UNIVERSIDAD NACIONAL SAN LUIS GONZAGA DE ICA

FACULTAD DE INGENIERIA DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

UNIVERSIDAD NACIONAL
'SAN LUIS GONZAGA' DE ICA

TESIS

PARA OPTAR EL TITULO DE INGENIERO DE SISTEMAS

"IMPACTO E INFLUENCIA DE LAS BASES DE DATOS OPEN SOURCE Y LA DIFERENCIA EN SU ELECCION"

PRESENTADO POR:

BACHILLERES:

Guerrero Valencia, Luis Alberto

Pacheco Vargas, Ricardo Raúl

ASESOR: DR. ERWIN PABLO PEÑA CASAS

ICA-PERÚ

2017

DEDICATORIA

A mis padres que me inculcaron valores y me ayudaron
A formarme integralmente como profesional
En ingeniería de sistemas ya que ellos son
El motor que me impulsa a seguir adelante
Para lograr mis aspiraciones y objetivos.

Luis Alberto Guerrero Valencia

DEDICATORIA

A Dios por dotarme de competencias
Que me permitan afrontar los retos
De un mundo globalizado usando
Como herramienta la tecnología y
Aportar al desarrollo de la sociedad.

Ricardo Raúl Pacheco Vargas

TABLA DE CONTENIDOS

Dedicatorias i

INDICE iv

INTRODUCCION 1

RESUMEN 2

CAPÍTULO I: PLANTEAMIENTO METODOLOGICO 4

1.1 Descripción de la Realidad Problemática 4

1.2 Delimitaciones y Definición del Problema 5

1.2.1. Delimitaciones 5

1.2.2. Definición del Problema 6

1.3. Formulación del Problema. 6

1.4. Objetivo de la Investigación 7

1.5. Hipótesis de la investigación 7

1.6. Variables e Indicadores 7

1.7. Justificación e Importancia de la Investigación 8

1.7.1. Justificación 8

1.7.2. Importancia 8

1.8. Limitaciones de la Investigación 8

1.9. Tipo y Nivel de la Investigación 9

1.9.1. Tipo de Investigación 9

1.9.2. Nivel de Investigación 9

1.10 Método y Diseño de la investigación 9

1.10.1. Método de la investigación 9

1.10.2. Diseño de la investigación 10

1.11. Técnicas e Instrumentos de Recolección de Información 10

1.11.1. Técnicas 10

1.11.2. Instrumentos 10

1.12. Cobertura de Estudio 11

CAPITULO II: MARCO TEORICO 12

2.1 Antecedentes de la Investigación 12

2.2 Marco Histórico 13

2.3 Marco Conceptual 15

CAPÍTULO III: RECOPILACION DE INFORMACION 54

3.1. Recopilación de los Datos 54

3.2. Pruebas de tiempo de respuesta de manejo de datos 54

CAPÍTULO IV: ANALISIS E INTERPRETACION DE RESULTADOS 58

4.1. Población y muestra 58

4.2. Nivel de confianza y grado de significancia 58

4.3. Tamaño de la muestra representativa 58

4.4. Análisis e interpretación de resultados 59

4.5. Prueba de Hipótesis 62

4.6. Prueba estadística utilizada 62

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES 72

1.1. Conclusiones 72

1.2. Recomendaciones 73

REFERENCIAS BIBLIOGRAFICAS 74

ANEXOS 77

INTRODUCCION

En la actualidad en que los sistemas de información avanzan de una manera vertiginosa se presenta la gran disyuntiva de toda empresa en preguntarse cuál será el motor de base de datos convenientes para mi empresa, se hace importante que dentro de esta información encontremos a disponibilidad además una serie de sistemas de base de datos disponibles en forma libre y que pueden ser la solución a muchos de los problemas de almacenamiento de la información de empresas y usuarios; es por ello del presente estudio denominado **"IMPACTO E INFLUENCIA DE LAS BASES DE DATOS OPEN SOURCE Y LA DIFERENCIA EN SU ELECCION",** cuyo objetivo es el de encontrar los sistemas de base de datos de uso libre que están disponibles en los sitios web de las organizaciones que las promueven.

El estudio de los motores de base de datos motivo de la presente tesis, que han sido seleccionado son los motores de base de datos representativos dentro de los software de base de datos libre que más relevancia tienen en el mundo de la informática: SQLite, HiperSQL y Firebird, cada uno de estos presentando sus ventajas y desventajas estudiadas en el estudio, hacen posible poner a disposición de toda organización y personas que deseen tenerla sin costo algún, con las características de motores de base de datos de nivel comercial.

Es de reconocer para los que hemos cursados estudios de ingeniería de sistemas, la importancia de las bases de datos en la estructura del área de TI de una empresa y más aún la importancia de los datos de una empresa motivo por el cual ha sido desarrollado el tema de investigación.

RESUMEN

La presente investigación tuvo por objetivo establecer una diferencia entre los motores de base de datos libre SQLite, HiperSQL y Firebird, esta información está basada en la comparación del impacto funcional de estas tres bases de datos.

En esta tesis se presenta la siguiente estructura desarrollada:

En el capítulo I: En el planteamiento metodológico, se especifica los puntos ordenados de la metodología de la investigación, con la finalidad de darle al estudio una estructura formal y rigurosa de estudio.

En el capítulo II del Marco Teórico, Se detalla las características de cada uno de los motores de base de datos, sus ventajas, desventajas, modos de instalación para cada uno de ellos.

En el capítulo III de la Recopilación de los datos, se utilizó la información de una tabla de datos de empleados obtenida de la Base de datos de Oracle; los datos fueron importado a los motores de base de datos y se hicieron las pruebas en cada uno de los motores para ir registrando los resultados en los cuadros que se muestran en este capítulo.

En el capítulo IV, Análisis e Interpretación de los resultados; se procesan los datos de los cuadros del capítulo III, y se hace un análisis estadístico descriptivo para cada uno de los indicadores; asimismo se realiza las pruebas de hipótesis en base a los indicadores INSERT donde el valor de la probabilidad es de 0.88 mayor al 0.05 de la significancia. Por lo que no existe diferencia significativa, para DELETE el valor de la probabilidad es de 0.177 mayor al 0.05 de la significancia. Por lo que no existe diferencia significativa, para UPDATE el valor de la probabilidad es de 0.749 mayor al 0.05 de la

significancia. Por lo que no existe diferencia significativa y para SELECT el valor de la probabilidad es de 0.92 mayor al 0.05 de la significancia. Por lo que no existe diferencia significativa, para los tres motores de base de datos; en estas pruebas se utilizó la prueba ANOVA, para las tres muestras.

Finalmente en el Capítulo V, se presentan las conclusiones y recomendaciones resultantes del estudio de la tesis.

CAPÍTULO I: PLANTEAMIENTO METODOLOGICO

1.1. Descripción de la Realidad Problemática

Dado que la información está llevando a las a tener que modificar y decidir cuál es la base de datos adecuada para su empresa y que este se encargue de apoyar a sus procesos de negocios; esto está obligando a las empresas a tener un mejor control sobre sus datos su procesamiento y la obtención de información que les permita poder competir en un medio cada vez más competitivo, ágil y flexible.

Un factor importante en el manejo de los datos de las empresas está relacionada a los sistemas de base de datos; pudiendo ser éstas base de datos comerciales o base de datos libres; en tal sentido estas dos categorías de las bases de datos están siendo utilizadas con marcado éxito por las empresas, sin embargo por la diversidad de base de datos libres, se presentas algunas interrogantes como:¿y qué sistema de base de datos libres, será la más adecuada?, ¿Cuál será la que más beneficios nos proveea?, en tal sentido, todos los sistemas de base de datos libres utilizan un único lenguaje para administrar los datos contenidos en ella. El lenguaje estructurado de consultas (SQL: Structure Query Languaje).

Finalmente desconocer, que diferencias existen en los motores de base de datos libres es una necesidad actual que se está desarrollando en forma vertiginosa.

1.3 Delimitaciones y Definición del Problema

1.3.1 Delimitaciones

A. Delimitación Espacial.

La delimitación del estudio del presente proyecto de investigación no se contempla porque solo se va a demostrar la diferencia entre 3 motores de base de datos, para ello se hará uso intensivo del internet como complemento de la investigación.

B. Delimitación Social

Según la naturaleza del trabajo investigativo, los roles sociales que intervienen son:

- ✓ Los Investigadores
- ✓ Asesor

C. Delimitación Conceptual.

Base de datos Libres, se considera a una base de datos libres, aquella que puede ser utilizada sin restricción de ninguna naturaleza. La libertad al uso, no teniendo ninguna relación con la base de datos el costo.

Impacto funcional, se refiere a las características funcionales relacionadas con el manejo y administración de los datos; enfocados en el almacenamiento, la actualización,

eliminación de los datos, y de las consultas para obtener información.

1.3.2 Definición del Problema

En la actualidad los sistemas de base de datos, son muy importantes para el almacenamiento de la información, las empresas pequeñas y medianas tienen el problema de costos de adquisición de las licencias de uso de los motores de base de datos comerciales como Sql server de Microsoft, Oracle de Oracle, etc, este problema podría solucionarse si se tienen una real información sobre los sistemas de base de datos libres que tienen las mismas prestaciones que los comerciales, pero que sin embargo existe muy poca información sobre el aspecto funcional de los motores de base de datos analizados.

1.4 Formulación del Problema.

Problema principal.

¿Cuál será las diferencias del impacto funcional de los sistemas de BD libres SQLite, HiperSQL y Firebird?

1.5 Objetivo de la Investigación

Objetivo General: Determinar las diferencias del impacto funcional de los sistemas de BD libres SQLite, HiperSQL y Firebird.

Objetivos Específicos:

- Evaluar la capacidad de almacenamiento de las bases de datos libres SQLite, HiperSQL y Firebird.
- Analizar los tiempos de respuesta en el manejo de los datos de las bases de datos libres SQLite, HiperSQL y Firebird.
- Determinar el tiempo de respuesta de las consultas de las bases de datos libres SQLite, HiperSQL y Firebird.

1.6 Hipótesis de la investigación.

Existirá una diferencia entre la utilización de los sistemas de Base de datos OPEN SOURCE SQLite, HiperSQL y Firebird.

1.7 Variables e Indicadores

1.7.1 Variable Independiente

X= Sistemas de BD libres, SQLite, HiperSQL y Firebird

1.6.2 Variable Dependiente

Y= Diferencia en su aplicación

A. Indicadores

Y1: Capacidad de almacenamiento

Y2: Tiempo utilizado para el ingreso de datos

Y3: Tiempo utilizado para la actualización de datos

Y4: Tiempo utilizado para la eliminación de datos

Y5: Tiempo utilizado para la búsqueda de información

1.8 Justificación e Importancia de la Investigación.

1.8.1 Justificación

El presente estudio se justifica, ya que se determinara cual es el motor de base de datos libre que podría ser utilizado para cualquier empresa beneficiando a muchos con su utilización.

1.8.2 Importancia

Es importante el estudio que se tiene que realizar, porque nos va a permitirá evaluar algunos de los motores de base de datos Open Source y determinar en forma global la existencia o no de funcionalidades, que orienten en forma general a las personas y empresas para el uso y manejo de la base de datos que le permita un mejor beneficio en el tratamiento de los sistemas de información de los datos de su empresa.

1.9 Limitaciones de la Investigación

No se encuentran limitaciones para el presente trabajo de tesis, ya que se cuenta con información de las bases de datos en estudio y el software para realizar las pruebas está disponible en la web.

1.10 Tipo y Nivel de la Investigación

1.10.1 Tipo de investigación

La naturaleza de este trabajo de investigación es **Aplicada** también denominada activa o dinámica y se encuentra ligada íntimamente a la investigación pura, ya que depende de sus descubrimientos y aportes teóricos. Busca confrontar la teoría con la realidad además de estar dirigida a su aplicación inmediata y no al desarrollo de teorías.[1]

1.10.2 Nivel de investigación.

El trabajo de investigación comienza a **Nivel descriptivo,** porque a ese nivel se describe la naturaleza del proceso de enseñanza – aprendizaje. El trabajo finaliza a **Nivel correlacional**, porque se mide la influencia de la variable independiente, que es el nivel de aprendizaje de la capacidad de los investigadores.

1.11 Método y Diseño de la investigación

1.11.1 Método de la investigación.

Para llevar a cabo el desarrollo de este trabajo de investigación se utilizará el **método científico** por ser el método que aborda en conjunto el método deductivo, el método inductivo y el método hipotético-deductivo para así poder realizar un planteamiento ordenado además de un alto nivel de rigurosidad, exigencia y exactitud que amerita el tratamiento de los datos y el respectivo

[1] Tamayo y Tamayo, Mario. El proceso de la Investigación Científica.4ª ed., México, 2004, Ed. Limusa, 43pp.

análisis de los resultados, por consiguiente se seguirá entonces un método que permite la comprobación o contrastación de la hipótesis anteriormente planteada.[2]

1.11.2 Diseño de la investigación.

El diseño de la investigación es No experimental, transversal ya que posee la característica de que los investigadores controlan y manipulan deliberadamente las condiciones que determinan los hechos en los que trabaja, para después observar los efectos que se producen. -[3].

1.12 Técnicas e Instrumentos de Recolección de Información

1.12.1 Técnicas.

Pruebas de BD

1.12.2 Instrumentos.

Ficha de resultados

1.13 Cobertura de Estudio

1.13.1 Universo.

La investigación se aplica a todos los sistemas de BD libres

1.13.2 Muestra

La muestra seleccionada es de tipo intencional, para las tres BD libre a analizarse.

[2]Klimovsky Gregorio, Las Desventuras del Conocimiento Científico- Una introducción a la epistemología. 6ª ed., 1997, A-Z Editora, 418pp.
[3] Escribano González, Alicia. Aprender a Enseñar Fundamentos de Didáctica General. 3ª ed., España, 2008, Gráficas Cuenca, 348pp.

CAPÍTULO II: MARCO TEORICO

2.4 Antecedentes de la Investigación

1. Rafael Camps Paré, Introducción a las BD, http://dataprix.com/files/UOC_OpenSource_Introduccion_a_las_bases_de_datos.pdf.

   Resumen: En esta unidad hemos hecho una introducción a los conceptos fundamentales del mundo de las BD y de los SGBD. Hemos explicado la evolución de los SGBD, que ha conducido de una estructura centralizada y poco flexible a una distribuida y flexible, y de una utilización procedimental que requería muchos conocimientos a un uso declarativo y sencillo.

   Hemos revisado los objetivos de los SGBD actuales y algunos de los servicios que nos dan para conseguirlos. Es especialmente importante el concepto de transacción y la forma en que se utiliza para velar por la integridad de los datos.

   La arquitectura de tres niveles aporta una gran flexibilidad a los cambios, tanto a los físicos como a los lógicos. Hemos visto cómo un SGBD puede funcionar utilizando los tres esquemas propios de esta arquitectura.

   Hemos explicado que los componentes de un modelo de BD son las estructuras, las restricciones y las operaciones. Los diferentes modelos de BD se diferencian básicamente por sus estructuras. Hemos hablado de los modelos más conocidos, especialmente del modelo relacional, que está basado en tablas.

2. Ernesto Quiñones Azcárate**. Charlas: Sistemas Administradores de Bases de Datos Libres para el entorno empresarial.** Sociedad nacional de Industrias. Lima, Http://www.eqsoft.net.
3. PLoS Medicine, articulo: Base de datos libres, para ensayos clínicos. http://www.scidev.net/es/south-asia/opinions/bases-de-datos-libres-para-los-ensayos-cl-nicos.html. marzo 2008.

Resumen:

Los ensayos que evalúan tratamientos para enfermedades olvidadas se hacen más difíciles por la escasez en los sistemas de software que manejan la información acorde con las guías internacionales, dicen a Gregorio W. Fegan y Trudie A. Lang en PLOS Medicine.

Argumentan que el prohibitivo costo de los programas relacionados con ensayos clínicos deja a muchas instituciones de países en desarrollo con una opción nada envidiable tanto por no ser capaces de cumplir con las normas internacionales, como por tener la necesidad de enviar los datos del ensayo fuera de sus instituciones para ser procesados.

Los autores alientan a los donantes internacionales a impulsar el uso de sistemas de base de datos de código abierto: “Así como los financiadores de investigación biomédica comienzan a requerir información científica para ser publicada en revistas de libre acceso, ¿no podrían ellos necesitar que el software usado para la gestión de ensayos clínicos también sea abierto?".

Fegan y Lang dicen que las ventajas incluirían el estímulo para que los investigadores en países pobres y de bajos recursos participen en investigaciones de alto nivel que, de otra manera, estarían fuera de su alcance y más allá de sus capacidades. Esto, al final, aumentaría el alcance y la variedad de los ensayos.

4. Ernesto Quiñones Azcárate. **Comparativas de PostgreSQL vs otros DBMS Libres.** Pg. Day 2007. www.postgresql.org.pe

2.2. Marco Histórico

2.2.1. Historia de la Base de Datos

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963.

En una primera aproximación, se puede decir que una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada.

Desde el punto de vista informático, una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Por su parte, un sistema de Gestión de Bases de datos es un tipo de software muy específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; o lo que es lo mismo, una agrupación de programas que sirven para

definir, construir y manipular una base de datos, permitiendo así almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Actualmente, las bases de datos están teniendo un impacto decisivo sobre el creciente uso de las computadoras.

Pero para poder entender más profundamente una base de datos cabe entender su historia.

2.2.2. Orígenes

Los orígenes de las bases de datos se remontan a la Antigüedad donde ya existían bibliotecas y toda clase de registros. Además también se utilizaban para recoger información sobre las cosechas y censos. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual.

Posteriormente, el uso de las bases de datos se desarrolló a partir de las necesidades de almacenar grandes cantidades de información o datos. Sobre todo, desde la aparición de las primeras computadoras, el concepto de bases de datos ha estado siempre ligado a la informática.

En 1884 Herman Hollerith creó la máquina automática de tarjetas perforadas, siendo nombrado así el primer ingeniero estadístico

de la historia. En esta época, los censos se realizaban de forma manual.

2.3. Marco Conceptual

2.3.1. Base de Datos SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB)[2] biblioteca escrita en C. SQLite es un proyecto de dominio público[1] creado por D. Richard Hipp.

A diferencia de los sistema de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, **SQLite** permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

El autor de SQLite ofrece formación, contratos de soporte técnico y características adicionales como compresión y cifrado.

Características

La biblioteca implementa la mayor parte del estándar SQL-92, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, y durabilidad (ACID), triggers y la mayor parte de las consultas complejas.

SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un *string* en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). Algunos usuarios consideran esto como una innovación que hace que la base de datos sea mucho más útil, sobre todo al ser utilizada desde un lenguaje de scripting de tipos dinámicos. Otros usuarios lo ven como un gran inconveniente, ya que la técnica no es portable a otras bases de datos SQL. SQLite no trataba de transformar los datos al tipo de la columna hasta la versión 3.

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente. En caso contrario, el acceso de escritura falla devolviendo un código de error (o puede automáticamente reintentarse hasta que expira un

tiempo de expiración configurable). Esta situación de acceso concurrente podría cambiar cuando se está trabajando con tablas temporales. Sin embargo, podría producirse un interbloqueo debido al multihilo.[1] Este punto fue tratado en la versión 3.3.4, desarrollada el 11 de febrero de 2006.

Existe un programa independiente de nombre sqlite que puede ser utilizado para consultar y gestionar los ficheros de base de datos SQLite. También sirve como ejemplo para la escritura de aplicaciones utilizando la biblioteca SQLite.

Lenguajes de Programación

- La biblioteca puede ser usada desde programas en C/C++, aunque están disponibles enlaces para Tcl y muchos otros lenguajes de programación interpretado.
- SQLite se encuentra embebido en el REALbasic framework, haciendo posible que aplicaciones desarrolladas en REALbasic para Windows, Linux o Mac OS X usen la base de datos SQLite.
- Existe un módulo DBI/DBD para Perl disponible en CPAN, *DBD::SQLite*, no es una interface para SQLite, sino que incluye el motor completo de SQLite en sí mismo por lo cual no necesita ningún software adicional.
- Python incluye soporte para SQLite nativamente desde la versión 2.5 incorporado en la Biblioteca Estándar como el módulo sqlite3.[3] Para versiones anteriores de Python, el

módulo no está incorporado y debe instalarse (su nombre es PySQLite).[4]

- Hay otro módulo para Visual Basic 6 llamado VBSqlite
- Desde Delphi se puede usar SQLite a través de los componentes libres ZeosLib.
- PHP incluye SQLite, desde la versión 5. SQLite también funciona con PHP 4 pero no viene incluido en él. Para más detalles vea el manual y PECL info.
- Desde Java se puede acceder mediante el driver de SQLite JDBC
- Desde .NET se puede acceder usando el proyecto de código abierto System.Data.SQLite
- Desde Lazarus 0.9.8 y Free Pascal 2.0.0, SQLite está disponibles para programadores de Pascal. Tutorial: «Lazarus Database Tutorial, Lazarus and SQLite» (en inglés).
- Mac OS X v10.4 incluye SQLite, y es una de las opciones en la Core Data API de Apple. AppleScript puede abrir, crear, y manipular base de datos SQLite por medio de la aplicación de ayuda "Database Events" de Mac OS X 10.4.
- BlitzMAX posee un MOD que permite trabajar con bases de datos SQLite. Para más detalles y descarga del MOD vea [2].
- El componente de base de datos (gb.db) de Gambas soporta SQLite en sus versiones 1, 2 y 3
- El lenguaje de programación de vídeo juegos Bennu tiene un mod de SQlite disponible

- El lenguaje de programación de scripting para Windows AutoIt v.3.x a través de la DLL SQLite.dll.

Software que utiliza SQLite

SQLite es utilizado en una gran variedad de aplicaciones, destacando las siguientes:

- Adobe Photoshop Elements utiliza SQLite como motor de base de datos en su última versión del producto (la 6.0) en sustitución del Microsoft Access, utilizado en las versiones anteriores.[5]
- Clementine usa SQLite para guardar su colección de datos por defecto.
- Kexi usa SQLite como un motor de base de datos interno por defecto.
- Mozilla Firefox usa SQLite para almacenar, entre otros, las cookies, los favoritos, el historial y las direcciones de red válidas.[5]
- Los desarrolladores de OpenOffice.org han considerado incluir SQLite en el modelo de base de datos de Base, pero esto depende en gran manera del progreso de sqlite-sdbc-driver, que está todavía en estado de alpha. Actualmente han decidido usar HSQLDB.
- Varias aplicaciones de Apple utilizan SQLite, incluyendo Apple Mail y el gestor de RSS que se distribuye con Mac OS X. El software Aperture de Apple guarda la información de las

imágenes en una base de datos SQLite, utilizando la API Core Data.[5]

- El navegador web Opera usa SQLite para la gestión de bases de datos WebSQL.
- Skype es otra aplicación de gran despliegue que utiliza SQLite.[6] [5]
- SQLFilter, un plugin para OmniPeek, usa SQLite para indexar paquetes en una base de datos para poder se consultada por medio de SQL.
- The New Yorker guarda el índice para un set de DVD conteniendo todos los números publicados por la revista.
- XBMC Media Center (antes conocido como "XBox Media Center") es un reproductor de medios de audio, video, fotos, etc de código libre (open source) multi-plataforma a la vez que un centro de entretenimiento. Usa SQLite para administrar las librerías de música, video y fotografías, listas de reproducción y bookmarks entre otras utilidades menores.

2.3.2. Base de Datos HSQLDB

HSQLDB es un sistema de gestión de base de datos relacionales escrito en Java. Como principal ventaja tiene su velocidad y su reducido tamaño. Además, puede mantener la base de datos en memoria o en ficheros en disco.

Se pueden realizar las operaciones más habituales de los sistemas de gestión de bases de datos (altas, bajas,

modificaciones, consultas) usando sintaxis SQL, soporta triggers, integridad referencial (claves extranjeras)…

Como curiosidad, HSQLDB es el gestor de base de datos usado por “Base”, el equivalente de Access dentro de Open Office

Características de **HSQLDB**:

- Escrito por completo en Java
- Completo sistema gestor de bases de datos relacional
- Tiempo de arranque mínimo y gran velocidad en las operaciones: SELECT, INSERT, DELETE y UPDATE
- Sintaxis SQL estándar
- Integridad referencial (claves foráneas)
- Procedimientos almacenados en Java
- Triggers
- Tablas en disco de hasta 8GB

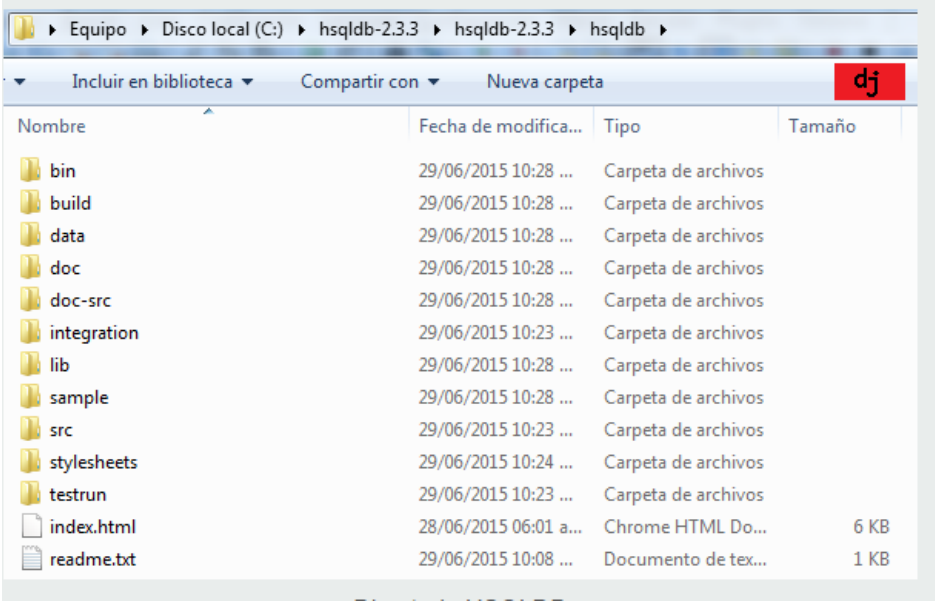
Consultas sobre una base de datos HSQLDB

Como en cualquier otro DBMS (Sistema de gestión de bases de datos), además de poder llamarlo desde nuestra aplicación, es interesante poder lanzar consultas sobre la base de datos de manera autónoma, ya sea para probar una query antes de incluirla en un programa, o para comprobar que los datos insertados por la aplicación son correctos. Ejemplos de productos que proporcionan esta funcionalidad son, entre otros Mysql Query Browser o SQLyog para MySQL, o TOAD para Oracle. HSQLDB

también tiene esta caraterística, usando una utilidad incluida en la distribución estándar.

Descargar HSQLDB

Vamos a ir a la página de sourceforge y descargamos el archivo `hsqldb-2.3.3.zip`. Lo descomprimimos (De preferencia en la raíz de C:) y podemos ver que dentro trae una gran cantidad de archivos, directorios... Algo así deberíamos ver



En la carpeta lib, se encuentra el archivo hsqldb.jar que vamos a importar al classpath de nuestro proyecto.

Bases de Datos en HSQLDB.

Hay varios tipos, en HSQLDB las bases de datos son llamadas "Catalogos", hay tres tipos de catálogos:

- **mem:** este tipo de catálogo se guarda directamente en la memoria RAM, no hay ningún tipo de persistencia, se cierra el proyecto de Java y se borran todos los datos ingresados anteriormente.

- **file:** se guarda en archivos en la raíz del proyecto Java.
- **res:** se guarda en el mismo proyecto Java, como un archivo JAR o ZIP.

mem: *todo en la memoria*, este tipo de catalogos se pueden usar para hacer pruebas de datos, hay que recordar que este tipo de bases de datos no crea nigún archivo.

file: este tipo de catálogos consiste en crear varios archivos, todos con el mismo nombre pero con diferentes extensiones, localizados en el mismo directorio. Por ejemplo si la base de datos la llamamos "prueba", estos son los archivos que se van a crear:

- prueba.properties: contiene la configuración de la base de datos.
- prueba.script: cotiene las definiciones de las tablas y objetos.
- prueba.log: contiene los cambios recientes que se la hayan hecho a la base de datos.
- prueba.data: contiene los datos de las tablas. (A veces no se crea)
- prueba.backup: contiene un backup del último estado consistente de la base de datos. (A veces no se crea)

res: este sirve para abrir un catalogo ya existente en una ruta especifica dentro de nuestra aplicación, cuando abrimos la conexión se descomprime el catalogo, y al cerrar la aplicación se comprime de nuevo. Este tipo de catálogos no se usa con gran cantidad de datos, en ese caso es recomendable usar el catalogo **file**.

Acceder a los catálogos de Bases de Datos en HSQLDB.

Hay 2 maneras de acceder:

En proceso (IN-PROCESS):

Utilizando **JDBC** podemos conectarnos a la Base de Datos, después solo tenemos que indicar el directorio donde se encuentra el catálogo y ya podemos hacer nuestras consultas. Para ingresar a un catálogo de tipo file tenemos que obtener la conexión de esta manera:

```
Connection connection = DriverManager.getConnection("jdbc:hsqldb:file:prueba", "sa", "");
```

Ahí obtenemos la conexión, si estaba creado obtiene la conexión a ese catálogo y si no simplemente lo crea.
Si es un catálogo de tipo mem, la ruta es simplemente un "nombre", recuerda que este tipo de catálogos no crea archivos ni nada de eso, podemos poner lo que sea y comenzar hacer pruebas. Para conectarnos lo hacemos de esta manera:

```
Connection connection = DriverManager.getConnection("jdbc:hsqldb:mem:pruebamem", "sa", "");
```

Para conectarnos a un catálogo de tipo res, como es un recurso Java (JAR, ZIP...), tenemos que indicar la ruta del recurso como si se tratará de una clase, por ejemplo supongamos que nuestro catálogo se encuentra en esta ruta: "*datojava/hsqldb/prueba*", tenemos que conectarnos al catálogo de esta manera:

```
Connection connection =
DriverManager.getConnection("jdbc:hsqldb:res:datojava.hsqldb.prueba", "sa", "");
```

Acceder de esta manera (IN_PROCESS) a los catálogos es bastante rápido ya que no hay que estar pasando datos entre redes o algo por el estilo, pero tienes que tomar en cuenta que solo un proceso Java (una aplicación, un Database Manager...) puede acceder a un catálogo de tipo file, si quieres que varias aplicaciones accedan al mismo tiempo a este tipo de catálogos (IN-PROCESS), tenemos que hacer el catalogo file de solo lectura (read-only), o cambiar a un catálogo de tipo res que son de tipo read-only.

Modo Servidor:

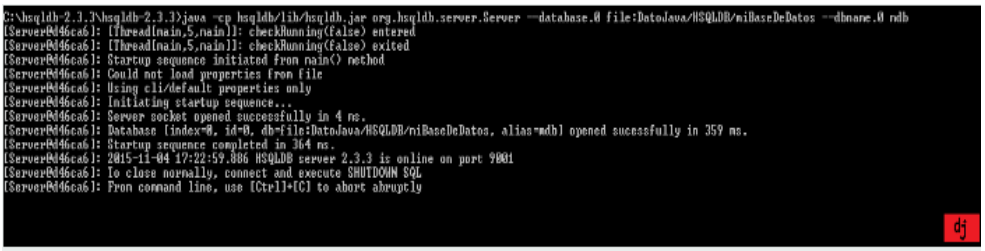
Es el que recomiendo, primero porque podemos conectarnos desde distintas aplicaciones Java al mismo catálogo, podemos ver y actualizar los datos que tengamos en el catálogo desde el Database Manager (viene incluido en el archivo que descomprimimos en el paso 1), mientras las demás aplicaciones están utilizando la base de datos, hay varias maneras de acceder en modo servidor, yo solo me voy a centrar en una manera de correr el servidor, se llama **HyperSQL HSQL Server** y es la más usada y la más rápida, para correr el servidor lo podemos hacer desde la línea de comandos o desde la misma aplicación. Para correr el servidor desde la línea de comandos tenemos que

ubicarnos en el directorio donde descomprimimos hsqldb-2.3.3, y ejecutar el siguiente comando:

java -cp hsqldb/lib/hsqldb.jar org.hsqldb.server.Server --database.0 file:DatoJava/HSQLDB/miBaseDeDatos --dbname.0 mdb

En ese comando estamos diciendo que vamos acceder o crear (en el caso de que no exista) ese catálogo llamado miBaseDeDatos en el directorio *"DatoJava/HSQLDB/"* con el alias mdb.

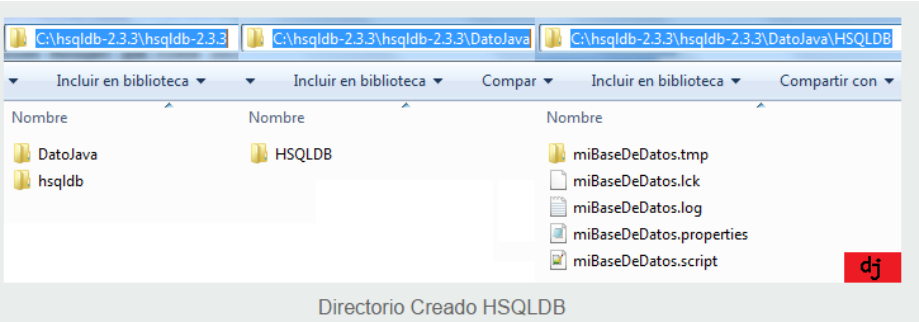
Al ejecutar ese comando deberíamos ver en la consola un mensaje como este:



Server Mode HSQLDB

Eso nos indica que ya podemos acceder al catálogo desde el **Database Manager** o bien desde cualquier aplicación Java mediante **JDBC**.

Para verificar que ese comando creo ese directorio con ese catálogo vamos a la ruta *"C:\hsqldb-2.3.3\hsqldb-2.3.3"*y deberíamos ver algo como esto:

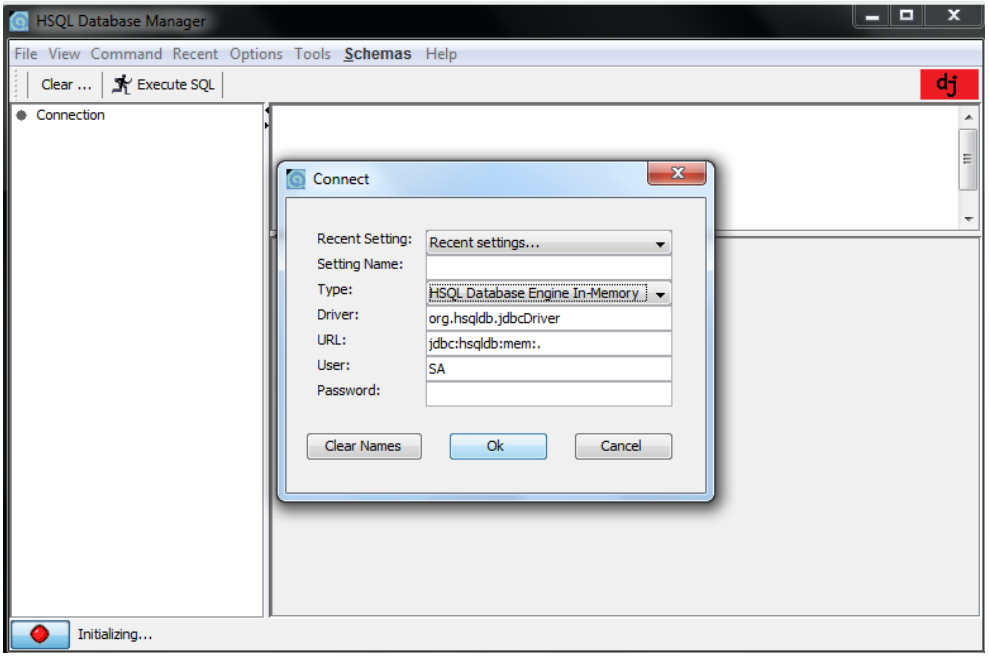


Directorio Creado HSQLDB

Ahora podemos ingresar con el **Database Manager** al mismo catalogo de esta manera, si queremos iniciar el Database Manager desde la consola, tenemos que ubicarnos en el directorio *"C:\hsqldb-2.3.3\hsqldb-2.3.3"* y ejecutar este comando en la consola:

java -cp hsqldb/lib/hsqldb.jar org.hsqldb.util.DatabaseManagerSwing

O podemos ir directamente a la ubicación del archivo batch del Database Manager (**runManagerSwing.bat**), que está en el directorio *C:\hsqldb-2.3.3\hsqldb-2.3.3\hsqldb\bin* y ejecutarlo. Las dos opciones inician el Manager.

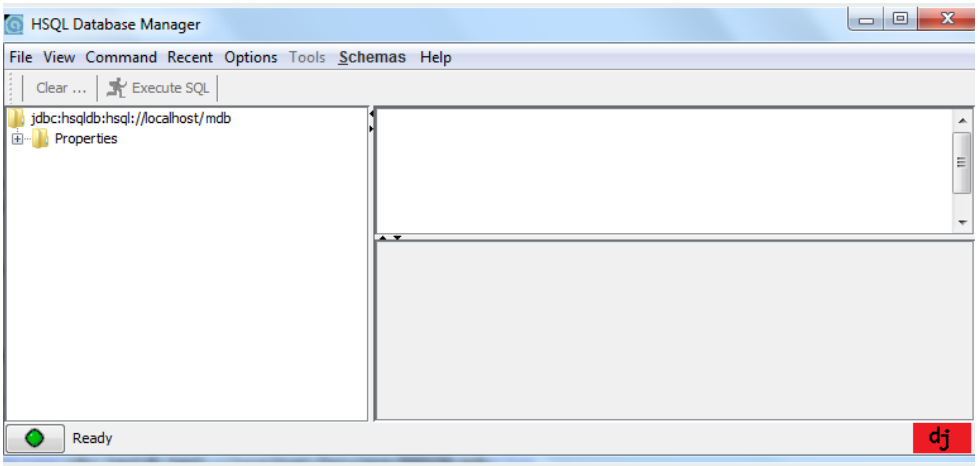


Database Manager HSQLDB

Para conectarnos al mismo catalogo que hemos creado anteriormente tenemos que configurar la conexión de esta manera:

- **Setting Name:** ingresamos el nombre "**DatoJava**"
- **Type:** HSQL Database Engine Server
- **Driver:** org.hsqldb.jdbcDriver
- **URL:** jdbc:hsqldb:hsql://localhost/mdb
- **User:** SA
- **Password:**

Ya con esa configuración podemos hacer consultas/actualizaciones en nuestro Catalogo:

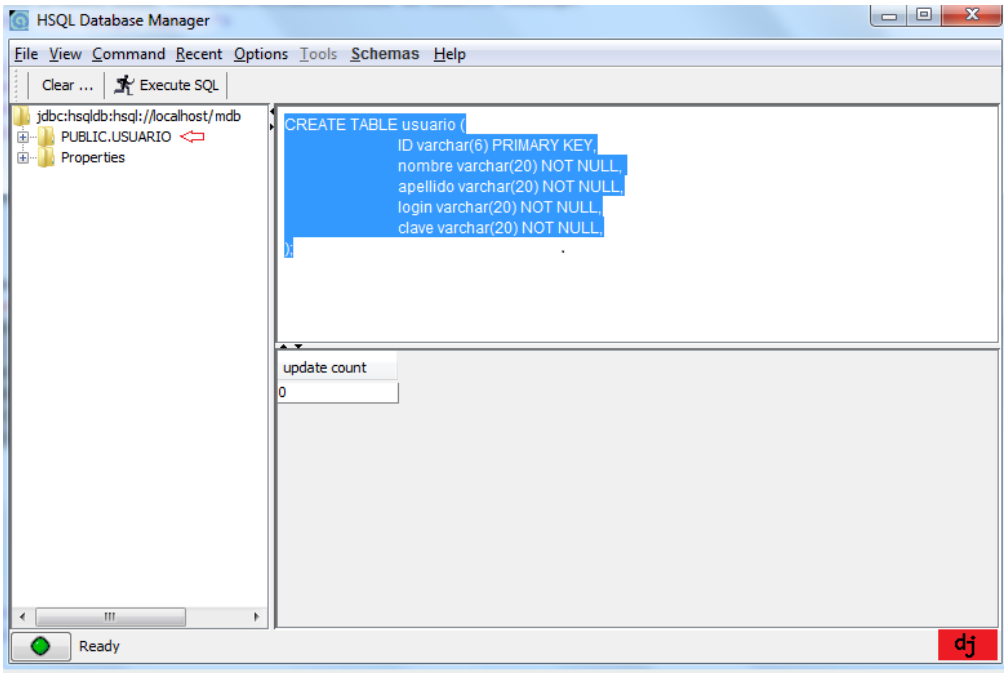


Database Manager HSQLDB

Por ejemplo si queremos crear una tabla nueva en nuestro catalogo "**miBaseDeDatos**", ejecutamos este script en el Database Manager:

```
CREATE TABLE usuario (
 ID varchar(6) PRIMARY KEY,
 nombre varchar(20) NOT NULL,
 apellido varchar(20) NOT NULL,
 login varchar(20) NOT NULL,
 clave varchar(20) NOT NULL,
);
```

Podemos ver en la imagen que ya fue creada la tabla.



Create Table HSQLDB

Para cerrar la Base De Datos tenemos que ejecutar **SHUTDOWN**, al ejecutar ese script todos los archivos se guardan de tal forma que la próxima vez que volvamos a conectarnos al catalogo, este abrirá rápidamente. La mejor opción de cerrar la Base De Datos es con **SHUTDOWN COMPACT**, este script sobrescribe el archivo *.data* que contiene información en CACHE acerca de las tablas y lo comprime al tamaño mínimo, se usa casi siempre y más aún cuando hacemos muchos inserts, updates, deletes, ya que ese archivo guarda esa información y el tamaño del archivo suele crecer rápidamente. Al ejecutarlo se cierra la conexión al catálogo:

Para correr el servidor desde una aplicación Java hay que crear una instancia del Servidor y asignarle las propiedades correspondientes. Con este código podemos iniciar el servidor desde nuestra aplicación:

```
1
2   //....
3   HsqlProperties hsqlProperties = new HsqlProperties();
4   hsqlProperties.setProperty("server.database.0",
5     "file:C:/hsqldb-2.3.3/hsqldb-2.3.3/DatoJava/HSQLDB/miBaseDeDatos");
6   hsqlProperties.setProperty("server.dbname.0", "mdb");
7
8   Server server = new Server();
9   server.setProperties(hsqlProperties);
10  server.setTrace(false); // No queremos loguear todo
11  server.start();
12
    //...
```

Como vemos nos estamos conectando al mismo catalogo que creamos anteriormente (*C:/hsqldb-2.3.3/hsqldb-2.3.3/DatoJava/HSQLDB/miBaseDeDatos*). Mientras no ejecutemos el script "**SHUTDOWN COMPACT**" la conexión queda abierta y cualquier aplicación se puede conectar, incluyendo el Database Manager.

2.3.3. Base de datos Firebird, Firebird**,** es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en 2000. Su código fue reescrito de C a C++. El proyecto se desarrolla activamente y el 18 de abril de 2008 fue liberada la versión 2.1.

Los objetivos de la Fundación FirebirdSQL son:

- Apoyar y lograr el avance del manejador de base de datos relacional Firebird.

- Proveer los mecanismos e infraestructura no comerciales para aceptar y administrar los fondos recaudados, e invertir tales fondos para promover el esfuerzo del desarrollo de esta base de datos.
- Fomentar la cooperación y la afiliación de individuos, organizaciones sin fines de lucro y compañías comerciales involucradas o que estén planeando estar involucradas en el desarrollo, apoyo y promoción de los proyectos de software de Firebird y sus productos y actividades asociadas.

Características:

- Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.
- Ejecutable pequeño, con requerimientos de hardware bajos. Arquitectura Cliente/Servidor sobre protocolo TCP/IP y otros (embedded).
- Soporte de transacciones ACID y claves foráneas. Es medianamente escalable.
- Buena seguridad basada en usuarios/roles. Diferentes arquitecturas, entre ellas el Firebird incrustado (embedded server) que permite ejecutar aplicaciones monousuario en ordenadores sin instalar el software Firebird.
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como cd-roms.

Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc.

- Requisitos de administración bajos, siendo considerada como una base de datos libre de mantenimiento, al margen de la realización de copias de seguridad.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL.
- Capacidad de almacenar elementos BLOB (Binary Large OBjects).

  Soporte de User-Defined Functions (UDFs). Versión autoejecutable, sin instalación, excelente para la creación de catálogos en CD-Rom y para crear versiones de evaluación de algunas aplicaciones.
- Tipos de servidor, Existen dos tipos de servidor Firebird para ser instalados: Classic y Super server. Si bien tienen varias diferencias menores entre sí, la principal consiste en que el super server maneja hilos de ejecución individuales para cada conexión. Por lo tanto para un número reducido de conexiones el recomendado sería el classic porque consumirá menor cantidad de recursos. En caso de arquitecturas SMP, se debe utilizar el servidor classic porque el Supersever no tiene soporte para este tipo de arquitectura.

- Los propios desarrolladores de Firebird recomiendan lo siguiente a la hora de decidirse por uno de estos servidores: En plataformas Windows seleccionar el Superserver. En Linux simplemente elegir cualquiera, según las conexiones estimadas. En la mayoría de las situaciones no se notará diferencias en la ejecución.
- Podría considerarse un tercer tipo, el Embedded. Éste consiste en una única biblioteca de enlace dinámico DLL (de unos 2 MB de tamaño) que contiene todo el servidor. De esta forma se puede tener un DBMS completo disponible y distribuíble junto con aplicaciones de usuario sin requerir que este se instale por separado.

http://www.firebirdmanual.com/firebird/es/firebird-manual/2/limites-de-firebird/36

Versiones de Instalación

Firebird viene en dos sabores, llamados *arquitecturas*: Classic Server y Superserver. ¿Cuál de ellos debería instalar? Bueno, eso depende de su situación. A continuación se ofrece un resumen de las diferencias más importantes.

TABLA Nº 01

CARACTERISTICAS DE LAS VERSIONES DE INSTALACION FIREBIRD

Classic Server	Superserver
Completamente maduro en Linux; todavía 'experimental' en cierta forma, en Windows.	Completamente maduro tanto en Windows como en Linux.
Crea un proceso por cada conexión cliente, cada uno con su propio caché. Utiliza menos recursos si la cantidad de conexiones es baja.	Proceso único con un hilo de ejecución (thread) separado para cada conexión. Se comparte el espacio de caché. Más eficiente si crece el número de conexiones simultáneas.
Permite E/S directa, rápida, a archivos de bases de datos para conexiones locales (sólo Linux).	Las conexiones locales deben hacerse con la forma de acceso remoto, conectando a `localhost`. En Windows se pueden hacer conexiones locales, pero no son tan veloces como las de la versión "Classic" en Linux, y también son menos seguras.
Windows: implementados parcialmente *Services Manager (Administrador de Servicios)*, tareas de soporte como backup/restore, database shutdown (sacar de línea la base de datos) etc. a través de la red. Otras tareas de servicio tienen que ser realizadas localmente usando las herramientas cliente (pequeños ejecutables independientes) que vienen con Firebird. Linux: Administrador de Servicios completo.	*Administrador de Servicios* completo (en Windows y Linux) que le permite realizar tareas de administración (backup/restore, database shutdown, manejo de usuarios, estadísticas, etc.) programáticamente. Se puede conectar al Administrador de Servicios a través de la red y por lo tanto realizar estas tareas en forma remota.
Soporte para SMP (multi-procesador). Mejor rendimiento en caso de un pequeño número de conexiones simultáneas que no se influencian entre sí.	No hay soporte para SMP. En máquinas multiprocesador con Windows, el rendimiento puede incluso caer dramáticamente cuando el SO cambia el proceso entre las CPUs. Para prevenir esto, fije el parámetro *`CpuAffinityMask`* en el archivo de configuración `firebird.conf`.

Como puede ver, ninguna de las arquitecturas es mejor en todos los aspectos. Esto no es una sorpresa: no estaríamos manteniendo dos arquitecturas si una de ellas fuera perdedora en todos los frentes.

- ✓ En Windows, elija Superserver.
- ✓ En Linux, elija cualquiera de los dos. En la mayoría de los casos, no notará una diferencia de rendimiento.

Note que Ud. puede cambiar en cualquier momento de una arquitectura a otra; sus aplicaciones y bases de datos seguirán funcionando (salvo que sus aplicaciones llamen a funciones no soportadas o no completadas del Administrador de Servicios en Classic).

Para Linux, los paquetes Superserver comienzan con FirebirdSS, los paquetes Classic con FirebirdCS.

Para Windows, hay un paquete de instalación combinado; se selecciona la arquitectura durante el proceso de instalación.

Ubicaciones en disco por defecto

La siguiente tabla describe las ubicaciones en disco por defecto para los componentes en Windows y Linux.

En la gran revisión del código base comenzada en v.1.5, fueron removidos los antiguos enlaces a 'artefactos' de Interbase, y muchos de los principales componentes fueron renombrados. Como resultado, Firebird 1.5 permite un servidor InterBase® corriendo al mismo tiempo, mientras los dos servidores no escuchen en el mismo puerto TCP/IP. Si Ud. necesita esta característica, vea *Configurando el puerto del servicio* en las

Notas de Versión que se incluyeron en su instalación de Firebird (busque en el subdirectorio doc).

TABLA Nº 02

CARACTERISTICA Y UBICACIÓN EN EL SISTEMA OPERATIVO WINDOWS

Plataforma	Componente	Nombre de archivo	Ubicación por defecto
Windows 32-bit y 64-bit (Windows 95, 98, ME, NT, 2000, XP, ...)	Directorio de instalación (referido de aquí en más como <InstallDir>)		C:\Archivos de programa\Firebird\Firebird_1_5
	Servidor Firebird	fbserver.exe (SS) o fb_inet_server.exe (CS)	<InstallDir>\bin
	Herramientas de línea de comandos	gbak.exe, gfix.exe, gstat.exe, etc.	<InstallDir>\bin
	Base de datos de ejemplo	employee.fdb	<InstallDir>\examples
	Librerías de funciones definidas por el usuario (UDF)	ib_udf.dll & fbudf.dll	<InstallDir>\UDF
	Cliente Firebird	fbclient.dll (con un archivo opcional gds32.dll, para las aplicaciones antiguas)	<InstallDir>\bin (con una copia opcional en el directorio de sistema de Windows - vea la nota debajo de la tabla)

TABLA Nº 03

CARACTERISTICA Y UBICACIÓN EN EL SISTEMA OPERATIVO LINUX - UNIX

Plataforma	Componente	Nombre de archivo	Ubicación por defecto
Linux y posiblemente otras distribuciones UNIX	Directorio de instalación (referido de aquí en más como <InstallDir>)		/opt/firebird
	Servidor Firebird	fbserver (SS) o fb_inet_server (CS)	<InstallDir>/bin
	Herramientas de línea de comandos	gbak, gfix, gstat, etc.	<InstallDir>/bin
	Base de datos de ejemplo	employee.fdb	<InstallDir>/examples
	Librerías UDF	ib_udf.so, fbudf.so	<InstallDir>/UDF
	Cliente Firebird	libfbclient.so.-1.5.n (binario); libfbclient.so.1, libfbclient.so (enlace simbólico) Los antiguos enlaces libgds* también se instalan.	/usr/lib (actualmente, los binarios reales están en <InstallDir>/lib, pero Ud. debería usar los enlaces en /usr/lib)

Instalación de Firebird

Discos de instalación

El servidor Firebird y cualquier base de datos que cree o a la que se conecte deben residir en un disco duro físicamente conectado a la máquina servidora. No puede ubicar componentes del servidor o una base de datos en un disco mapeado, un directorio compartido o un sistema de archivos de red.

Aunque es posible instalar Firebird por algún método del sistema de archivos tal como “desempaquetar” (untar) un archivo de instantánea (snapshot) o descomprimir un archivo estructurado .zip de Winzip se recomienda encarecidamente que utilice el paquete de distribución la primera vez que instale Firebird. El ejecutable de

instalación de Windows, el programa rpm (RedHat Package Manager) de Linux y el archivo .tar.gz oficial para otras plataformas Posix realizan algunas tareas esenciales de configuración. Si Ud. sigue las instrucciones correctamente, no debería quedar nada por hacer después de completado el proceso, ¡sólo conectarse y comenzar!.

El instalador de Firebird le permite escoger entre instalar la versión Superserver o la versión Classic Server. Como se mencionó antes, Ud. debería elegir Superserver a menos que conozca las diferencias y tenga razones para preferir Classic.

Si instala Firebird bajo Windows 95/98/ME, desmarque la opción de instalar el applet del Panel de Control. No funciona en estas plataformas. Más adelante en esta guía le daremos un enlace a un applet usable.

En plataformas de servidor Windows NT, 2000 and XP el servicio Firebird estará corriendo cuando la instalación se complete. La próxima vez que reinicie su servidor, el servicio se iniciará automáticamente.

Probando su instalación

Si todo funciona como fue diseñado, el proceso del servidor Firebird estará corriendo en su servidor después de completada la instalación. Arrancará automáticamente cada vez que reinicie su servidor.

En este punto, se asume que utilizará el protocolo TCP/IP recomendado para su red cliente/servidor Firebird.

Nota

Por información sobre cómo utilizar el protocolo NetBEUI en un entorno completamente Windows, refiérase al capítulo 6, *Configuración de red* en el manual *Using Firebird*

Aviso

Las redes IPX/SPX no son soportadas por Firebird.

Haciendo Ping al servidor

Usualmente, lo primero que querrá hacer una vez que la instalación está completa es hacer ping al servidor.

Esto le dará una comprobación de que su máquina cliente puede ver el equipo servidor de su red. Por ejemplo, si la dirección IP en el dominio en que es visible para su cliente es 192.13.14.1, abra una ventana del shell y escriba el comando:

ping 192.13.14.1

Substituya esta dirección IP por la dirección IP en que su servidor transmite.

Note que si se está conectando al servidor desde un cliente local - esto es, un cliente corriendo en la misma máquina que el servidor- puede hacer ping al servidor virtual de loopback TCP/IP:

ping localhost –o bien– **ping 127.0.0.1**

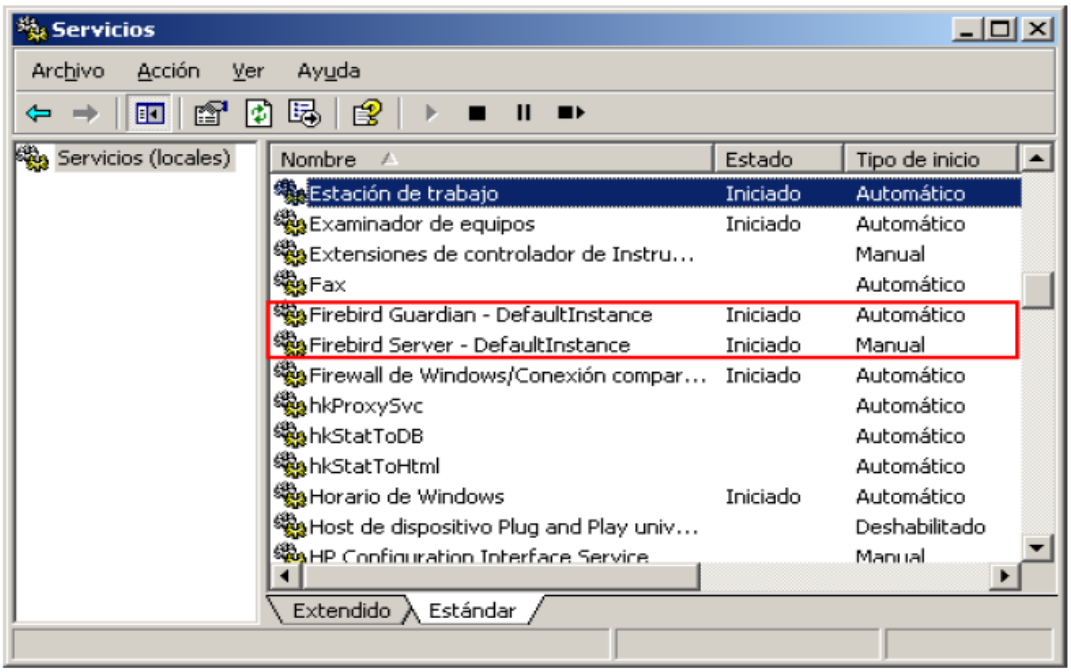
Comprobar que el servidor Firebird está ejecutándose

Abra el Panel de Control -> Servicios (NT) o Panel de Control ->Herramientas Administrativas-> Servicios (2000, XP).

La ilustración muestra el applet de Servicios en Windows 2000. La apariencia puede cambiar de una edición de Windows a otra.

FIGURA Nº 01

COMPROBACION DEL ESTADO DEL SERVICIO FIREBIRD



Si el guardián está corriendo (como se muestra en la imagen, arriba) puede tener un nombre de servicio diferente debido a cambios de versión.

Applets del panel de control para Windows

Desde la versión 1.03, se incluye un applet para el panel de control en la distribución de Firebird. Aunque el applet no es esencial, provee una forma conveniente para arrancar y detener el servidor. Desafortunadamente, el applet incluido en el kit solamente funciona en Windows NT, 2000 y XP. En Windows 9x y ME, si Ud. desea un práctico applet como este, visite este sitio: http://www.achim-kalwa.de/fbcc.phtml y descargue el Centro de Control de Firebird (Firebird Control Center) fbcc-0.2.6.exe.

FIGURA Nº 02

CONTROL DEL SERVIDOR FIREBIRD

Una dirección de red para el servidor

- ✓ Si Ud. está en una red administrada, obtenga la dirección IP del servidor de su administrador de sistemas.
- ✓ Si tiene una red simple de dos máquinas enlazadas por un cable cruzado, puede configurar su servidor con cualquier

dirección IP que desee excepto 127.0.0.1 (que está reservada para un servidor de realimentación local) y, por supuesto, la dirección IP que está usando para su máquina cliente.

- ✓ Si conoce la dirección IP “nativa” de sus tarjetas de red, y son diferentes, puede simplemente usar ésas.
- ✓ Si su intención es probar una instalación en una única máquina como cliente y servidor, debería usar la dirección de realimentación local - localhost, con la dirección IP 127.0.0.1

Nombre de usuario y clave por defecto

El usuario SYSDBA posee todos los privilegios sobre el servidor. Dependiendo de la versión, SO, y arquitectura, el programa de instalación:

- ✓ instalará el usuario SYSDBA con la clave masterkey (actualmente, masterke: se ignoran los caracteres más allá del octavo), o • le pedirá que ingrese una clave durante la instalación, o
- ✓ generará una clave aleatoria para el usuario SYSDBA y la almacenará en el archivo /opt/firebird/SYSDBA.password

Si su servidor está expuesto *aunque sea mínimamente* a la Internet y la clave es masterkey, debería cambiarla inmediatamente usando la utilidad de línea de comando gsec

Como cambiar la clave de SYSDBA

Importante

Con algunas instalaciones de Firebird, Ud. sólo puede ejecutar gsec si está autenticado en el sistema operativo como Superusuario (root en Linux) o como el usuario bajo el cual corre el proceso del servidor Firebird.

Supongamos que decide cambiar la clave de SYSDBA a icuryy4me.

1. Abra una ventana de terminal en su servidor y posiciónese en el directorio donde se localizan las utilidades de línea de comandos. Refiérase a Tabla de componentes de una instalación Firebird para encontrar esta ubicación.
2. Escriba lo siguiente (sensible a mayúsculas en todas las plataformas excepto Windows):

   gsec -user sysdba -password masterkey

3. Debería ver el prompt de la utilidad gsec : GSEC>
4. Escriba este comando: **modify sysdba -pw icuryy4me**
5. Presione **Enter**. La nueva clave icuryy4me está ahora encriptada y almacenada y masterkey ya no es válida.
6. Ahora salga de la terminal de gsec:

   quit

Una herramienta de administración

La distribución de Firebird no incluye una herramienta de administración visual. Contiene un conjunto de herramientas de línea de comandos, programas ejecutables que se localizan en el subdirectorio bin de su instalación de Firebird.

Las herramientas visuales disponibles para usar en un equipo cliente con Windows son demasiado numerosas para describirlas aquí. Existen también, en distinto estado de construcción, unas pocas herramientas escritas en Kylix de Borland para usar en máquinas clientes Linux.

Inspeccione la página Downloads > Contributed > Admin Tools page (Descargas de contribuidores; página de herramientas de administración) en http://www.ibphoenix.com para ver todas las opciones.

Seguridad

Firebird 1.5 tiene unas cuantas provisiones nuevas en la parte de seguridad. ¡Explórelas! Muchas de las características configurables toman como valor por defecto el antiguo, comportamiento “inseguro” para no interferir con las aplicaciones existentes, pero se puede mejorar significativamente la seguridad de su sistema si eleva el nivel de protección dondequiera que sea posible. Entre las nuevas y/o diferentes características de seguridad se cuentan:

- ✓ En los sistemas Posix, Firebird ahora se ejecuta como usuario firebird por defecto, no como root.

- ✓ En las plataformas Windows, también se puede ejecutar el servicio de Firebird bajo una cuenta de usuario designada (por ej. Firebird). La práctica actual -ejecutar el servicio como el usuario LocalSystem- implica un riesgo de seguridad si su sistema está conectado a la Internet. Consulte README.instsvc en el subdirectorio doc para aprender sobre esta configuración.
- ✓ *Alias de Bases de Datos* esconden al cliente la ubicación física de las bases de datos. Usando alias, un cliente puede por ejemplo conectar a “frodo:zappa” sin necesidad de saber que la ubicación real es:

  frodo:/var/firebird/music/underground/mothers_of_invention.fdb
- ✓ Los alias también le permiten reubicar bases de datos manteniendo la misma cadena de conexión en los clientes.
- ✓ El parámetro *DatabaseAccess* puede tomar el valor Restrict para limitar el acceso a directorios explícitos del sistema de archivos, o incluso None para permitir el acceso a bases de datos sólo a través de alias. El valor por defecto es All, esto es, sin restricciones.
- ✓ El parámetro *ExternalFileAccess* permite controlar el acceso a tablas externas.
- ✓ El parámetro *UdfAccess* indica las ubicaciones permitidas para librerías de funciones definidas por el usuario.
- ✓ Los alias de bases de datos residen en el archivo aliases.conf, los parámetros de configuración en

- ✓ firebird.conf. Por favor consulte las notas de versión de su distribución de Firebird para ver su uso exacto (y algún otro buen consejo).

Conectar a la base de datos de ejemplo

En el subdirectorio examples de su instalación de Firebird hay una base de datos de ejemplo llamada employee.fdb. Puede usar esta base de datos para “probar sus alas”.

Nombre del servidor y ruta de acceso

Si Ud. cambia de lugar la base de datos de ejemplo, asegúrese de moverla a un disco duro que esté físicamente unido a su equipo servidor. Los discos compartidos, mapeados o (en Unix) sistemas SMB (Samba) no funcionarán. La misma regla se aplica a cualquier base de datos que Ud. cree.

Hay dos elementos en una cadena de conexión: el nombre del servidor y la ruta de acceso al archivo. El formato es como sigue:

- ✓ Para un servidor Linux:

  servidor:/ruta_al_archivo/archivo_de_la_base_de_datos

  Ejemplo en un servidor Linux o algún otro Posix llamado serverxyz:

  serverxyz:/opt/interbase/examples/employee.fdb

- ✓ Para un servidor Windows:

  servidor:letra_de_disco:\ruta\archivo_de_base_de_datos

  Ejemplo en Windows:

  castillo:D:\Cultura\BasesDeDatos\MúsicaNo.fdb

La sentencia CONNECT

Conectar a una base de datos Firebird siempre requiere que el usuario “se identifique” usando un nombre de usuario y clave válidos -operación normalmente denominada log-in o login. Cualquier usuario aparte de SYSDBA, root (en los sistemas Posix), o Administrador (en los sistemas Windows, si Firebird se ejecuta con ese usuario) necesita también tener permisos a los objetos dentro de una base de datos. Por simplicidad, veremos aquí la autenticación como SYSDBA usando la clave masterkey.

Usando isql

Hay varias formas diferentes para conectar con una base de datos usando isql. Una forma es ejecutar isql en su terminal interactiva. Diríjase al subdirectorio bin de su instalación y en el prompt tipee el comando **isql** (nota: **#** significa “presione **Enter** ”):

C:\Archivos de programa\Firebird\Firebird_1_5\bin>isql#

Use CONNECT or CREATE DATABASE to specify a database

SQL>CONNECT "C:\Archivos de programa\Firebird\Firebird_1_5\examples\employee.fdb"#

CON>user 'SYSDBA' password 'masterkey';#

En este punto, isql le informará que Ud se ha conectado:

DATABASE "C:\Archivos de programa\Firebird\Firebird_1_5\examples\

employee.fdb", User: sysdba

SQL>

Ahora puede seguir jugando con la base de datos employee.fdb. Los caracteres isql significan *interactive SQL [utility]*. Puede usarlo para consultar datos, obtener información acerca de los metadatos, crear objetos de base de datos, ejecutar scripts de definición de datos y mucho más. Para volver a la línea de comandos escriba

SQL>QUIT;#

Para más información sobre isql, vea *Using Firebird*, capítulo 10: *Interactive SQL Utility (isql)*.

Crear una base de datos usando isql

Hay más de una manera de crear una base de datos usando isql. Aquí veremos una sola forma simple de crear una base de datos en forma interactiva aunque, para el trabajo serio de definición de bases de datos, Ud. debería crear y mantener los metadatos de sus objetos usando scripts de definición de datos.

Arrancar isql

Para crear una base de datos en forma interactiva usando la interfaz de comandos de isql, se debe trabajar en el servidor. Posicione una terminal de comandos en el subdirectorio bin y arranque isql como sigue:

C:\Archivos de programa\Firebird\Firebird_1_5\bin>isql#

Use CONNECT or CREATE DATABASE to specify a database

La sentencia CREATE DATABASE

Ahora puede crear su nueva base de datos interactivamente. Supongamos que desea crear una base de datos llamada test.fdb y almacenarla en un directorio llamado data en su disco D:

SQL>CREATE DATABASE 'D:\data\test.fdb' page_size 8192#

CON>user 'SYSDBA' password 'masterkey';#

La base de datos será creada y, luego de unos breves instantes, el prompt SQL volverá a aparecer. Ahora Ud. está conectado a la nueva base de datos y puede proceder a crear algunos objetos de prueba en ella.

Para verificar que realmente hay una base de datos ahí, escriba esta consulta:

SQL>SELECT * FROM RDB$RELATIONS;#

¡La pantalla se llenará con una gran cantidad de datos! Esta consulta selecciona todas las filas de la tabla de sistema adonde Firebird almacena los metadatos para las tablas. Una base de datos “vacía” no está vacía -contiene una base de datos que será completada con metadatos a medida que Ud. cree objetos en ella.

Para volver a la línea de comandos tipee

SQL>QUIT;#

Para más información acerca de isql, vea *Using Firebird*, capítulo 10: *Interactive SQL Utility (isql)*.

El símbolo delimitador de cadenas

Las cadenas de caracteres en Firebird están delimitadas por un par de comillas simples -'I am a string'– (código ASCII 39, *no* 96). Si Ud. ha usado versiones previas del pariente de Firebird, Interbase ®, recordará que las comillas simples y dobles se podían intercambiar como delimitadores de cadenas. En Firebird, las comillas dobles no se pueden usar como delimitadores de cadenas.

Identificadores con comillas dobles

Antes del estándar SQL-92, no era legal tener nombres de objetos (identificadores) en una base de datos que fueran iguales a las palabras claves del lenguaje, distinguieran mayúsculas de minúsculas, o contuvieran espacios. SQL-92 introdujo un nuevo estándar para hacer legales todas esas cosas, siempre que los identificadores fueran rodeados por un par de símbolos de comilla doble (ASCII 34) y fueran siempre referidos delimitados por comillas dobles.

El propósito de este “regalo” era hacer más fácil migrar metadatos desde RDBMSs no estándares. La parte mala es que, si Ud. elige encerrar un identificador con comillas dobles, distinguirá entre mayúsculas y minúsculas y será obligatorio siempre escribirlo entre comillas dobles.

Firebird permite una ligera relajación de esta regla si se cumple un conjunto de condiciones muy especial: si el identificador que fue definido entre comillas dobles:

1. fue definido totalmente en mayúsculas,

2. no es una palabra clave, y

3. no contiene espacios,

Entonces puede ser usado en SQL sin comillas y sin prestar atención a mayúsculas y minúsculas (¡pero en cuanto le pone comillas alrededor, deben coincidir las mayúsculas nuevamente!)

A menos que tenga una razón de peso para definir identificadores con comillas, se recomienda que los evite. Firebird acepta sin problemas una mezcla de identificadores con y sin comillas -por lo que no es problema incluir esa palabra clave que Ud. obtuvo de una base de datos antigua, si realmente lo necesita.

Apóstrofos en cadenas

Si Ud. necesita usar un apóstrofo dentro de una cadena de Firebird, puede “escapar” el carácter del apóstrofo precediéndolo con otro. Por ejemplo, esta cadena producirá un error:

'Joe's Emporium' porque el evaluador encuentra el apóstrofo e interpreta la cadena como 'Joe' seguida por algunas palabras claves desconocidas.

Para convertir el ejemplo en una cadena legal, duplique el caracter apóstrofo:

'Joe''s Emporium'

Note que son DOS comillas simples, no una doble.

Concatenación de cadenas

El símbolo de concatenación en SQL es un doble “pipe” (ASCII 124, un par sin espacio entremedio).

En SQL, el símbolo “+” es un operador aritmético y provocará un error si intenta usarlo para concatenar cadenas. La siguiente expresión agrega el siguiente texto “ Reportado por: ” delante de cada apellido:

'Reportado por: ' || LastName

Tenga cuidado con las concatenaciones. Tenga en cuenta que Firebird generará un error si su expresión intenta concatenar dos o más columnas de tipo char o varchar si la longitud combinada puede exceder el límite máximo de longitud para el tipo char o varchar (32 KB).

Vea también en las notas más abajo, Expresiones con NULL, sobre la concatenación de expresiones que involucran NULL.

El Proyecto Firebird

Los desarrolladores, diseñadores y testers que le han dado Firebird y varios de los manejadores son miembros del proyecto de código abierto Firebird en SourceForge, esa fantástica comunidad virtual que es el hogar para miles de equipos de desarrollo de software de código abierto. La dirección del proyecto Firebird aquí es http://sourceforge.net/projects/firebird. En este sitio se encuentra el árbol de código fuente, el seguimiento de errores (bug tracker) y una cantidad de archivos técnicos que pueden ser descargados por

varias razones relacionadas con el desarrollo y prueba de las bases de código.

Los desarrolladores y testers usan un foro de lista de correo – firebird-devel@lists.sourceforge.net – como su “laboratorio virtual” para comunicarse unos con otros acerca de su trabajo en mejoras, corrección de errores y producción de nuevas versiones de Firebird.

Cualquiera que esté interesado en observar el progreso puede unirse a este foro. No obstante, las preguntas de soporte de usuarios son una distracción que no es bienvenida. Por favor no intente enviar sus preguntas de soporte allí! éstas pertenecen al grupo firebird-support@yahoogroups.com.

CAPÍTULO III: RECOPILACION DE INFORMACION

3.1. Recopilación de Datos

Para el manejo de la recopilación de datos se utilizó un formato en una hoja de Excel para recoger toda la información de las pruebas para cada una de las características funcionales de las bases de datos seleccionadas para el estudio; los datos de prueba fueron extraídos de la información de la base de datos de Oracle Express (que también es de uso libre).. Se realizó las pruebas en cada una de las Bases de Datos en estudio y se obtuvieron los resultados que se presentan en las siguientes secciones:

3.2. Pruebas de tiempo de respuesta de manejo de datos.

Pruebas de tiempo para INSERT (Ingresar datos a la tabla)

1. Insertar un registro con todos los datos de los campos de la tabla:
2. Insertar solo algunos datos de prueba: INSERT INTO `data`.`datos` (`Id-Emp`, `nombre`, `apellidos`, `email`, `telefono`, `f_nac`, `sueldo`) VALUES ('302', 'JULIO', 'DONAYRE RAMOS', 'jramos@gmail.com', '534.215.15121', '1985-07-15', '25000')
3. Insertar los datos de id-Emp, nombre, apellidos, email, sueldo
4. Insertar los datos: código empleado, nombre y sus apellidos
5. Insertar los datos: código empleado, nombre y sus apellidos, y email

TABLA Nº 06

RESULTADOS DE LAS PRUEBAS PARA INSERT

Nº	Acción	SQLite	HiperSQL	Firebird
1	Insert into datos (todos los campos)	0,041	0,042	0,041
2	INSERT INTO `data`.`datos` (`Id-Emp`, `nombre`, `apellidos`, `email`, `telefono`, `f_nac`, `sueldo`) VALUES ('302', 'JULIO', 'CASTRO MARTINEZ', 'cmartinez@gmail.com', '534.215.15121', '1985-07-15', '25000')	0,039	0,039	0,037
3	Insertar los datos de id-Emp, nombre, apellidos, email, sueldo	0,043	0,044	0,042
4	Insertar los datos: codigo empleado, nombre y sus apellidos	0,035	0,034	0,035
5	Insertar los datos: código empleado, nombre y sus apellidos, y email	0,038	0,038	0,037

Fuente: Elaboración investigadores

Pruebas de tiempo para UPDATE (Actualizar o modificar)

1. Actualizar el correo y el sueldo de una empleado cuyo código de empleado es 085: UPDATE `data`.`datos` SET `email` = 'JMURMAN@HOTMAIL.COM', 'sueldo'=53000 WHERE CONVERT('datos'.'id_emp')='112'
2. Modificar todos los códigos de departamento que tengan el valor de 20, con el valor de 112:
3. Actualizar el sueldo a 30000 a todos los empleados que pertenecen al departamento cuyo código es 100.
4. Actualizar los trabajos IT_PROG (programador de tecnologías) por Tecnología
5. Cambiar el nombre a carlos al empleado de código 206

TABLA Nº 07

RESULTADOS DE LAS PRUEBAS PARA UPDATE (Actualización)

Nº	Acción	SQLite	HiperSQL	Firebird
1	UPDATE `data`.`datos` SET `email` = 'PDIAZ@HOTMAIL.COM', 'sueldo'=8700 WHERE CONVERT('datos'.'id_emp')='112'	0,041	0,042	0,041
2	Actualizar todos los Depart_id 20, con 112	0,0025	0,0025	0,0024
3	Actualizar el sueldo a 30000 todos los Depart_id=112	0,0032	0,0032	0,0032
4	Actualizar los trabajos IT_PROG por Tecnología	0,0617	0,0616	0,0615
5	Cambiar el nombre a Jorgito al empleado de código 206	0,0664	0,0064	0,0064

Fuente: Elaboración grupo de investigación

Pruebas de tiempo para DELETE (Eliminación de datos)

1. Eliminación los datos del empleado cuyo código de empleado es el 120: DELETE FROM DATOS WHERE 'ID-EMP'='085'
2. Eliminar todos los datos de los empleados cuyo trabajo es IT_PROG: DELETE FROM DATOS WHERE 'ID_TRAB'='IT_PROG'
3. Eliminar todos los datos cuya comisión es 0,15: DELETE FROM DATOS WHERE 'COMISION'=0,18
4. Eliminar todos los datos de los empleados cuyo sueldo es igual 80000: DELETE FROM DATOS WHERE 'SUELDO'=80000
5. Eliminar todos los empleados cuyo código de departamento es 20: DELETE FROM DATOS WHERE 'DEPART-ID'='20'

TABLA Nº 08

RESULTADOS DE LAS PRUEBAS PARA DELETE (Eliminación)

Nº	Acción	SQLite	HiperSQL	Firebird
1	DELETE FROM DATOS WHERE 'ID-EMP'='180'	0,0013	0,0013	0,0013
2	DELETE FROM DATOS WHERE 'ID_TRAB'='IT_PROG'	0,0013	0,0013	0,0013
3	DELETE FROM DATOS WHERE 'COMISION'=0,18	0,0014	0,0014	0,0012
4	DELETE FROM DATOS WHERE 'SUELDO'=50000	0,0013	0,0014	0,0013
5	DELETE FROM DATOS WHERE 'DEPART-ID'='20'	0,0013	0,0013	0,0013

Fuente: Elaboración grupo de investigación

3.1 Pruebas de tiempo para respuesta a consultas.

2. Consultar todos los datos de los empleados: Select * from datos
3. Consultar todos los datos de los empleados cuyo sueldo es menor que 12000: aSelect * from datos where sueldo<12000
4. Consultar todos los datos de los empleados cuyo código de departamento es 50: Select * from datos where depart_id='50'
5. Consultar todos los datos de los empleados cuyo código de trabajo es IT_PROG: Select * from datos where id_trabajo='IT_PROG'
6. Consultar todos los datos del empleado cuyo nombre es Carlos: SELECT * FROM `datos` WHERE `nombre`='Carlos'

TABLA Nº 09

RESULTADOS DE LAS PRUEBAS PARA SELECT (Consultas)

Nº	Acción	MySql	HiperSQL	PostgrSql
1	Select * from datos	0,0021	0,0020	0,0021
2	Select * from datos where sueldo<12000	0,0018	0,0018	0,0017
3	Select * from datos where depart_id='50'	0,0022	0,0022	0,0021
4	Select * from datos where id_trabajo='IT_PROG'	0,0022	0,0022	0,0022
5	SELECT * FROM `datos` WHERE `nombre`='carlos'	0,0022	0,0021	0,0022

Fuente: Elaboración grupo de investigación

CAPÍTULO IV: ANALISIS E INTERPRETACION DE LOS RESULTADOS

4.1 Población y muestra:

4.1.1 Población.

Teniendo en cuenta el objetivo Determinar las diferencias del de los sistemas de BD libres SQLite, HiperSQL y Firebird, se considera como población a los principales sistemas de bases de datos libres en la web.

4.1.2 Muestra.

El tipo de muestra fue probabilística eligiendo 5 formas de medir el impacto funcional en cada uno de los sistemas de BD.

4.2 Nivel de confianza y grado de significancia:

El nivel de confianza aplicada en probar las hipótesis está diseñado de la siguiente manera:

Nivel de confianza: 95%.

Significancia: 5%

4.3 Tamaño de la muestra representativa:

Son 5 proceso elegidos intencionalmente, para poder establecer el análisis posterior.

4.4 Análisis e interpretación de resultados:

TABLA N° 10

Estadística descriptiva para la primera prueba funcional INSERT en las tres bases de datos

Insert					
SQLite		HiperSQL		Firebird	
Media	0.0392	Media	0.0394	Media	0.0384
Desviación Estándar	0.0030332	Desviación Estándar	0.003847	Desviación Estándar	0.002966
Rango	0.008	Rango	0.01	Rango	0.007
Mínimo	0.035	Mínimo	0.034	Mínimo	0.035
Máximo	0.043	Máximo	0.044	Máximo	0.042

En la prueba funcional de INSERT, el promedio para SQLite es de 0.0392, para HiperSQL es de 0.0394, mientras que para Firebird obtiene un valor de 0.0384

Para esta prueba el valor más alto y más bajo para SQLite es de 0.043 y 0.035, para HiperSQL es de 0.044 y 0.034 y para Firebird asciende a 0.043 y 0.035.

Mientras que el rango para SQLite 0.008, con HiperSQL 0.01 y con Firebird 0.007.

Además se puede observar que con respecto al promedio, los datos muestran una desviación estándar para SQLite 0.00303315, HiperSQL 0.00384708 y Firebird 0.00296648.

TABLA N° 11

Estadística descriptiva para la primera prueba funcional DELETE en las tres bases de datos

Delete					
SQLite		HiperSQL		Firebird	
Media	0.00132	Media	0.00134	Media	0.00128
Desviación Estándar	4,47E-02	Desviación Estándar	5.48E-05	Desviación Estándar	4.47E-05
Rango	0.0001	Rango	0.0001	Rango	0.0001
Mínimo	0.0013	Mínimo	0.0013	Mínimo	0.0012
Máximo	0.0014	Máximo	0.0014	Máximo	0.0013

En la prueba funcional de DELETE, el promedio para SQLite es de 0.00132 para HiperSQL es de 0.00134, mientras que para Firebird obtiene un valor de 0.00128.

Para esta prueba el valor más alto y más bajo para SQLite es de 0.0014 y 0.0013, para HiperSQL es de 0.0014 y 0.0013 y para Firebird asciende a 0.0013 y 0.0012.

Mientras que el rango para las tres bases de datos es de 0.0001

Además se puede observar que con respecto al promedio, los datos muestran una desviación estándar para SQLite 0.00004472, HiperSQL 0.00005477 y Firebird con 0.00004472.

TABLA N° 12

Estadística descriptiva para la primera prueba funcional UPDATE en las tres bases de datos

Update					
SQLite		HiperSQL		Firebird	
Media	0.03496	Media	0.02314	Media	0.0229
Desviación Estándar	0.0308315	Desviación Estándar	0.027105	Desviación Estándar	0.026917
Rango	0.0639	Rango	0.0591	Rango	0.0591
Mínimo	0.0025	Mínimo	0.0025	Mínimo	0.0024
Máximo	0.0664	Máximo	0.0616	Máximo	0.0615

En la prueba funcional de UPDATE, el promedio para SQLite es de 0.003496 para HiperSQL es de 0.02314, mientras que para Firebird obtiene un valor de 0.0229.

Para esta prueba el valor más alto y más bajo para SQLite es de 0.0664 y 0.0025, para HiperSQL es de 0.0616 y 0.0025 y para Firebird asciende a 0.0615 y 0.0024.

Mientras que el rango para SQLite es de 0.0639, para HiperSQL es de 0.02710 y para Firebird 0.02691728

Además se puede observar que con respecto al promedio, los datos muestran una desviación estándar para SQLite 0.030831526, HiperSQL 0.02710494 y Firebird 0.02691728.

TABLA N° 13

Estadística descriptiva para la primera prueba funcional SELECT en las tres bases de datos

Select					
SQLite		HiperSQL		Firebird	
Media	0.0021	Media	0.00206	Media	0.00206
Desviacion Estándar	0.0001732	Desviacion Estándar	0.000167	Desviacion Estándar	0.000207
Rango	0.0004	Rango	0.0004	Rango	0.0005
Minimo	0.0018	Minimo	0.0018	Minimo	0.0017
Maximo	0.0022	Maximo	0.0022	Maximo	0.0022

En la prueba funcional de SELECT, el promedio para SQLite es de 0.0021 para HiperSQL es de 0.00206, mientras que para Firebird obtiene un valor de 0.00206.

Para esta prueba el valor más alto y más bajo para SQLite es de 0.0022 y 0.0018, para HiperSQL es de 0.0022 y 0.0018 y para Firebird asciende a 0.002 y 0.0017.

Mientras que el rango para SQLite es de 0.0004, para HiperSQL es de 0.0004 y para Firebird 0.0005.

Además se puede observar que con respecto al promedio, los datos muestran una desviación estándar para SQLite 0.000173205, HiperSQL 0.00016733 y Firebird 0.00020736

4.5 Prueba de Hipótesis:

4.5.1 Hipótesis de investigación.

Hi=No existen diferencias de Impacto de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.5.2 Hipótesis nula.

H_0=Existen diferencias significativa de impacto de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.5.3 Hipótesis estadística.

H_o: $\mu_1 = \mu_2 = \mu_3$

H_1: No todas las medias son iguales.

4.6 Prueba estadística utilizada:

Para contabilizar el tipo de investigación y el diseño seleccionado se ha utilizado como método de prueba estadística la hipótesis, a través del análisis de varianza (ANOVA) para comparar las medias de más de dos grupos, el procedimiento usado se denomina ANOVA de una vía. Este procedimiento es una extensión de una prueba t para la diferencia entre dos medias. Aunque ANOVA es un acrónimo para Análisis Of Variance, tendrá como objetivo analizar las diferencias entre las medias en su impacto funcional los tres sistemas de BD libres, SQLite, HiperSQL y Firebird.

La fórmula asociada es la siguiente:

$$CMTR = \frac{SCTR}{c - 1}$$

$$CME = \frac{SCE}{n - c}$$

Donde: **de la prueba de hipótesis**

Nivel de significación:5%

Nivel de confianza: 95%

Se encuentra el valor crítico de F con 2 gl1 numerador (3 -1 sistemas de base de datos), y 10 grados de libertad en el denominador (15-2) datos, en un nivel de significancia de 0.05 nos da el nivel crítico de 3.89

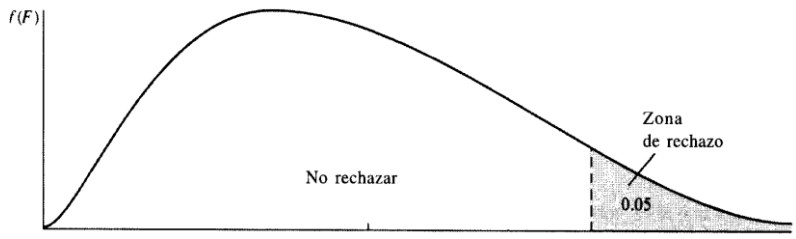
TABLA Nº 14

GRADOS DE LIBERTAD PARA LAS PRUEBAS ANOVA

	$F_{0.95}$; $\alpha = 0.05$								
Grados de libertad del denominador	Grados de libertad del numerador								
	1	2	3	4	5	6	7	8	9
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80

FIGURA Nº 03

CURVA DE ACEPTACION O RECHAZO PARA LA PRUEBA ANOVA



CRITERIOS DE DECISIÒN

Si el estadístico de prueba calculado Fcal es menor que F crítico, la hipótesis nula estadística no se rechaza, se acepta, se concluye que no existe una diferencia significativa en la media del impacto funcional en las tres bases de datos libres.

Si el valor p es mayor que el especificado de 0.05, no se rechaza hipótesis nula.

4.6.1 Prueba de Hipótesis Especifica Nº 1 INSERT

4.6.1.1 Hipótesis alterna.

Hi=No existen diferencias de Impacto funcional INSERT de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.1.2 Hipótesis Nula

H_0=Existen diferencias significativa de impacto funcional INSERT de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.1.3 Hipótesis Estadísticas

H_o: $\mu_1 = \mu_2 = \mu_3$

H_1: No todas las medias son iguales.

Grado de significancia: $\alpha = 0,05$

Estadígrafo de Prueba: La Prueba Anova de un solo factor o una sola vía.

Se recurre al programa del Excel 2007

TABLA Nº 15

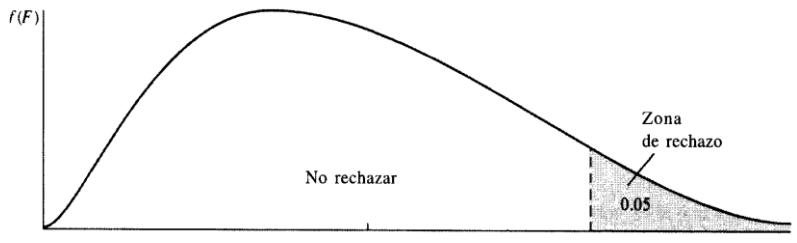
RESULTADOS PRUEBA ESTADISTICA ANOVA PARA INSERT

RESUMEN INSERT

Grupos	Cuenta	Suma	Promedio	Varianza
SQLite	5	0.196	0.0392	9.2E-06
HiperSQL	5	0.197	0.0394	0.0000148
Firebird	5	0.192	0.0384	0.0000088

GRAFICO Nº 04

RESULTADO PRUEBA ANOVA PARA INSERT



0.128 3.89

4.6.1.4 Decisión

Como F (calculado) = 0.128 < Fcrítico = 3.89, entonces se acepta hipótesis nula (H_o) estadística. Además el valor de la probabilidad es de 0.88 es mayor al 0.05. Por tanto no

existe diferencia significativa entre las 3 bases de datos al aplicar el impacto funcional de INSERT

4. 6.1.5 Conclusión:

En conclusión, queda demostrado que no existe diferencia significativa entre las bases de datos.

4.6.2 Prueba de Hipótesis Especifica Nº 2 DELETE

4.6.2.1 Hipótesis alterna.

Hi=No existen diferencias de Impacto funcional DELETE de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.2.2 Hipótesis Nula

H_0=Existen diferencias significativa de impacto funcional DELETE de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.2.3 Hipótesis Estadísticas

H_o: $\mu_1 = \mu_2 = \mu_3$

H_1: No todas las medias son iguales.

Grado de significancia: $\alpha = 0.05$

Estadígrafo de Prueba: La Prueba Anova de un solo factor o una sola vía.

Se recurre al programa del Excel 2007

TABLA Nº 16

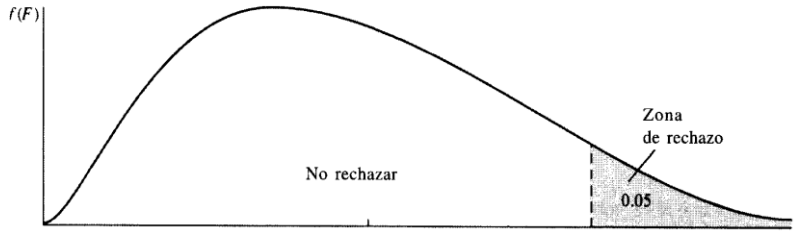
RESULTADOS PRUEBA ESTADISTICA ANOVA PARA DELETE

RESUMEN DELETE

Grupos	Cuenta	Suma	Promedio	Varianza
SQLite	5	0.0066	0.00132	2,00E-09
HiperSQL	5	0.0067	0.00134	3,00E-09
Firebird	5	0.0064	0.00128	2,00E-09

GRAFICO Nº 05

RESULTADO PRUEBA ANOVA PARA DELETE



4.6.2.4 Decisión

Como F (calculado) = 2 < Fcrítico = 3.89, entonces se acepta hipótesis nula (H_o) estadística. Además el valor de la probabilidad es de 0.177 es mayor al 0.05. Por tanto no existe diferencia significativa entre las 3 bases de datos al aplicar el impacto funcional de DELETE

4. 6.2.5 Conclusión:

En conclusión, queda demostrado que no existe diferencia significativa entre las bases de datos.

4.6.3 Prueba de Hipótesis Especifica Nº 3 UPDATE

4.6.3.1 Hipótesis alterna.

Hi=No existen diferencias de Impacto funcional UPDATE de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.3.2 Hipótesis Nula

H_0=Existen diferencias significativa de impacto funcional UPDATE de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.3.3 Hipótesis Estadísticas

H_o: $\mu_1 = \mu_2 = \mu_3$

H_1: No todas las medias son iguales.

Grado de significancia: $\alpha = 0,05$

Estadígrafo de Prueba: La Prueba Anova de un solo factor o una sola vía.

Se recurre al programa del Excel

TABLA Nº 17

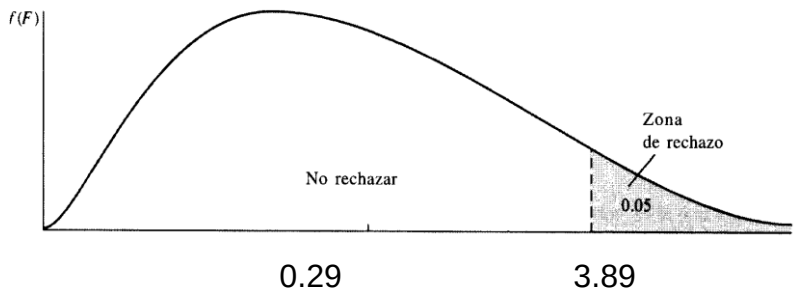
RESULTADOS PRUEBA ESTADISTICA ANOVA PARA UPDATE

RESUMEN UPDATE

Grupos	Cuenta	Suma	Promedio	Varianza
SQLite	5	0.1748	0.03496	0.00095058
HiperSQL	5	0.1157	0.02314	0.00073468
Firebird	5	0.1145	0.0229	0.00072454

GRAFICO Nº 06

RESULTADO PRUEBA ANOVA PARA UPDATE



4.6.3.4 Decisión

Como F (calculado) =0.29 < Fcrítico = 3.89, entonces se acepta hipótesis nula (H_o) estadística. Además el valor de la probabilidad es de 0.749 es mayor al 0.05. Por tanto no existe diferencia significativa entre las 3 bases de datos al aplicar el impacto funcional de UPDATE.

4. 6.3.5 Conclusión:

En conclusión, queda demostrado que no existe diferencia significativa entre las bases de datos.

4.6.4 Prueba de Hipótesis Especifica Nº 4 SELECT

4.6.4.1 Hipótesis alterna.

Hi=No existen diferencias de Impacto funcional SELECT de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.4.2 Hipótesis Nula

H_0=Existen diferencias significativa de impacto funcional SELECT de los sistemas de BD libres, SQLite, HiperSQL y Firebird.

4.6.4.3 Hipótesis Estadísticas

H_o: $\mu_1 = \mu_2 = \mu_3$

H_1: No todas las medias son iguales.

Grado de significancia: $\alpha = 0,05$

Estadígrafo de Prueba: La Prueba Anova de un solo factor o una sola vía.

Se recurre al programa del Excel

TABLA Nº 18

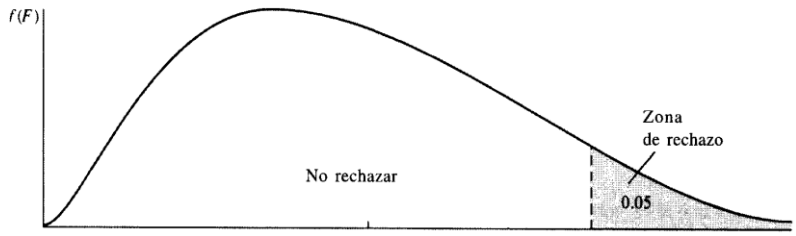
RESULTADOS PRUEBA ESTADISTICA ANOVA PARA SELECT

RESUMEN SELECT

Grupos	Cuenta	Suma	Promedio	Varianza
SQLite	5	0.0105	0.0021	0.00000003
HiperSQL	5	0.0103	0.00206	0.000000028
Firebird	5	0.0103	0.00206	0.000000043

GRAFICO Nº 07

RESULTADO PRUEBA ANOVA PARA SELECT



0.079 3.89

4.6.4.4 Decisión

Como F (calculado) =0.079 < Fcrítico = 3.89, entonces se acepta hipótesis nula (H_o) estadística. Además el valor de la probabilidad es de 0.92 es mayor al 0.05. Por tanto no existe diferencia significativa entre las 3 bases de datos al aplicar el impacto funcional de SELECT.

4. 6.4.5 Conclusión:

En conclusión, queda demostrado que no existe diferencia significativa entre las bases de datos.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Como consecuencia del trabajo de tesis realizado se ha llegado a las siguientes conclusiones:

- ✓ Como consecuencia del desarrollo de la tesis se concluye que se ha cumplido con el objetivo de determinar que no existe diferencia significativa en los impactos funcionales en las tres bases de datos analizadas por lo que sería su uso similares.
- ✓ El impacto funcional de INSERT es similar en las tres bases de datos.
- ✓ El impacto funcional de DELETE es similar en las tres bases de datos.
- ✓ El impacto funcional de UPDATE es similar en las tres bases de datos.
- ✓ El impacto funcional de SELECT es similar en las tres bases de datos.

6.2. Recomendaciones

Al concluir nuestro proyecto de tesis podemos sugerir las siguientes recomendaciones:

1. Se pueden utilizar cualquier motor de base de datos de uso libre para la construcción de las aplicaciones que desee realizar cualquier desarrollador de sistemas.
2. Fomentar el uso de las bases de datos SQLite y de HiperSQL y el Firebird, ya que presentan algunas ventajas sobre otras BD libres.
3. Incluir en las asignatura de Base de Datos, el empleo de las base de datos analizadas, que permitan poner a disposición de los futuros profesionales de más alternativas en su vida profesional para beneficio de los negocios donde existe limitaciones de inversión sobre todo en el sector de la Pymes.

REFERENCIAS BIBLIOGRAFICAS

1. Gómez, Marcelo. Introducción a la Metodología de la Investigación Científica. Córdoba, 2006, Brujas, 95pp.

2. **Rodrigo Daniel Guayaquil loor y José Luis Silva palma**

   http://www.dspace.espol.edu.ec/handle/123456789/16718

   Consultado el 10 abril de 2013

3. **Nieto, Enrique Peña. www.gacontent.edomex.gob.mx.**

   [.http://qacontent.edomex.gob.mx/edomex/noticias/EDOMEX_NOTICIAS_125910 Consultado el 10 abril de 2013

4. **Javier Briceño Sanz 2010 Leganés**

   http://earchivo.uc3m.es/bitstream/handle/10016/10587/PFC_FranciscoJavier_Briceno_Sanz.pdf?sequence=1

   Consultado el 10 abril de 2013

5. **Washington lizano1, Kleber Palacios, Miguel Vargas, Edgar Leyton 2002**

   http://www.dspace.espol.edu.ec/bitstream/123456789/665/1/1171.pdf

   Consultado el 10 abril de 2013

6. **Municipalidad Distrital de San Isidro**

   http://www.netkrom.com/es/success_stories_peru7.php?id=peru

   Consultado el 14 Abril de 2013

7. **Municipalidad Distrital de San Borja**

   http://www.netkrom.com/es/success_stories_peru6.php?id=peru

   Consultado el 14 Abril de 2013

8. **Municipalidad de Arequipa.**

http://www.netkrom.com/es/success_stories_peru1.php?id=peru

Consultado el 28 Abril de 2013

9. **Bamir Impex SRL.** www.compute-rs.com. http://www.compute-rs.com/es/consejos-527034.htm.

Consultado el 02 mayo de 2013

10. **Telefónica.** Ingeniería de Seguridad.

http://www.tiseguridad.com.pe/ccerrado.shtml.

Consultado el 5 mayo de 2013

11. **Proyecta Consultores Asociados SAC.** www.proyectaperu.com. http//www.proyectaperu.com.

Consultado el 10 mayo de 2013

12. **PROYECTA Consultores Asociados SAC.** Proyecta Consultoría. http:/proyectaperu.com/svi2.pdf.

Consultado el 22 mayo de 2013

13. **Demodesk.** Demodesk. http://www.domodesk.com/.

Consultado el 1 Junior de 2013

14. Wikipedia Enciclopedia Libre. http://es.wikipedia.org/wiki/Circuito.

Consultado el 6 junio de 2013

19. **Tech Community. Tech Community. http://www.tech-faq.com/es/dvr.html.**

Consultado el 10 junio de 2013

20. **IT Morelia. www.itmorelia.galeon.com.**

http://itmorelia.galeon.com/concepto.htm.

Consultado el 15 junio de 2013

21. **Proyecta Consultoria. Proyecta Consultoria.**

    http://www.proyectaperu.com/svi.pdf.

    Consultado el 18 junio de 2013

22. **Master Magazine. www.mastermagazine.info.**

    http://www.mastermagazine.info/termino/4156.php.

    Consultado el 19 junio de 2013

23. **Isidro, La Municipalidad Distrital de San. www.netkrom.com.**

    http://www.netkrom.com/es/success_stories_lima5.html.

    Consultado el 21 junio de 2013

ANEXOS

ANEXO Nº 01: MATRIZ DE CONSISTENCIA

TÍTULO: "IMPACTO E INFLUENCIA DE LAS BASES DE DATOS OPEN SOURCE Y LA DIFERENCIA EN SU ELECCION"

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLES	INDICADORES	ÍNDICES	MÉTODOS	TÉCNICAS	INSTRUMENTOS
Problema Principal	***Objetivo General***	***Hipótesis General***						
Problema principal. ¿Cuál será las diferencias del impacto funcional de los sistemas de BD libres SQLite, HiperSQL y Firebird?	**Objetivo General:** Determinar las diferencias del impacto de los sistemas de BD libres SQLite, HiperSQL y Firebird. **Objetivos Específicos:** ○ Evaluar la capacidad de almacenamiento de las bases de datos libres SQLite, HiperSQL y Firebird.	H_1: No existen diferencias de Impacto funcional de los sistemas de BD libres, SQLite, HiperSQL y Firebird. H_o: Existen diferencias significativa de impacto funcional de los sistemas de BD libres, SQLite, HiperSQL y Firebird..	**DEFINICIÓN CONCEPTUAL** ***Variable Independiente:*** X= Sistemas de BD libres, SQLite, HiperSQL y Firebird ***Variable Dependiente:*** Diferencia en su aplicación.		No – Si 300-420 seg. 10-15 min. 15-20 300-600 seg. S/.	**Tipo de Investigación:** Aplicada **Nivel de investigación:** Descriptiva-correlacional **Diseño de la investigación:** Cuasi Experimental Ge: O_1 X O_2 Universo: La investigación se aplica a todos los sistemas de BD libres Muestra:La muestra seleccionada es de tipo intencional, para las tres BD libre a analizarse	Pruebas de BD Instrumentos. Ficha de resultados.	Fichas de Observación.

	○ Analizar los tiempos de respuesta en el manejo de los datos de las bases de datos libres SQLite, HiperSQL y Firebird. Determinar el tiempo de respuesta de las consultas de las bases de datos libres SQLite, HiperSQL y Firebird.							

