una interfaz Java y depositando el .jar correspondiente al desarrollo en el directorio "lib" de JMeter.
Además permite realizar pruebas distribuidas en distintos ordenadores que actúan como clientes. Puede simular una gran carga en el servidor, HTTP y FTP testing y bases de datos mediante JDBC, multitarea y con grandes facilidades para extender su funcionalidad mediante plugings.

Los elementos jerárquicos son los elementos de escucha, los elementos de configuración y los cronómetros, y los elementos ordenados serían los controladores y los agente de pruebas. (21)

2.1.3 Entornos de desarrollo: Eclipse

Eclipse es uno de los más potentes entornos integrados de desarrollo (IDE) que permite la construcción de aplicaciones en Java. Admite la incorporación de otros plug-ins para obtener un mayor número de funcionalidades, es una herramienta de código abierto, multiplataforma y compatible con los sistemas operativos más utilizados en el mundo. Además, posee la capacidad de ser soportado para distintas arquitecturas, control de versiones con cvs o con subversión, resaltado de sintaxis, un potente refactorizado de código, asistentes (wizards): para la creación, exportación e importación de proyectos. (22)

2.1.4 Lenguaje de programación: Java

El lenguaje de programación Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Su diseño fue concebido para que los programadores puedan lograr fluidez con el lenguaje. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++

son lenguajes más difundidos, por ello, Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje, además, el mismo elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica o recolector de basura). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread (hilo) de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de esta.

Una de las características más importantes de Java es que posee una arquitectura neutral, es decir el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar de ningún modo la máquina en que ha sido generado.

Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando para la portabilidad a otras plataformas.

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Java, además, construye sus interfaces de usuario a través de un sistema abstracto de ventanas, de forma que ellas puedan ser implantadas en entornos Unix, Pc o Mac. (23)

2.1.5 Generadores de datos

Para realizar las pruebas es necesario generar los datos a cada base de datos, por lo que se necesitará un generador de datos. Estos generadores nos brindan un script en desarrollado en un lenguaje de programación que nos permite generar grandes volúmenes de datos personalizados en una variedad de formatos. En el presente caso se realizó una pequeña aplicación que generará el número de datos necesarios para cada gestor a estudiar.

2.2 Procedimiento para el diseño de los casos de prueba

Se necesita un diseño de casos de pruebas que incluya la mayor cantidad de aspectos relacionados con la preparación que lleva la creación de un correcto procedimiento de pruebas. Esto permitirá el éxito del trabajo realizado. Es por esto los aspectos a tener en cuenta en el diseño de caso de pruebas son la

elección de las variables adecuadas, el diseño del entorno de pruebas, la configuración de las pruebas en la herramienta y el modelo de la planificación

2.2.1 Elección de las variables adecuadas:

Variables de prueba de rendimiento

De acuerdo con los requerimientos establecidos, las variables de entrada que se van a utilizar son 50 usuarios concurrentes para cada una de las bases de datos, pues está establecido que debido a que el trabajo está diseñado para mejorar la gestión de la información en la plataforma alasGRATO el número de usuario conectados concurrentemente no deben ser más de tres, el sistema debe permitir sin problemas la conexión de 10 personas simultáneamente. Para las pruebas de Estrés se utilizará el valor definido para las pruebas de carga con el objetivo de comprobar si el sistema está apto para recibir esa carga de trabajo bajo condiciones de estrés y a su vez se establecerán casos de prueba para las funcionalidades buscando el valor de conexión que provoca que la aplicación deje de responder. Para ambos casos las variables de salida serán analizadas verificando los valores obtenidos como respuesta del sistema, comprobándolos con una variación de cantidad en las conexiones de usuarios. Esto se refiere a que en las pruebas de Carga la variable que tiene como valor 50, tomará rimero valor 25 para comprobar el rendimiento con un número de peticiones media, logrando de esta forma valores para comparar a la hora de dar resultados. En el caso de la variable que tiene por valor 25 será doblada con el mismo objetivo.

Variables de entrada:

Número de usuarios.

Es importante mencionar que a veces es un poco difícil definir el número de usuarios que se usarán para la prueba, hay demasiados parámetros involucrados a la hora de darle carga a un sistema. Es por esto que la guía será el establecimiento de una carga que simule el uso normal del sistema. Variándola en dependencia del tipo de prueba de rendimiento, verificando siempre los valores extremos de usuarios que debe soportar el sistema.

Variables a generar:

A continuación se muestran las variables que serán usadas para la interpretación de los resultados:

- ✓ Niveles de rendimiento.
- ✓ Número de muestras.

- ✓ Tiempo de respuesta.
- ✓ Por ciento (%) de errores.

Variables de prueba de Volumen

En las pruebas de volumen deben ser determinadas cuáles son las variables de entrada y salida con las que se va a trabajar, se tienen en cuenta los valores que tomarán las variables que necesita la base de datos con el cual se realizarán las pruebas. De esta forma, se garantiza una correcta interpretación de los resultados. Es por esto que tienen que ser definidas antes de la realización de las pruebas. Para este procedimiento las variables adecuadas son:

Variables de entrada:

Cantidad de datos

Variables a generar:

A continuación se muestran las variables que serán usadas para la interpretación de los resultados:

- ✓ Errores al insertar grandes volúmenes de datos.
- ✓ Tiempo en que transcurrió introducir los datos.

2.2.2 Diseño del entorno de pruebas

Para la aplicación del procedimiento de pruebas de rendimiento diseñado al software se identifica un entorno con las siguientes características: Tres bases de datos locales y como sistema operativo Windows Servi Pack 2.

Especificaciones de Hardware:

Hardware	**Datos**
Microprocesador	Intel(R) Core(TM)2 Duo CPU E4500 a 2.20 GHz
Memoria RAM	1 GB
Disco Duro	160 GB

Software	**Datos**
Sistema Operativo	Windows XP Servi Pack 2
Bases de Datos	Apache Derby,H2 y SQLite
Antivirus	Kaspersky Workstation 6.0
Otras herramientas	Microsoft Office 2007
Máquina Virtual	Java Virtual Machine 1.3 o superior
Software de prueba	JMeter Versión 2.3.1

2.2.3 Diseño de las pruebas en la herramienta

Teniendo en cuenta la especificación de las variables independientes y las que se van a generar se procede al diseño de las pruebas en la herramienta especificada, este diseño consta de dos elementos fundamentales, la grabación de los escenarios de prueba y la confección de los planes de prueba.

La grabación de los escenarios de prueba se encuentran dirigidas a recoger todas las peticiones que se generan al realizar acciones entre los clientes, mientras que la confección de los planes de prueba para las mediciones de carga y estrés se realiza utilizando el método de pruebas complejas que consiste en la simulación de usuarios concurrentes accediendo a la información de la base de datos, debido a que hay que entrar datos al sistema y enviarlos a la base de datos del mismo. Esto último se realiza para lograr una mayor similitud con los escenarios reales.

2.2.4 Plantilla para el diseño de casos de pruebas

A la hora de confeccionar los casos de prueba para las pruebas se hace referencia a las pruebas de rendimiento donde se puede apreciar los dos tipos de criterios que interesan en este caso, estos son las pruebas de estrés y las pruebas de carga. Para este tipo de pruebas se traza una estrategia en la cual a las bases de datos se le conectan una cantidad de usuarios concurrentes accediendo a la información.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	Cantidad de usuarios accediendo a la información de la base de datos.
Tiempo entre conexión y conexión: [Período de subida (en segundos)]	[Tiempo que ocurrirá entre la conexión de los usuarios de la primera iteración y todas las demás]
Número de iteraciones: [Contador del bucle]	[Cantidad de veces que se repetirá la simulación de conexión de usuarios]

En la tabla se aprecian tres variables las cuales son pedidas por el software de pruebas JMeter y definidas por el usuario que realizará las pruebas:

Número de usuarios concurrentes (Número de Hilos): Variable que establece la cantidad de usuarios conectados simultáneamente que simulará el programa para el tipo de prueba establecida. Es importante saber qué **Número de Hilos** es el nombre que trae definido el software para esta variable.

Tiempo entre conexión y conexión (Período de subida (en segundos)): Variable que representa el tiempo que ocurrirá entre las conexiones de los usuarios definidos en Número de Hilos. Para el caso de la prueba de carga esta variable tendrá por valor 0, pues las conexiones ocurrirán simultáneamente sin tiempo entre un grupo de conexión y otro. En el caso de la prueba de estrés el valor de esta variable será definido por el probador, para especificar el tiempo de espera.

Número de iteraciones (Contador del Bucle): Esta variable es la encargada de recoger la cantidad de veces que el usuario desea que se conecte el grupo de hilos definido. Para el caso de las pruebas de carga, esta variable toma por defecto el valor 1, pues el grupo de usuarios (Hilos) se conectará una sola vez. En el caso de las pruebas de estrés esta variable tomará el valor definido por el probador, pues éste es el encargado de definir la cantidad de veces que el grupo de hilos se conectará, llegando en cierta cantidad de tiempo al número de usuarios definido en la prueba.

Resultados Generales para las pruebas de carga y estrés:

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg	% CPU

Resultados Generales para las pruebas de volumen:

Datos /Tablas	Datos Insertados	Tiempo	Errores

La tabla de **Resultados Generales** es la encargada de recoger el resumen de los valores de las pruebas efectuadas. A través del análisis y de la comparación de los mismos se llegará a los resultados y se darán las conclusiones. Los resultados finales se obtendrán de la comparación y análisis de los valores cantidad de muestras (**Muestras**), el por ciento de error (**% Error**) y el valor del rendimiento (**Rendimiento**) medido por el número de peticiones por segundo, pues estas variables resumen el contenido que es necesario verificar dentro de la aplicación para comprobar su rendimiento. (Para más información sobre las variables de la tabla dirigirse al Anexo 1).

2.2.5 Casos de pruebas.

Caso de prueba 1:

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Carga

Base de datos a probar:

La base de datos (es)	Comentario
H2	La base de datos contiene una gran cantidad de información generada aleatoriamente por el generador de datos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**25**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**
Número de iteraciones: [Contador del bucle]	**1**

Variables	Valores

Número de usuarios concurrentes: [Número de hilos]	**50**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**
Número de iteraciones: [Contador del bucle]	**1**

Caso de prueba 2:

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Carga

Base de datos a probar:

La base de datos (es)	Comentario
SQLite	La base de datos contiene una gran cantidad de información generada aleatoriamente por el generador de datos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**25**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**
Número de iteraciones: [Contador del bucle]	**1**

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**50**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**

Número de iteraciones: [Contador del bucle]	**1**

Caso de prueba 3:

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Carga

Base de datos a probar:

La base de datos (es)	Comentario
Apache Derby	La base de datos contiene una gran cantidad de información generada aleatoriamente por el generador de datos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**25**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**
Número de iteraciones: [Contador del bucle]	**1**

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**50**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**0**
Número de iteraciones: [Contador del bucle]	**1**

Caso de prueba 4

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Stress

Base de datos a probar:

La base de datos (es)	**Comentario**
H2	La base de datos contiene una gran cantidad de información generada aleatoriamente por el generador de datos.

Variables	**Valores**
Número de usuarios concurrentes: [Número de hilos]	**10**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**5**
Número de iteraciones: [Contador del bucle]	**26**

Variables	**Valores**
Número de usuarios concurrentes: [Número de hilos]	**130**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**2**
Número de iteraciones: [Contador del bucle]	**2**

Caso de prueba 5

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Stress

Base de datos a probar:

La base de datos (es)	Comentario
SQLite	La base de datos contiene una gran cantidad de información generada aleatoriamente por el generador de datos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**10**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**5**
Número de iteraciones: [Contador del bucle]	**26**

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**130**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**2**
Número de iteraciones: [Contador del bucle]	**2**

Caso de prueba 6

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Stress

Base de datos a probar:

La base de datos (es)	Comentario
Apache Derby	La base de datos contiene una gran cantidad de información generada aleatoriamente por el

	generador de datos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**10**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**5**
Número de iteraciones: [Contador del bucle]	**26**

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	**130**
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	**2**
Número de iteraciones: [Contador del bucle]	**2**

Caso de prueba 7

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Volumen

Base de datos a probar:

La base de datos (es)	Comentario
H2	La base de datos se encuentra totalmente vacía.

Variables	Valores
Cantidad de datos	60000

Variables	Valores
Cantidad de datos	600000

Variables	Valores
Cantidad de datos	6000000

Caso de prueba 8

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Volumen

Base de datos a probar:

Base de datos	Comentario
SQLite	La base de datos se encuentra totalmente vacía

Variables	Valores
Cantidad de datos	60 000

Variables	Valores
Cantidad de datos	600 000

Variables	Valores
Cantidad de datos	6 000 000

Caso de prueba 9

Nombre del Proyecto: alasGRATO

Tipo de prueba: Prueba de Volumen

Base de datos a probar:

Base de datos	Comentario
Apache Derby	La base de datos se encuentra totalmente vacía

Variables	Valores
Cantidad de datos	60 000

Variables	Valores
Cantidad de datos	600 000

Variables	Valores
Cantidad de datos	6 000 000

2.3 Validación de los resultados

Para realizar la validación de los resultados de las pruebas se verifica la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, elementos que pueden provocar cierto retraso en la generación de la carga en las pruebas e interferencia en la obtención de los resultados, dichos parámetros a tener en cuenta son:

Utilización del CPU: Una utilización por debajo del 70% demuestra un rendimiento aceptable, sin embargo, por encima de este valor demuestra que el sistema está sobrecargado. Para la verificación de la utilización del CPU, se accede a su valor a través del administrador de tareas de Windows. **Utilización de la memoria:** Un uso de la memoria física de la máquina menor al 75 % es aceptable, sin embargo si el valor es igual a este o está por encima de él, demuestra la invalidación de los resultados de las pruebas.

Existen factores adicionales que se pueden usar para validar los resultados de las pruebas, estos pueden ser:

Porcentaje de error: En el resultado de las pruebas un elevado por ciento de errores puede representar varias cosas como por ejemplo, problemas con los drivers de comunicaciones de las PC Clientes,

problemas con las funcionalidades del sistema que invalida la posibilidad de ser sometido a pruebas de rendimiento y errores en la introducción de datos en la herramienta. Por lo general no se aceptan porcentajes de error superior al 1% de las transacciones realizadas por el sistema.

Margen de error: Es importante saber que por muy específico que sea el proceso de pruebas, los resultados nunca serán 100 % reales en comparación a cuando el sistema esté en explotación, aun así los resultados ayudarán a mejorar al correcto funcionamiento del mismo.

Conclusiones

En este capítulo se realizó la descripción de los programas que se utilizaran en el desarrollo de las pruebas; como la metodología a utilizar para la realización de pruebas. A lo largo del procedimiento se verificaron y se les dio respuesta a todos los aspectos que se definieron en un principio como objetivos para desarrollar un proceso correcto y eficaz de pruebas. Definiéndose los casos de pruebas; así como validándose los resultados para su posterior aceptación .Se espera que en los resultados de las bases de datos demuestre su fiabilidad.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

Introducción

Es última fase del estudio y el procedimiento de pruebas, en él se procede al análisis de los resultados en función de las pruebas realizadas y sus valores. Los criterios que se deben tener en cuenta para situaciones en las que se decida terminar los ciclos de pruebas están encaminados al rendimiento de las bases de datos. Finalmente, cuando todas las pruebas sean realizadas, y se pase a la última fase del procedimiento, análisis e interpretación de los resultados, estos serán documentados y realizando un análisis de los mismos para trabajar en su completa comparación de rendimiento de las diferentes bases de datos seleccionadas y se realizará la incorporación de la base de datos más eficiente a un módulo del proyecto.

3.1 Análisis e interpretación de los resultados

Todas las bases de datos seleccionadas fueron probadas, pues respecto a la conexión no surgieron problemas. Las bases de datos fueron accesibles en todo momento, garantizando la calidad de los resultados. Los resultados para los casos de prueba expuestos anteriormente se muestran a continuación:

Tipo de prueba: Prueba de Carga

Resultados generales de los casos de pruebas:

Número de usuarios: 25

Base de datos	Muestras	Media	Mediana	Min	Max	Línea 90%	% error	Rendimiento	Kb/seg	%CPU
H2	25	21	16	0	32	32	0.00%	531.9/sec	5197.6	10
SQLite	25	353	312	156	547	516	0.00%	44.5/sec	6411.3	54
Apache Derby	25	504	515	375	609	609	0.00%	41.1/sec	6394.8	41

Durante la ejecución de la prueba a las diferentes bases de datos seleccionadas, el CPU se mantuvo por debajo del 70% de utilización, donde se demostró que las tres bases de datos seleccionadas muestran un rendimiento aceptable, no sobrecargando el sistema; la memoria física se mantuvo por debajo del 75 %. Se analizó los valores de cantidad de peticiones, y rendimiento se puede concluir que las bases de datos responden satisfactoriamente. El número de peticiones esperado era 25 para 25 usuarios concurrentes. Según los resultados arrojados por las pruebas de carga se llega a la conclusión de que todas las bases de datos seleccionadas se pueden incorporar al sistema. Destacándose con las mejores características para la incorporación al sistema H2. La cual muestra un mayor rendimiento en comparación con las demás bases de datos seleccionadas para esta prueba como muestra en la figura 1.

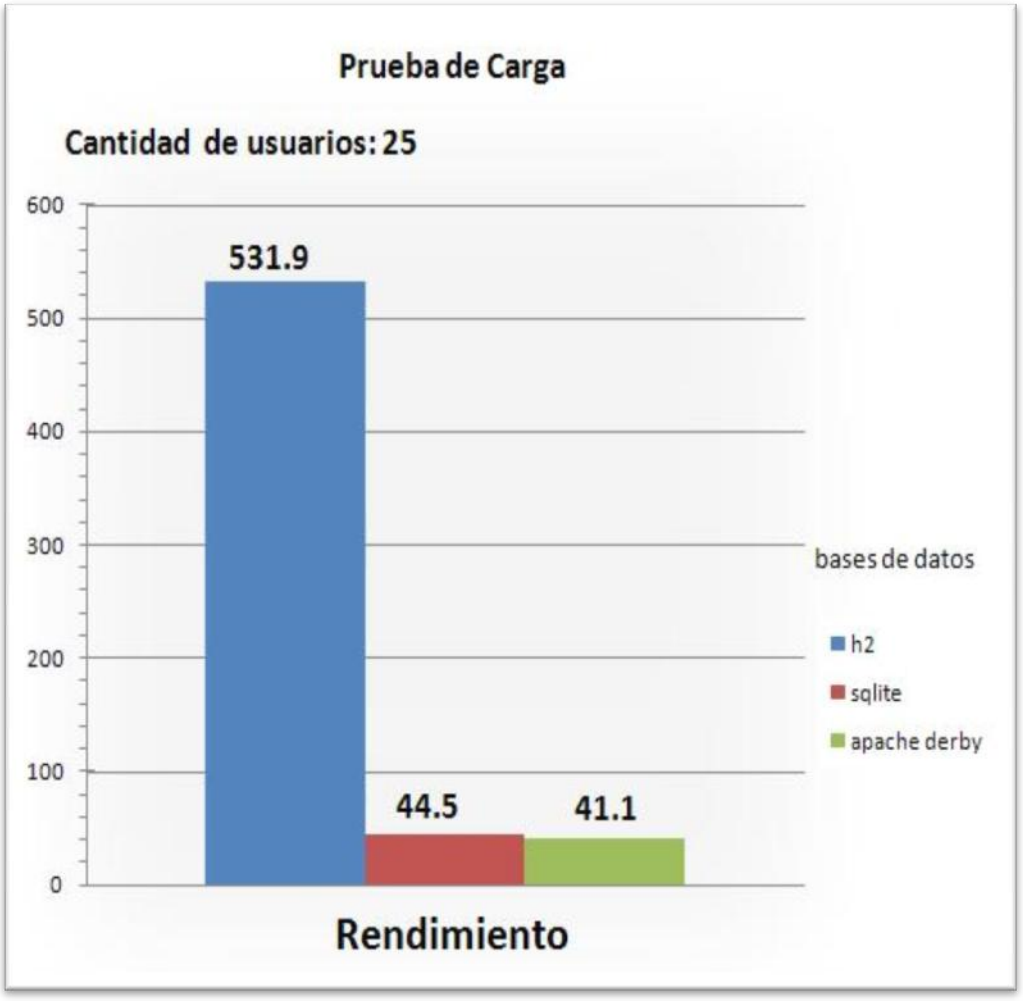


Figura 1: Prueba de carga para 25 usuarios.

Número de usuarios: 50

Base de datos	Muestras	Media	Mediana	Min	Max	Línea 90%	% error	Rendimiento	Kb/seg	%CPU
H2	50	7	0	0	32	16	0.00%	632.9/sec	6184.5	11
SQLite	50	580	594	140	984	938	0.00%	49.3/sec	7090.8	60
Apache Derby	50	130	140	15	313	250	0.00%	45.3/sec	6990.3	42

Durante la ejecución de la prueba a las diferentes bases de datos seleccionadas, el CPU se mantuvo por debajo del 70% de utilización como se muestra en la tabla, donde se demostró que las tres bases de datos seleccionadas muestran un rendimiento aceptable, no sobrecargando el sistema; la memoria física se mantuvo por debajo del 75 %. Se analizó los valores de cantidad de peticiones, y rendimiento se puede concluir que las bases de datos responden satisfactoriamente. El número de peticiones esperado era 50 para 50 usuarios concurrentes. Según los resultados arrojados por las pruebas de carga se llega a la conclusión de que todas las bases de datos seleccionadas se pueden incorporar al sistema. Destacándose con las mejores características para la incorporación al sistema H2. La cual muestra un mayor rendimiento en comparación con las demás bases de datos seleccionadas para esta prueba como muestra en la figura 2.

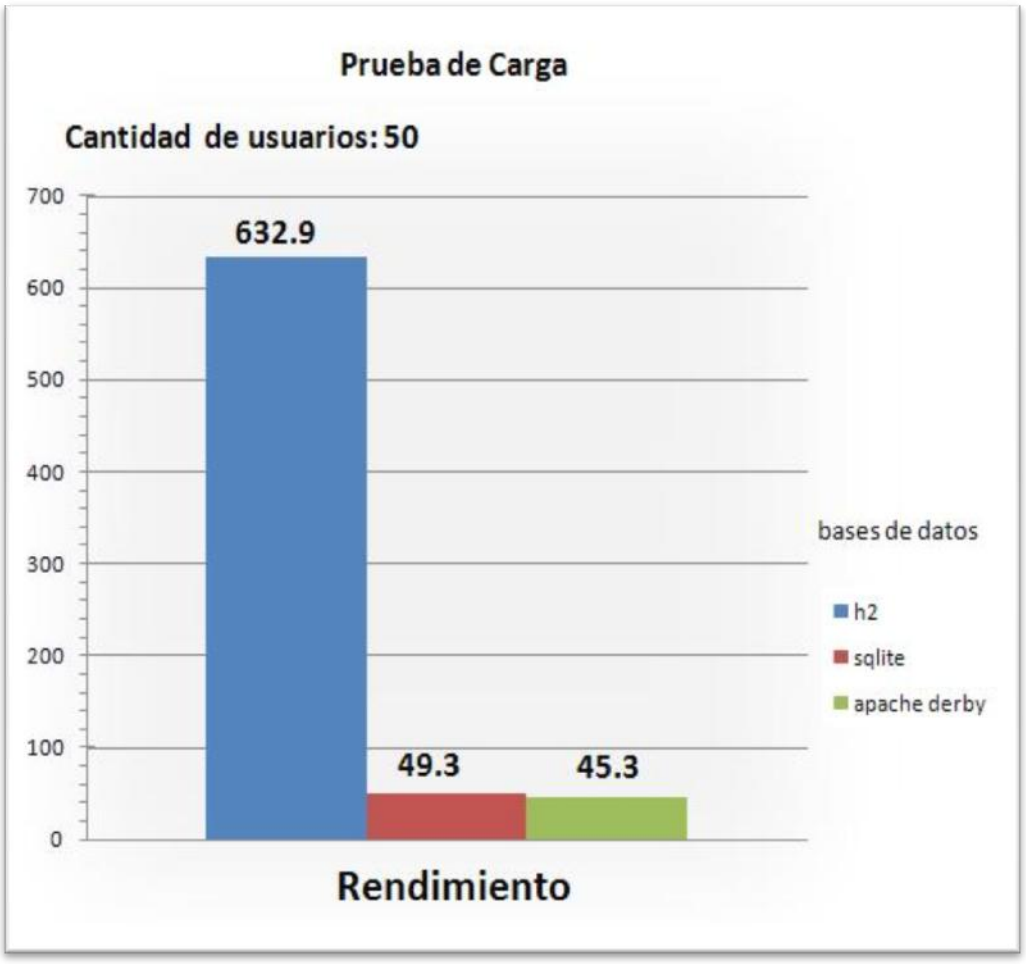


Figura 2: Prueba de carga para 50 usuarios.

Tipo de prueba: Prueba de Estrés

Resultados generales de los casos de pruebas:

Cantidad de usuarios: 10

Tiempo entre conexión y conexión: 5

Numero de iteraciones: 26

Total: 260

Base de datos	Muestras	Media	Mediana	Min	Max	Línea 90%	% error	Rendimiento	Kb/seg	%CPU
H2	260	0	0	0	16	0	0.00%	57.2/sec	558.7	14
SQLite	260	32	31	15	63	47	0.00%	48.9/sec	7053.0	90
Apache Derby	260	17	16	15	32	31	0.00%	52.0/sec	8100.4	48

Durante la ejecución de la prueba a las diferentes bases de datos seleccionadas el CPU se mantuvo por debajo de 70 % de utilización en las bases de datos H2 y Apache Derby como se muestra en la tabla, donde se demostró que estas bases de datos muestran un rendimiento aceptable, no sobrecargando el sistema, no siendo así en la base de datos SQLite la cual no cumple con los parámetros establecidos en la validación de resultados. La memoria física durante la ejecución de las pruebas se mantuvo por debajo del 75 %.

Se analizó los valores de cantidad de peticiones, y rendimiento concluye que las bases de datos responden satisfactoriamente. El número de peticiones esperado era 260 para 260 usuarios concurrentes. Según los resultados arrojados por las pruebas de estrés se llega a la conclusión de que las bases de datos H2 y Apache Derby se pueden incorporar al sistema. Destacándose con las mejores características para la incorporación al sistema H2. La cual muestra un menor uso de CPU y un mayor rendimiento en comparación con las demás bases de datos seleccionadas para esta prueba como muestra en la figura 3.

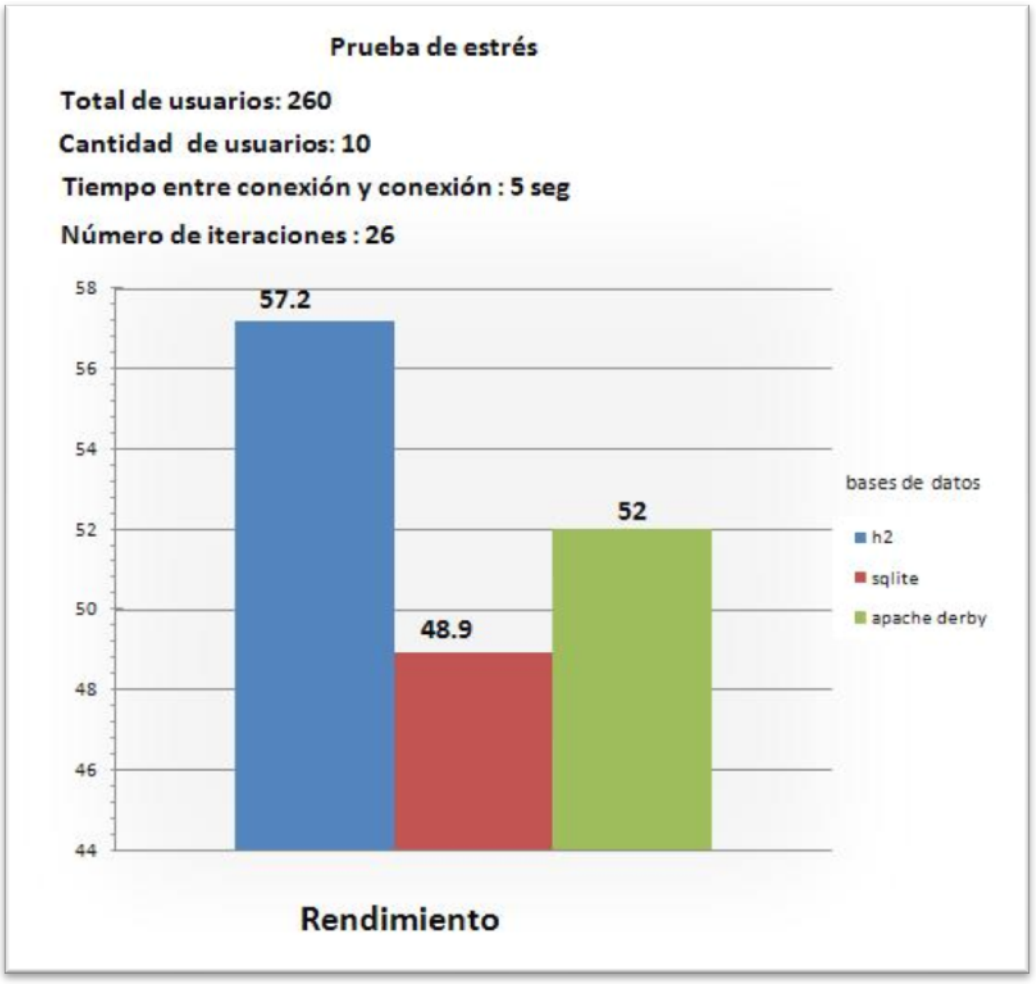


Figura 3: Prueba de estrés para 10 usuarios.

Cantidad de usuarios: 130

Tiempo entre conexión y conexión: 2

Numero de iteraciones: 2

Total: 260

Base de datos	Muestras	Media	Mediana	Min	Max	Línea 90%	% error	Rendimiento	Kb/seg	%CPU
H2	260	0	0	0	16	0	0.00%	129.0/sec	1260.8	17
SQLite	260	1112	1203	46	2813	1719	0.00%	51.7/sec	7448.3	100
Apache Derby	260	180	46	15	1485	609	0.00%	88.5/sec	13785.5	97

Durante la ejecución de la prueba a las diferentes bases de datos seleccionadas el CPU se mantuvo por debajo de 70 % de utilización en la base de datos H2 como se muestra en la tabla, donde se demostró que esta base de datos muestran un rendimiento aceptable, no sobrecargando el sistema, no siendo así en la base de datos SQLite y Apache Derby las cuales no cumplen con los parámetros establecidos en la validación de resultados. La memoria física durante la ejecución de las pruebas se mantuvo por debajo del 75 %.

Se analizó los valores de cantidad de peticiones y rendimiento, se concluye que las bases de datos responden satisfactoriamente. El número de peticiones esperado era 260 para 260 usuarios concurrentes. Según los resultados arrojados por las pruebas de estrés de que la base de datos que se pueden incorporar al sistema es H2 y destacándose con las mejores características para la incorporación. La cual muestra el menor uso de CPU y el mayor rendimiento en comparación con las demás bases de datos seleccionadas para esta prueba como muestra la figura 4.

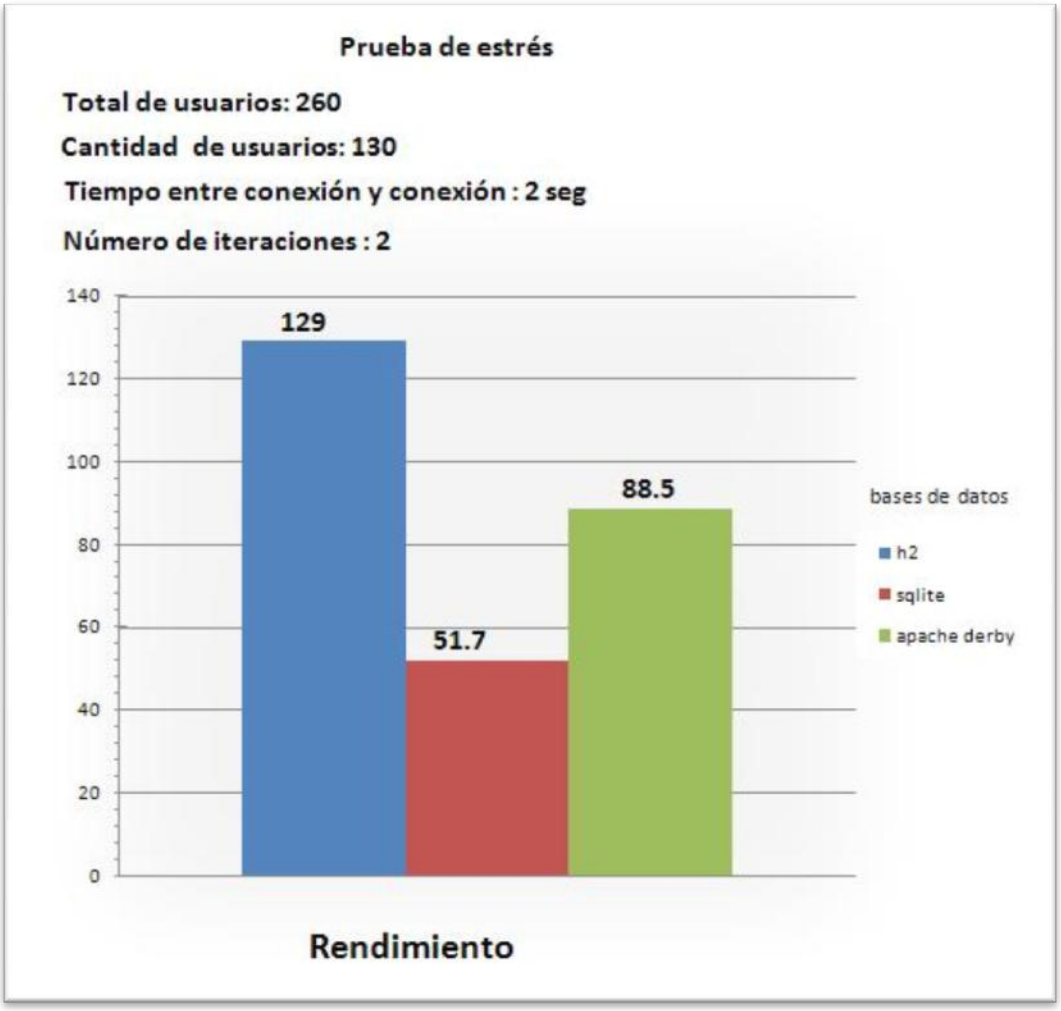


Figura 4: Prueba de estrés para 130 usuarios.

Tipo de prueba: Prueba de Volumen

Resultados generales de los casos de pruebas:

Base de Datos	Datos Insertados	Tiempo/seg	Errores
H2	**60 000**	**15**	**Ninguno**
SQLite	**60 000**	**592**	**Ninguno**
Apache Derby	**60 000**	**100**	**Ninguno**

Con el uso de una pequeña aplicación desarrollada en eclipse se generaron a las diferentes bases de datos seleccionadas, volúmenes de 60 000 valores aleatorios. Destacándose en su rapidez al insertar los datos la base de datos H2 como se muestra en la tabla y en la figura 5.

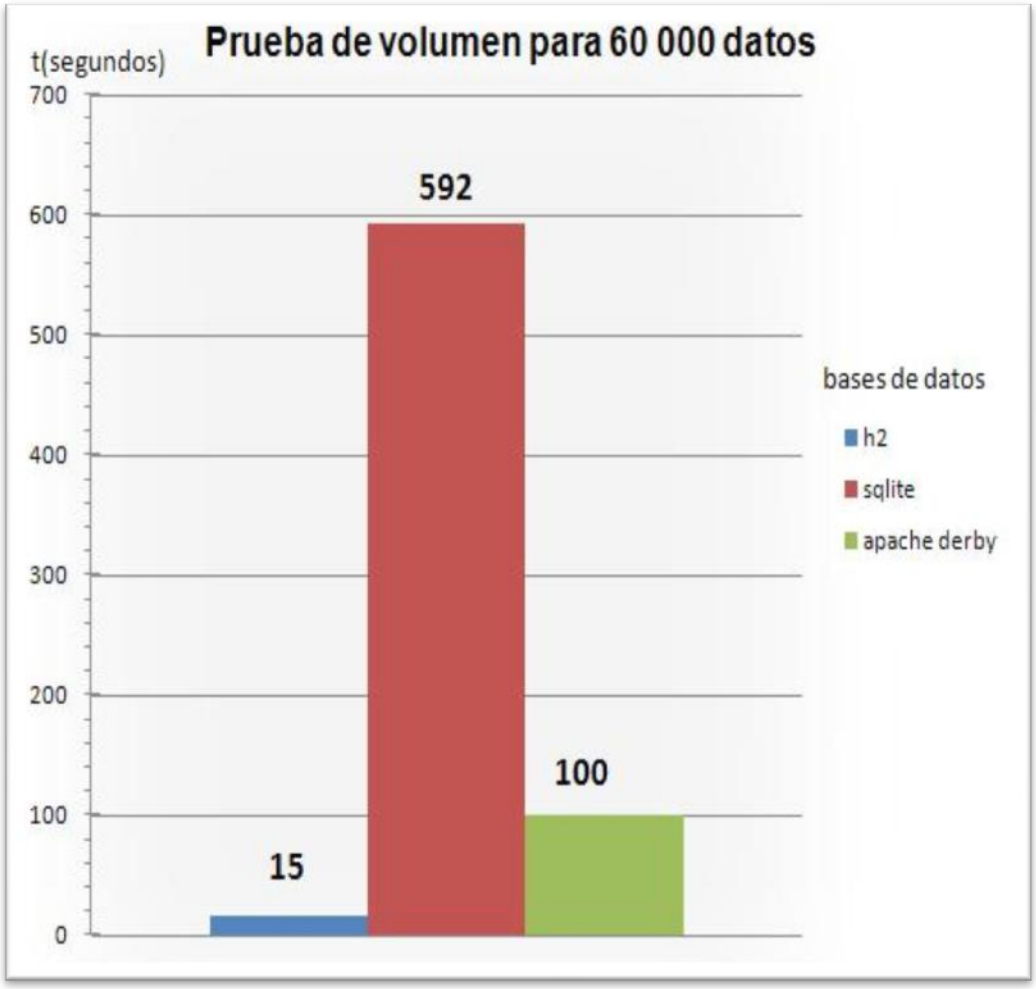


Figura 5: Prueba de volumen para 60 000.

Resultados Generales:

Base de Datos	Datos Insertados	Tiempo/seg	Errores
H2	**600 000**	**96**	**Ninguno**
SQLite	**600 000**	**5423**	**Ninguno**

Apache Derby	**600 000**	**2257**	**Ninguno**

En la segunda prueba se generaron a las diferentes bases de datos seleccionadas, volúmenes de 600 000 valores aleatorios. Destacándose en su rapidez al insertar los datos la base de datos H2 como se muestra en la tabla y en la figura 6.

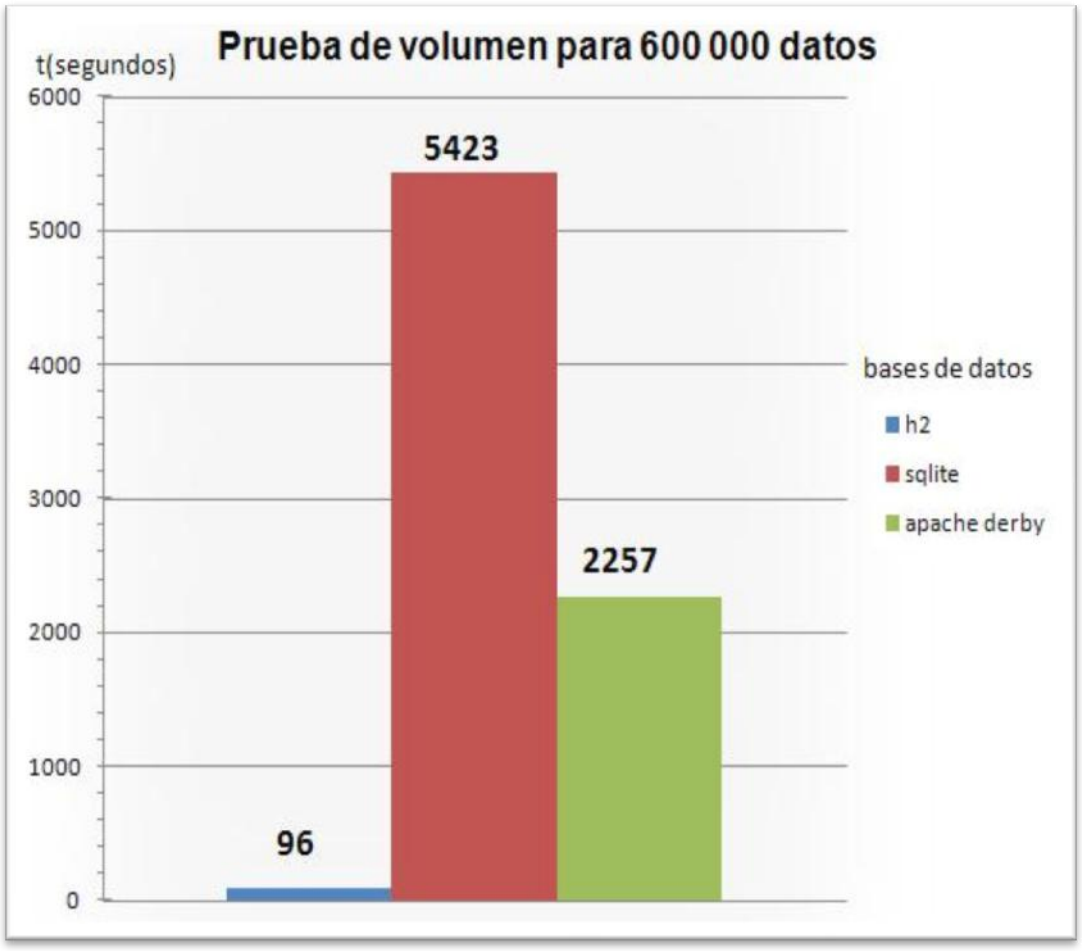


Figura 6: Prueba de volumen para 600 000.

Resultados Generales:

Base de Datos	**Datos Insertados**	**Tiempo/seg**	**Errores**

H2	**6 000 000**	**703**	**Ninguno**
SQLite	**6 000 000**	**47029**	**Ninguno**
Apache Derby	**6 000 000**	**22578**	**Ninguno**

Por último se generaron a las diferentes bases de datos seleccionadas, volúmenes de 600 000 valores aleatorios. Destacándose en su rapidez al insertar los datos la base de datos H2 como se muestra en la tabla y en la figura 7.

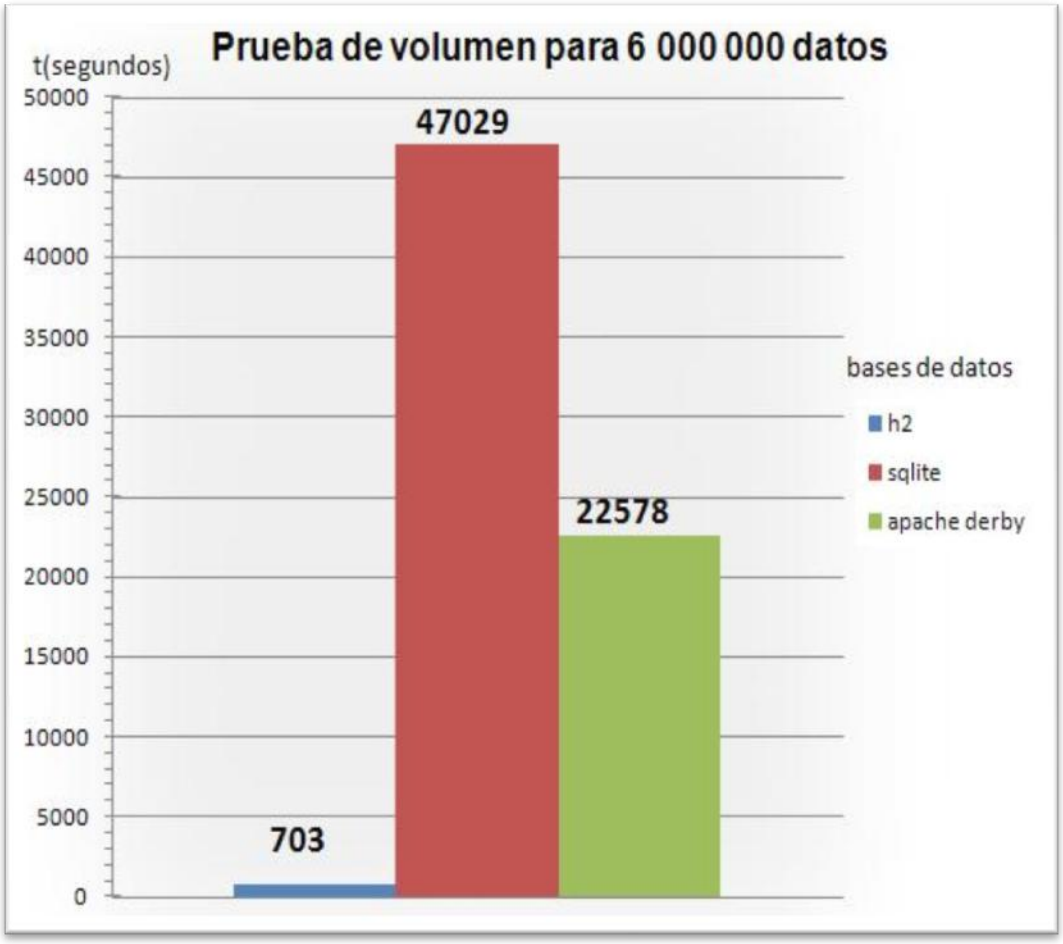


Figura 7: Prueba de volumen para 6 000 000.

La base de datos más eficiente según los resultados de las pruebas es H2, la cual presenta una gran rapidez del trabajo a la hora de insertar y extraer datos, teniendo así un mejor rendimiento en las pruebas.

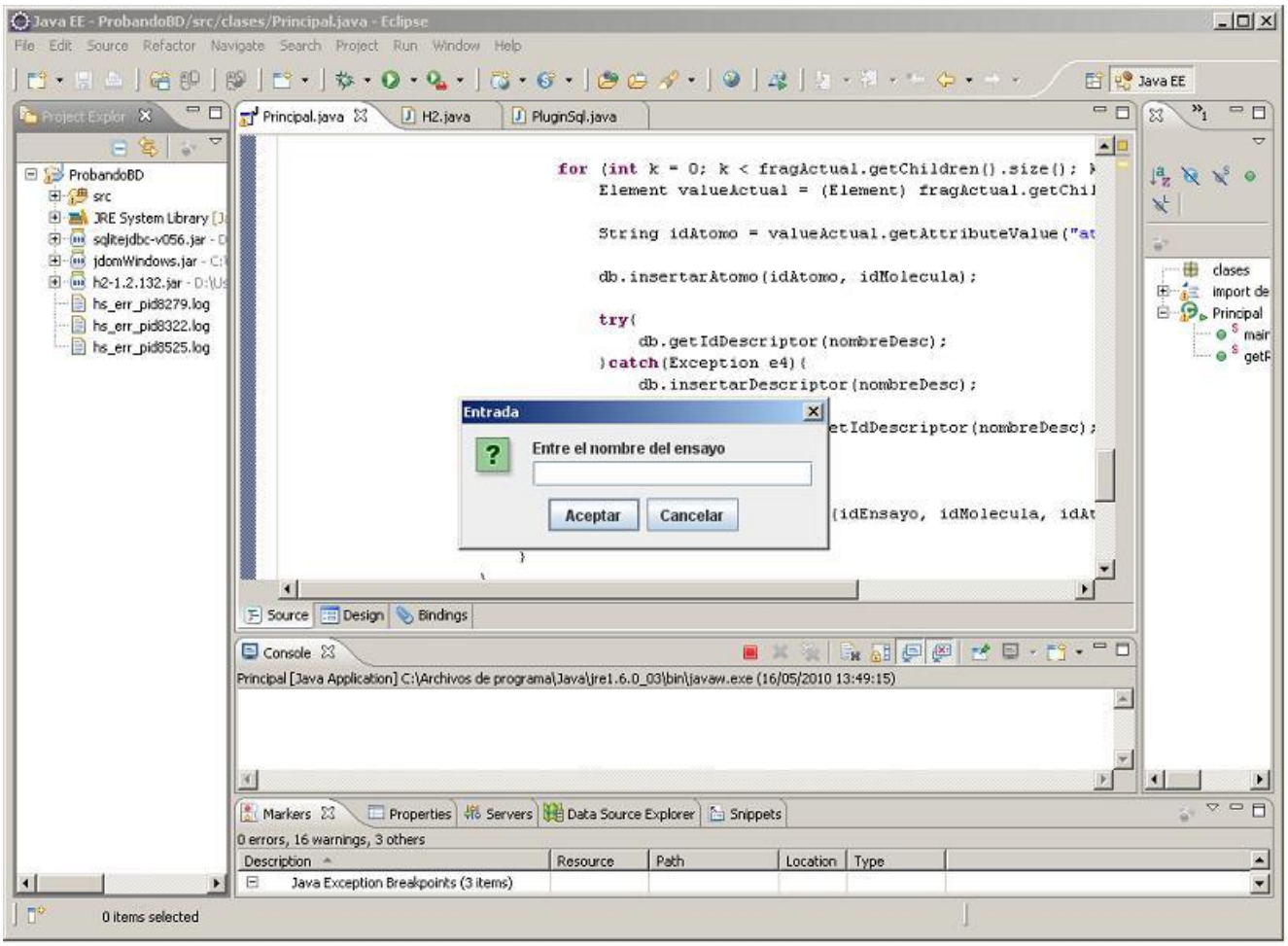
3.2 Incorporación de la base de datos seleccionada al módulo de Cálculo de Descriptores

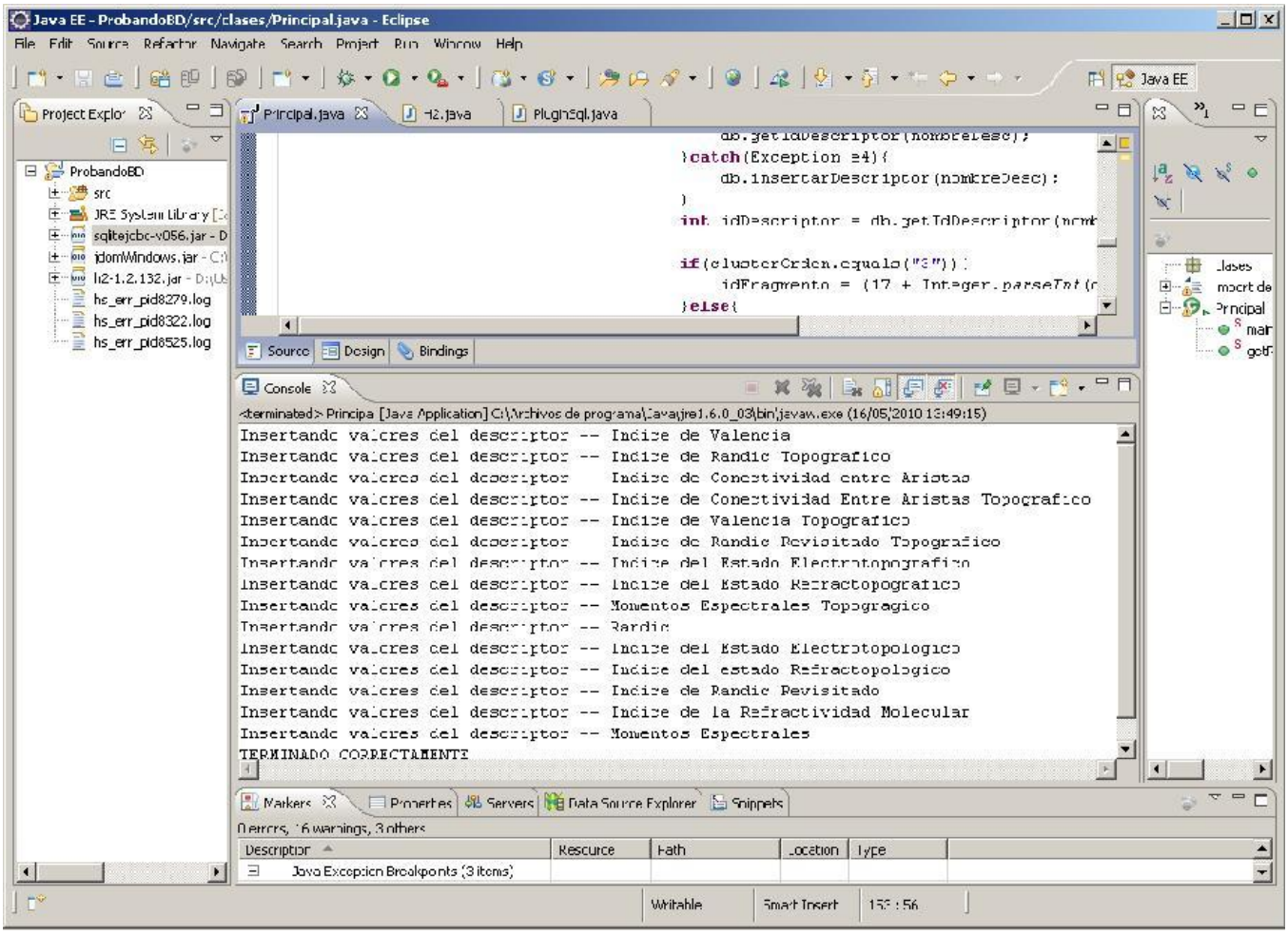
En el módulo Cálculo de Descriptores es donde se aplican algoritmos matemáticos a las estructuras moleculares, arrojando gran cantidad de resultados. Por motivos de centralización y de tener vías ágiles de acceder a la información es que se hace necesario el almacenar los resultados en una base de datos para tener mayor control y organización de los mismos.

3.2.2 Código fuente que incorpora H2

```
Public H2 (String db) {
 this.db = new File (db);
      }
      protected void conect () throws SQLException, ClassNotFoundException {
            System.out.println ("jdbc: h2:" + db.getAbsolutePath ());
            Class.forName ("org.h2.Driver");
            con = DriverManager
                        .getConnection ("jdbc: h2:" + db.getAbsolutePath ());
      }


Try {

                  File f = new File ("descDB.db");
                  if (!f.exists()){
                        f.createNewFile ();
                        PluginH2 db = new PluginH2 (f.getAbsolutePath ());


                        db.createDataBase ();
                  }

                  PluginH2 db = new PluginH2 (f.getAbsolutePath ());

                  File fXML = new File ("D:\\4.xml");
```

3.2.3 Imagen de una prueba corriendo





Conclusiones

Luego de la aplicación del procedimiento para la realización de pruebas de volumen y rendimiento, en especial carga y estrés a las bases de datos seleccionadas, donde se llegó a la conclusión que la base de datos según los resultados de las pruebas que la más eficiente es H2, que permite una rapidez del trabajo a la hora de insertar y extraer datos de la misma, teniendo así un mejor rendimiento. Además obtuvo los resultados más satisfactorios. En este capítulo se aplicó el procedimiento desarrollado en el capítulo 2, con el objetivo de demostrar su validez y de que sea utilizado para la realización de las pruebas de rendimiento.

CONCLUSIONES GENERALES

- Se selecciona H2 como la base de datos a incorporar en los diferentes módulos debido a los altos valores de respuesta según los casos de prueba realizados.
- La base de datos seleccionada se incorporó a través del módulo de Cálculo de Descriptores de la plataforma alasGrato.

RECOMENDACIONES

Luego de haber analizado los resultados del presente trabajo de diploma, resulta factible arribar a las siguientes recomendaciones:

- Incorporar H2 como base de datos empotrada a los demás módulos.

- Valorar la posibilidad de emplear H2 como base de datos interna, en el manejo de transacciones.

REFERENCIAS BIBLIOGRÁFICAS

1. Damián Pérez Valdés . maestrosdelweb. *maestrosdelweb.* [En línea] 2010. [Citado el: 11 de enero de 2010.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

2. SQLite. *SQLite.* [En línea] [Citado el: 25 de mayo de 2010.] http://www.somoslibres.org/modules.php?name=News&file=article&sid=183.

3. López, Antoni Aloy**.** *Empress Offers an Effective Embedded Database Solution", 2005.* 2009.

4. Netrino. *Netrino.* [En línea] 2009. [Citado el: 15 de enero de 2010.] http://www.netrino.com/Embedded-Systems/Glossary.

5. Apache Derby. *Apache Derby.* [En línea] 19 de mayo de 2010. [Citado el: 29 de mayo de 2010.] http://db.apache.org/derby/ .

6. Java.6.Derby. *java.6.Derby.* [En línea] 19 de marzo de 2010. [Citado el: 25 de mayo de 2010.] http://acercadejava.blogspot.com/2010/03/java-6-derby-la-base-de-datos-incluida.html.

7. SQLite. *SQLite.* [En línea] [Citado el: 15 de febrero de 2010.] http://www.sqlite.org/.

8. Aplicaciones Empresariales. *Aplicaciones Empresariales.* [En línea] 2010. [Citado el: 15 de abril de 2010.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

9. H2. *H2.* [En línea] 2010. [Citado el: 18 de abril de 2010.] http://www.h2database.com/html/main.html.

10. Db4objects. *db4objects.* [En línea] 2010. [Citado el: 15 de enero de 2010.] http://www.db4o.com/espanol/.

11. Oracle. *Oracle.* [En línea] 2010. [Citado el: 15 de abril de 2010.] http://www.oracle.com/technology/products/berkeley-db/index.html.

12. Equi4. *equi4.* [En línea] 2009. [Citado el: 16 de abril de 2010.] http://www.equi4.com/metakit/.

13. Aloy, Antoni López**.** Gadfly: SQL Relational Database in Python. *Gadfly: SQL Relational Database in Python.* [En línea] 2010. [Citado el: 15 de marzo de 2010.] http://gadfly.sourceforge.net/gadfly.html.

14. Daffodil. *Daffodil.* [En línea] 2009. [Citado el: 16 de marzo de 2010.] http://db.daffodilsw.com/server_edition.html.

15. Perst - An open source, object-oriented embedded database. *Perst - An open source, object-oriented embedded database.* [En línea] 2010. [Citado el: 25 de febrero de 2010.] http://www.mcobject.com/perst.

16. Metanotion BlockFile Database. *Metanotion BlockFile Database.* [En línea] 2006. [Citado el: 16 de febrero de 2010.] http://www.metanotion.net/software/sandbox/block.html.

17. Quadcap Embeddable Database. *Quadcap Embeddable Database.* [En línea] 2009. [Citado el: 22 de abril de 2010.] http://www.java2s.com/Open-Source/Java-Document/Database-DBMS/Quadcap-Embeddable-Database/CatalogQuadcap-Embeddable-Database.htm.

18. Martin Solari**.** Prueba de software. *Prueba de software.* [En línea] [Citado el: 25 de mayo de 2010.] http://athenea.ort.edu.uy/publicaciones/ingsoft/ortsf/areas/IntroduccionPrueba.pdf.

19. Testhouse .*testhouse.* [En línea] 2010. [Citado el: 25 de enero de 2010.] http://es.testhouse.net/index.php/servicios/pruebas-no-funcionales/pruebas-de-rendimiento.

20. Mijao. [En línea] 21 de 5 de 2007. [Citado el: 20 de febrero de 2010.] http://mijao.blogspot.com/2007/05/pruebas-en-soaarquitecturas-orientadas.html.

21. The Apache Jakarta Project. *The Apache Jakarta Project.* [En línea] 1999-2009. [Citado el: 26 de febrero de 2010.] http://jakarta.apache.org/jmeter/.

22. Friends of Helios. *Friends of Helios.* [En línea] 2010. [Citado el: 18 de enero de 2010.] http://www.eclipse.org/.

23. Java. *java.* [En línea] 2010. [Citado el: 12 de febrero de 2010.] http://www.java.com/es/download/index.jsp.

BIBLIOGRAFÍA

1. Apache Derby. *Apache Derby.* [En línea] 19 de mayo de 2010. [Citado el: 29 de mayo de 2010.] http://db.apache.org/derby/ .

2. Aloy, Antoni López**.** Gadfly: SQL Relational Database in Python. *Gadfly: SQL Relational Database in Python.* [En línea] 2010. [Citado el: 15 de marzo de 2010.] http://gadfly.sourceforge.net/gadfly.html.

3. Aplicaciones Empresariales. *Aplicaciones Empresariales.* [En línea] 2010. [Citado el: 15 de abril de 2010.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

4. Chaviano, Dayana Brunet y Martínez, Adrian Pérez. "Base de Datos del Simulador de Sistemas Biológicos: BioSyS Versión 2.0" .Ciudad de la Habana: s.n., 2009.

5.Daffodil. *Daffodil.* [En línea] 2009. [Citado el: 16 de marzo de 2010.] http://db.daffodilsw.com/server_edition.html.

6. Daswani, P., Rodrigo, J.J y Rosales, J. Medición de rendimiento de servicios WMS CON JMeter. Medición de rendimiento de servicios WMS CON JMeter. [En línea] [Citado el: 16 de abril de 2010.] http://blog.grafcan.es/resources/jidee08/086.pdf

7.Damián Pérez Valdés . maestrosdelweb. *maestrosdelweb.* [En línea] 2010. [Citado el: 11 de enero de 2010.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

8.Db4objects. *db4objects.* [En línea] 2010. [Citado el: 15 de enero de 2010.] http://www.db4o.com/espanol/.

9. Equi4. *equi4.* [En línea] 2009. [Citado el: 16 de abril de 2010.] http://www.equi4.com/metakit/.

10. Fernández, Luis. 2002. Presentación del Grupo de Calidad de Software. Presentación del Grupo de Calidad de Software. [En línea] 2002. [Citado el: 25 de febrero de 2009.] http://www.ati.es/spip.php?rubrique5

11. Friends of Helios. *Friends of Helios.* [En línea] 2010. [Citado el: 18 de enero de 2010.] http://www.eclipse.org/.

12. Guzmán Arenas, Adolfo. 2003. Mitos, creencias y supersticiones sobre la calidad del software y de su enseñanza. 2003.

13. Guzmán Cortés, Oscar Hernando. 2004. Aplicación práctica del diseño de pruebas de software a nivel de programación. Aplicación práctica del diseño de pruebas de software a nivel de programación. [En línea] 2004. [Citado el: 16 de abril de 2010.] http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf.

14.H2. *H2.* [En línea] 2010. [Citado el: 18 de abril de 2010.] http://www.h2database.com/html/main.html.

15. Java. *java.* [En línea] 2010. [Citado el: 12 de febrero de 2010.] http://www.java.com/es/download/index.jsp.

16. Java.6.Derby. *java.6.Derby.* [En línea] 19 de marzo de 2010. [Citado el: 25 de mayo de 2010.] http://acercadejava.blogspot.com/2010/03/java-6-derby-la-base-de-datos-incluida.html.

17. Jiménez, Darwin y Aguirre, Carlos Eduardo. Presentación de pruebas. Modelo de Pruebas de Software. [En línea] [Citado el: 5 de abril de 2010.] http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

18. JMeter, Jakarta. 2009. The Jakarta Site. Jakarta JMeter. [En línea] 2005. [Citado el: 13 de abril de 2010.] http://jakarta.apache.org/jmeter/.

19.López, Antoni Aloy**.** *Empress Offers an Effective Embedded Database Solution", 2005.* 2009.

20. Martin Solari**.** Prueba de software. *Prueba de software.* [En línea] [Citado el: 25 de mayo de 2010.] http://athenea.ort.edu.uy/publicaciones/ingsoft/ortsf/areas/IntroduccionPrueba.pdf.

21. Metanotion BlockFile Database. *Metanotion BlockFile Database.* [En línea] 2006. [Citado el: 16 de febrero de 2010.] http://www.metanotion.net/software/sandbox/block.html.

22. Mijao. [En línea] 21 de 5 de 2007. [Citado el: 20 de febrero de 2010.] http://mijao.blogspot.com/2007/05/pruebas-en-soaarquitecturas-orientadas.html.

23. MWAST. 2001. Microsoft Web Application Stress Tool. The Apache Software Fundation. [En línea] 12 de junio de 2001. [Citado el: 26 de abril de 2010.] http://people.apache.org/~sgoeschl/presentations/was-20010628.pdf.

24. Netrino. *Netrino.* [En línea] 2009. [Citado el: 15 de enero de 2010.] http://www.netrino.com/Embedded-Systems/Glossary.

25. Oracle. *Oracle.* [En línea] 2010. [Citado el: 15 de abril de 2010.] http://www.oracle.com/technology/products/berkeley-db/index.html.

26. Prueba de Carga Básica en JMeter. Prueba de Carga Básica en JMeter. [En línea] 2005. http://www.osmosislatina.com/jmeter/pruebabasica.htm.

27. Perst - An open source, object-oriented embedded database. *Perst - An open source, object-oriented embedded database.* [En línea] 2010. [Citado el: 25 de febrero de 2010.] http://www.mcobject.com/perst.

28. Quadcap Embeddable Database. *Quadcap Embeddable Database.* [En línea] 2009. [Citado el: 22 de abril de 2010.] http://www.java2s.com/Open-Source/Java-Document/Database-DBMS/Quadcap-Embeddable-Database/CatalogQuadcap-Embeddable-Database.htm.

29. Sánchez, Liudmila. 2008. Estrategia para Pruebas Automatizadas de Carga y Estrés. La Habana: s.n., 2008.

30.SQLite. *SQLite.* [En línea] [Citado el: 25 de mayo de 2010.] http://www.somoslibres.org/modules.php?name=News&file=article&sid=183.

31.SQLite. *SQLite.* [En línea] [Citado el: 15 de febrero de 2010.] http://www.sqlite.org/.

32. Testhouse. *testhouse.* [En línea] 2010. [Citado el: 25 de enero de 2010.] http://es.testhouse.net/index.php/servicios/pruebas-no-funcionales/pruebas-de-rendimiento.

33. The Apache Jakarta Project. *The Apache Jakarta Project.* [En línea] 1999-2009. [Citado el: 26 de febrero de 2010.] http://jakarta.apache.org/jmeter/.

34. Valencia, Gustavo. Servicio de Pruebas de Software. Servicio de Pruebas de Software. [En línea] http://www.greensqa.com/archivos/Servicio%20Pruebas%20de%20Software.pdf.

ANEXOS

Anexo 1: Plantilla para Casos de Prueba de rendimiento usando la herramienta JMeter:

Tipo de Prueba (Carga o Estrés): [Se hace necesario antes de la realización de las pruebas, la revisión y análisis del manual de uso del JMeter para efectuar pruebas de rendimiento a las bases de datos]

Prueba de Carga: [En caso de que la prueba a realizar sea Prueba de Carga] La prueba de carga tiene como objetivo comprobar la respuesta del sistema una vez que se interactúe con él y se simule el trabajo de una cantidad esperada de usuarios que introduzcan datos y naveguen por el sitio, comprobando si el sistema es capaz de responder correctamente bajo condiciones extremas. En esta prueba se introducen datos en el sistema para ver cómo responde la Base de datos ante las peticiones de información.

Variables	**Valores**
Número de usuarios concurrentes: [Número de hilos]	
Tiempo entre conexión y conexión: [Período de subida (en segundos)]	[cero por defecto, pues todas las conexiones se realizan de forma simultánea]
Número de iteraciones: [Contador del bucle]	[Tiene como valor 1 por defecto, pues la iteración se realiza una sola vez.]

Prueba de Estrés: [En caso de que la prueba a realizar sea de Estrés].

La prueba de estrés tiene como objetivo comprobar cómo responde el sistema bajo una serie de peticiones que se incrementarán con el paso del tiempo al doble, hasta llegar a un valor extremo. En este tipo de prueba solo se comprueba la conexión a las interfaces, pero sin el intercambio de información, por lo que no se efectúa la entrada de datos al sistema.

Variables	**Valores**
Número de usuarios concurrentes: [Número de hilos]	
Tiempo entre conexión y conexión: [Período de subida (en segundos)]	[Tiempo que ocurrirá entre la conexión de los usuarios de la primera iteración y todas las demás]

Número de iteraciones: [Contador del bucle]	[Cantidad de veces que se repetirá la simulación de conexión de usuarios]

Resultados generales:

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg	% CPU

[Petición: Número de la petición]

[Los demás datos de la tabla se encuentran en el resultado de la prueba efectuada con el Software JMeter]

Media (Average): matemáticamente se refiere al promedio del juego de datos. Es una medida que por sí sola dice muy poco sobre los resultados, pero nos da una idea de cuál es el comportamiento general de la aplicación.

Mediana (Median): Simplemente es el valor medio del juego de datos cuando estos son ordenados de menor a mayor. En los casos en que la cantidad de datos es par, se toman los dos valores centrales y se saca el promedio de estos.

Desviación Estándar (Standard Deviation): es una medida de dispersión para variables de razón (ratio o cociente) y de intervalo, de gran utilidad en la estadística descriptiva. Es una medida (cuadrática) que informa de la media de distancias que tienen los datos respecto de su media aritmética, expresada en las mismas unidades que la variable. Una norma utilizada para esta métrica es: “Los datos con una desviación estándar superior a la mitad de su media debe ser tratada como sospechosa. Si es exactamente ese valor, el juego de datos no tiene una distribución normal”.

Rendimiento (Throughput): Es la tasa promedio de mensajes entregados satisfactoriamente. Comúnmente se mide en peticiones/segundos/minutos/horas.

Latencia (Latency): Tiempo necesario para reunir la solicitud y el montaje de la respuesta.

Línea del 90% (90% Line): Es el tiempo máximo que toman el 90% de las muestras más rápidas. Las muestras restantes se demoran más.

Conclusiones: [Las conclusiones se dan en base al rendimiento, la cantidad de peticiones y los errores obtenidos].

GLOSARIO DE TÉRMINOS

Actividad Biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos.

ACID: Es conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

Algoritmo AES: Es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos

API: (application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos).

Base de datos: Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Base de datos relacional: Es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

Caso de prueba*:* Conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito. También se puede referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba.

Cliente-Servidor: Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

CMS: Es un programa que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas web, por parte de los participantes.

Compuestos Orgánicos: Compuestos cuya composición fundamental es sobre la base del elemento químico carbono.

CPU: Circuito microscópico que interpreta y ejecuta instrucciones.

C++: Lenguaje de programación, diseñado como extensión del lenguaje de programación C.

C#: Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET

Gestión de la información: Es un proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y colaboración de la información, y que gestiona el acceso y los derechos de los usuarios sobre la misma.

Hardware*:* Conjunto de componentes físicos de una computadora. Refiérase a objetos tangibles y palpables como son los discos, lectores de discos, monitores, teclados, las impresoras, tarjetas y chips.

Herramienta: Subprograma o módulo encargado de funciones específicas y afines entre sí para realizar una tarea. Una aplicación o programa puede contar con múltiples herramientas a su disposición. Por ejemplo, el corrector ortográfico puede ser una herramienta en una aplicación para redactar documentos, pero no es una aplicación en sí misma.

JDBC: (Java Database Connectivity) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java

Linux: Sistema operativo de código abierto.

Licencia GPL: Es una licencia creada para proteger la libre distribución, modificación y uso de software.

Metodología: Es un conjunto de procedimientos, técnicas, instrumentos y documentos que ayudan a los analistas y programadores en sus esfuerzos para obtener un nuevo sistema informativo. Consiste en fases que guían al diseñador en la elección de las técnicas más apropiadas en cada paso del proyecto y lo ayudan a planificar, dirigir, controlar y evaluar el mismo.

MySQL: Sistema Gestor de Base de Datos.

Norma SQL 92: Es un estándar para la sintaxis y semántica de los lenguajes de bases de datos.

Open Source: Cualidad de algunos software de incluir el código fuente en la distribución del programa. En general se usa para referirse al software libre.

PC: Una computadora personal u ordenador personal

Perl: Lenguaje de programación diseñado con características del lenguaje C.

PHP: Lenguaje de programación para el desarrollo de software.

Prueba: Prueba de software. Ejecución de un sistema bajo condiciones específicas, se observan y se analizan los resultados realizándose una evaluación de los mismos.

Python: Lenguaje de programación para el desarrollo de software.

Portabilidad: Se define como la característica que posee un software para ejecutarse en diferentes plataformas

Ruby: Lenguaje de programación interpretado, reflexivo y orientado a objetos, Combina una sintaxis inspirada en Python y Perl .

Software: Es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

SQL: Es un Lenguaje de consulta estructurado.

TCL: (Tool Command Language) Es un lenguaje interpretado, y su código puede ser creado y modificado dinámicamente.

Windows: Sistema operativo propietario.