

Universidad Cenfotec

Maestría en Tecnología de Bases de Datos

Documento final de Proyecto de Investigación Aplicada 2

Evaluación de Motores de Bases de Datos en Memoria

Morún Camacho, Olger

Sánchez Chaves, Iron

Noviembre 2016

Declaratoria de derechos de autor

Queda prohibida la reproducción y distribución de esta obra para fines lucrativos, está permitido fotocopiar o escanear algún fragmento de ésta obra para fines académicos, de investigación o mejoras al trabajo realizado.

Agradecimientos

Al profesor tutor José A. Cabezas Jaikel:

Por haber aceptado nuestra solicitud de apoyo para la realización de este proyecto, ya que requeríamos de alguien con el suficiente compromiso a lo largo del proceso y el amplio conocimiento en los temas relacionados con bases de datos como el que posee.

A mi compañero de trabajo:

Ya que mantuvo en todo momento su compromiso y luchó fuertemente por ver un trabajo bien elaborado, con el cual dejar un buen aporte de un tema poco explorado a nivel nacional y que pueda ser de provecho para futuros proyectos.

A la familia:

Que de cierta manera nos apoyó durante toda la carrera al ver nuestros deseos de superación a nivel personal y profesional.

Índice de Contenidos

Abstract x

Capítulo 1. Introducción 1

1.1. Generalidades de las bases de datos en memoria 1

1.2. Antecedentes del problema 1

1.3. Definición y descripción del problema 2

1.4. Justificación 2

1.5. Viabilidad 3

1.5.1. Punto de vista técnico 3

1.5.2. Punto de vista económico 3

1.6. Objetivos 4

1.6.1. Objetivo general 4

1.6.2. Objetivos específicos 4

1.7. Alcances y Limitaciones 5

1.7.1. Alcances 5

1.7.2. Limitaciones 5

1.8. Estado de la cuestión 5

1.8.1. Ejecución de la revisión 5

1.8.2. Ejecución de la selección en la fuente IEEE 5

1.8.3. Selección de estudios iniciales 6

1.8.4. Evaluación de la calidad de los estudios 8

1.8.5. Revisión de la selección 9

1.8.6. Extracción de información 9

Capítulo 2. Marco Conceptual 17

2.1. ¿Qué es una base de datos en memoria? 17
2.2. Historia 18
2.3. Control de concurrencia 19
2.4. Memoria Caché 19
2.5. Paginación en Memoria 19
2.6. Máquinas Multi-Core 19
2.7. Hardware 20
2.8. ¿Qué es un ETL? 20
2.9. ¿Qué es una llave Primaria? 20
2.10. OLTP 21
2.11. OLAP 21
Capítulo 3. Marco Metodológico 22
3.1. Tipo de investigación 22
3.2. Alcance Investigativo 22
3.3. Enfoque de la investigación 22
3.4. Diseño 23
3.5. Población y muestreo 23
3.6. Estrategia de desarrollo de la propuesta 23
Capítulo 4. Análisis del diagnóstico 24
4.1. Actualidad de las bases de datos en memoria 24
4.2. OLAP en memoria 25
4.3. OLTP en memoria 26
4.4. Consideraciones para implementar bases de datos en memoria 27
4.5. Los sistemas sin bases de datos en memoria 28
4.6. ¿Quiénes utilizan bases de datos en memoria? 30

4.7. La selección de los motores de bases de datos en memoria 32
Capítulo 5. Propuesta de solución 34
5.1. SolidDB 34
5.1.1. Arquitectura de SolidDB 34
5.1.2. Instalación de SolidDB 38
5.1.3. Configuración de SolidDB 41
5.2. H2 44
5.2.1. Arquitectura 45
5.2.2. Características 47
5.2.3. Instalación 47
5.2.3.1. Requerimientos 47
5.2.3.2. Consola de H2 47
5.2.3.3. Plataformas Soportadas 48
5.2.3.4. Instalación de H2 48
5.3. Derby DB 50
5.3.1. Arquitectura 51
5.3.2. Características 52
5.3.3. Requerimientos 53
5.3.4. Instalación 53
5.3.5. Consola del motor 54
5.3.6. Plataformas soportadas 55
5.4. SQLite 56
5.4.1. Arquitectura 56
5.4.2. Características 57
5.4.3. Instalación 58

5.4.4. Requerimientos 58
5.4.5. Consola de SQLite 59
5.4.6. Plataformas soportadas 59
5.5. Pruebas de rendimiento 60
5.5.1. Elección del equipo para ejecución de pruebas 61
5.5.2. Recopilación de datos 63
5.5.2.1. Formato de recopilación de datos 64
5.5.2.2. Definición y estructura de las tablas utilizadas 64
5.5.3. Herramienta de monitoreo 66
5.5.4. Definición de pruebas de rendimiento 67
5.5.4.1. Diseño Experimental 68
5.5.4.2. Aspectos a considerar para evitar contaminación y sesgos 69
5.5.4.3. Prueba de rendimiento: Carga de Datos 70
5.5.4.3.1. Justificación del uso del dataset 71
5.5.4.3.2. Preparación y ejecución de la prueba 71
5.5.4.3.3. Gráficas de rendimiento 75
5.5.4.3.3.1. Carga Masiva de Datos 75
5.5.4.3.3.2. Consulta Cruzada de Inventarios 79
5.5.4.4. Análisis y consideraciones de los resultados obtenidos 82
5.5.4.5. Plantilla para evaluación de rendimiento 83
Capítulo 6. Conclusiones y Recomendaciones 87
6.1. Conclusiones 87
6.2. Recomendaciones 93
Capítulo 7. Reflexiones Finales 96
Capítulo 8. Trabajos a Futuro 97

Capítulo 9. Referencias 98

Capítulo 10. Anexos 1

Índice de Tablas

Tabla 1 - Cadena de búsqueda 7
Tabla 2 - Resultados búsqueda IEEE 8
Tabla 3 - The Performace Survey of in Memory Database 10
Tabla 4 - Simulation of distributed real-time transactions 11
Tabla 5 - A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services 12
Tabla 6 - Performance optimizations for distributed real-time text indexing 13
Tabla 7 - Towards real-time parallel processing of spatial queries 16
Tabla 8 - A scalable lightweight performance monitoring tool for storage clusters 17
Tabla 9 - Diferencias entre las MMDB y las Bases de Datos Tradicionales 19

Índice de Figuras

Ilustración 1 Criterios de búsqueda IEEE 7
Ilustración 2 Cantidad de Hilos de Ejecución (Threads) 75
Ilustración 3 Cantidad de Hilos de Ejecución (Threads) 75
Ilustración 4 Tiempo de Ejecución 76
Ilustración 5 Uso de Memoria (Carga de Datos) 76
Ilustración 6 Uso del Procesador Ciclos de Reloj (Carga de Datos). Métricas expresadas en Millones. 77
Ilustración 7 Lectura de Disco Duro (Carga de datos) 78
Ilustración 8 Cantidad Escritura Disco Duro 78

Ilustración 9 Tiempo Ejecución Consulta 80
Ilustración 10 Uso del Procesador (Ciclos de Reloj). Métricas expresadas en Millones. 81
Ilustración 11 Cantidad Lecturas a Disco Duro..... 81
Ilustración 12 Cantidad Escrituras a Disco Duro 82

Abstract

Las bases de datos en memoria (MMDB) han empezado a prosperar, en una era en la que la distribución y procesamiento de la información casi en tiempo real se ha vuelto una necesidad en diferentes áreas de la industria a nivel global. Para las grandes compañías se ha vuelto indispensable tener la información lista para analizarla, se han dado cuenta de que la información es una herramienta muy poderosa que les permite competir con sus rivales.

El procesamiento de los datos en tiempo real para satisfacer las necesidades de diferentes objetivos de rendimiento traza nuevos retos, tanto en la forma de almacenar la información como en la capacidad de procesamiento a nivel de equipos de hardware. En este proyecto se investigaron varias arquitecturas de bases de datos en memoria para analizar algunos modelos que permitan optimizar el rendimiento para carga de datos y consultas a modo de reportes en sistemas altamente transaccionales.

Capítulo 1. Introducción

Las bases de datos en memoria (MMDB) son un tema relativamente nuevo, pero gracias al cambio que ha venido ocurriendo durante los últimos años en relación con el costo en infraestructura tecnológica y uso de la memoria RAM, ha sido posible reinventar la forma en que se almacenan los datos. Sin duda alguna las MMDB están cobrando mucha popularidad en la actualidad ya que están diseñadas para manejar información procesada, lista para ser utilizada sin tener que utilizar el acceso a disco duro para obtener la información, lo que permite optimizar el rendimiento de las consultas.

En este proyecto se configuraron cuatro motores de bases de datos, cuya arquitectura se basa principalmente en el almacenamiento en memoria, algunos con características replicación en disco, para garantizar la persistencia de la información. Posterior a esta tarea de configuración se realizaron diferentes procesos de carga de datos utilizando muestras representativas de información para el análisis y pruebas de rendimiento para cada una de las arquitecturas elegidas.

1.1. Generalidades de las bases de datos en memoria

Se va a empezar por explicar brevemente el funcionamiento de las bases de datos en memoria que, según Berkowitz (2002), utilizan una memoria compartida para llevar un registro en caché de las llaves leídas en una base de datos, así mismo controlan la actualización de registros y llaves a través de un proceso de administración que es arbitrado por el motor de base de datos.

1.2. Antecedentes del problema

Muchos sistemas de base de datos tienen motores de almacenamiento que fueron diseñados asumiendo que los datos serían almacenados en el disco y con frecuencia paginados dentro y fuera de la memoria. Los motores de bases de datos tradicionales ofrecen pocas posibilidades de alcanzar el rendimiento máximo en los servidores actuales y futuros (máquinas multi-core).

1.3. Definición y descripción del problema

Existe la necesidad de mejorar la forma en que se almacena y se consulta la información generada por sistemas altamente transaccionales. Las bases de datos tradicionales basadas únicamente en almacenamiento en disco duro han empezado a experimentar problemas de rendimiento, debido a la capacidad limitada que se tiene a nivel de hardware para el acceso concurrente de los datos. El acceso se ve limitado por la capacidad de lectura y escritura que tienen los discos duros, por lo cual a mayor capacidad de lectura y escritura es mayor el costo de adquisición de las unidades de disco.

Es por lo anterior que en este proyecto se pretende describir el funcionamiento de arquitecturas de almacenamiento en memoria, con el objetivo de analizar modelos eficientes de bases de datos en memoria principal. El fin es de eliminar el "cuello de botella" que existe actualmente a nivel de hardware, producto del almacenamiento y recuperación de la información que se alberga en disco duro.

1.4. Justificación

Actualmente en nuestro país no se encontró ninguna empresa que esté utilizando este tipo de tecnologías (MMDB); con esto no se quiere decir que no se esté utilizando, sino más bien que aún no es común encontrar empresas que lo utilicen. Por lo tanto, la idea con este proyecto fue desarrollar un análisis que le permita al lector tener un criterio y un panorama de lo que implica implementar un sistema de bases de datos en memoria, así como de las ventajas que podría obtener en términos de rendimiento para ejecución de procesos transaccionales.

En este proyecto se realizó un análisis a diferentes motores de bases de datos en memoria, donde se utilizó diferentes escenarios a fin de evaluar el rendimiento y la eficacia de cada uno de ellos. El propósito es explicar el comportamiento ante distintas situaciones, con lo cual el lector pueda tener una idea general y un punto de partida para tomar una decisión sobre qué motor de base de datos en memoria podría eventualmente ajustarse a una necesidad particular, o incluso evaluar un motor de bases de datos para intentar tropicalizarlo a su problema específico, en caso de requerirlo.

1.5. Viabilidad

1.5.1. Punto de vista técnico

Hoy día existen motores que realizan el manejo de los datos en memoria, algunos de ellos almacenan información del tipo llave-valor en memoria para pequeñas cadenas de datos arbitrarios (textos, objetos) de resultados de llamadas a base de datos. Otros pueden funcionar como una base de datos tradicional, o bien de forma independiente, algunos tienen soporte para varios lenguajes de programación. Algunos disponen de tablas in-memory y basadas en disco, y con soporte cliente-servidor. La mayoría disponen de herramientas de tipo consola de comandos SQL o una interfaz gráfica para las consultas.

Se cuenta con conocimientos previos adquiridos a lo largo de la experiencia profesional, lo cual permite como investigadores estar familiarizados con herramientas y lenguajes de programación, así como la ejecución de consultas en lenguaje SQL. Este conocimiento facilita el desarrollo del proyecto.

1.5.2. Punto de vista económico

Para las diferentes pruebas y análisis que se realizaron en este proyecto se utilizaron motores de bases de datos y herramientas tanto de software libre como de código propietario (que ofrecen un tiempo de prueba para evaluar su utilidad, conocido comúnmente como versiones trial), por lo que no se incurrió en costos adicionales de licenciamiento. Por otra parte en cuanto a equipo de hardware se refiere, se aprovechó un equipo disponible proporcionado por una empresa, el cual en sí no es un servidor, sin embargo tiene muy buenas características, en términos de recursos, capaz de cumplir con las necesidades de procesamiento para la ejecución de las pruebas realizadas durante el desarrollo de este proyecto.

Por otra parte en cuanto a costos de investigación y ejecución se refiere, se pueden contabilizar las siguientes cifras con un estimado del coste total:

Cantidad de Recursos	2 personas
Horas por Mes	80

Cantidad de Meses	7
Total de Horas	560
Costo por Hora	$25
Costo Total	$14 000

1.6. Objetivos

1.6.1. Objetivo general

Evaluar el rendimiento de cargas masivas de datos y ejecución de consultas en motores de bases de datos en memoria.

1.6.2. Objetivos específicos

Para este proyecto se propuso un enfoque orientado hacia los siguientes objetivos específicos, los cuales permitieron realizar la evaluación de los motores de bases de datos estudiados:

1. Conocer información relevante de las tecnologías actuales disponibles, en el ámbito de las bases de datos en memoria.
2. Comprender el funcionamiento interno de cuatro de los modelos actuales.
3. Realizar la instalación y configuración de las cuatro plataformas de bases de datos en memoria investigadas.
4. Comparar los cuatro modelos de arquitectura instalados para someterlos a pruebas de rendimiento de carga y consultas, empleando volúmenes de datos.
5. Organizar los resultados obtenidos, basados en las pruebas realizadas, con el fin de documentar el comportamiento de las tecnologías seleccionadas.

1.7. Alcances y Limitaciones

1.7.1. Alcances

El alcance de este proyecto está definido en la formulación del presente documento, donde se plasmaron los hallazgos que se obtuvieron a partir de las pruebas realizadas utilizando los MMDB, tomando como objeto de estudio la data que fue posible manipular y que fue lo suficientemente representativa. Con esta información se pudo obtener una diferencia en resultados que hicieron posible la determinación en las diferencias de rendimiento para cada una de las bases de datos analizadas.

1.7.2. Limitaciones

La propuesta de desarrollo de esta investigación no involucró la entrega de un sistema para uso operativo en ninguna institución. De ninguna forma se realizó indicaciones o aseveraciones de cuál herramienta debe ser utilizada, ya que este documento es solamente un punto de partida para que el lector pueda tomar una decisión o emprender su propio análisis de acuerdo con su necesidad, utilizando este documento como una guía que lo pueda apoyar sobre cómo realizar diferentes pruebas de rendimiento a un motor de base de datos en memoria.

1.8. Estado de la cuestión

1.8.1. Ejecución de la revisión

La revisión para esta investigación fue realizada sobre la fuente de datos disponible a través de la IEEE, utilizando el motor de búsqueda que esta posee: IEEE Xplore Digital Library.

1.8.2. Ejecución de la selección en la fuente IEEE

Se utilizó la búsqueda de palabras claves avanzadas y algunas frases. En un principio se utilizó la búsqueda simple pero los resultados devueltos por el motor de búsqueda no eran los requeridos para este trabajo de investigación. Luego se realizaron búsquedas basadas en los filtros especiales, utilizando títulos de documentos y publicaciones, pero los resultados no fueron los esperados. Al final se realizó la búsqueda de publicaciones basándose solamente en la metadata de los documentos.

1.8.3. Selección de estudios iniciales

La búsqueda realizada fue sobre la opción de “Búsqueda avanzada”, seleccionando las siguientes opciones:

Tipo de búsqueda: Solo en la metadata (en el contenido completo del documento)

Filtros de contenido: Todos los resultados

Años de publicación: Todos

Tema: In memory database

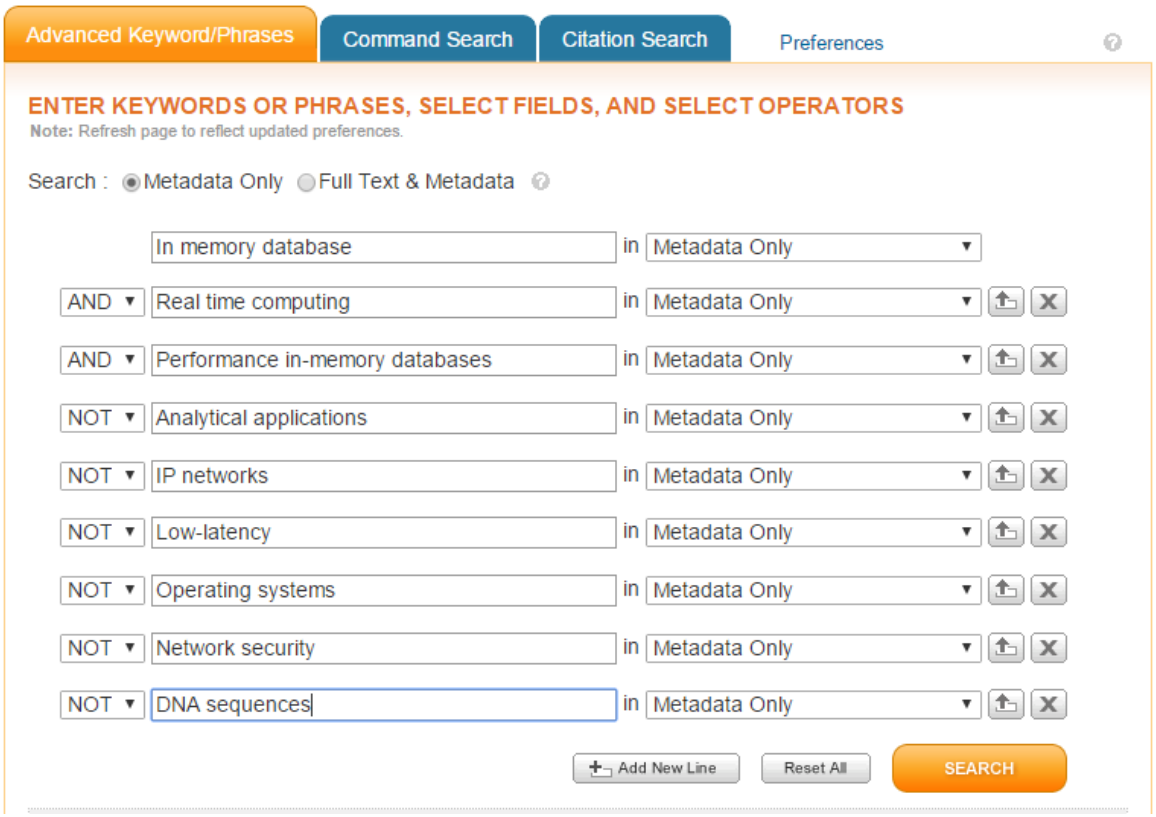


Ilustración 1 Criterios de búsqueda IEEE

Una vez configurada la búsqueda se procedió a realizar la consulta adaptando la cadena a la sintaxis propia de la fuente. La cadena que se muestra a continuación fue generada automáticamente por el motor de búsqueda de la IEEE.

(((((((((in memory database) AND real time computing) AND performance in-memory databases) NOT analytical applications) NOT IP networks) NOT low-latency) NOT operating systems) NOT network security) NOT DNA sequences)

Tabla 1 - Cadena de búsqueda

La ejecución de la búsqueda en la IEEE lanzó 7 resultados. A continuación se detallan los resultados obtenidos.

1	**The Performance Survey of in Memory Database** Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on 2015 Yinfeng Wang; Guiquan Zhong; Lin Kun; Longxiang Wang; Huang Kai; Fuliang Guo; Chengzhe Liu; Xiaoshe Dong

2	**Simulation of distributed real-time transactions** Parallel and Distributed Real-Time Systems, 1994. Proceedings of the Second Workshop on 1994 Van der Stok, P.D.V.; Thijssen, P.T.A.
3	**A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services** Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on 2012 Jing Zhang; Gongqing Wu; Xuegang Hu; Xindong Wu
4	**T*-tree: a main memory database index structure for real time applications** Real-Time Computing Systems and Applications, 1996. Proceedings., Third International Workshop on 1996 Kong-Rim Choi; Kyung-Chang Kim
5	**Performance optimizations for distributed real-time text indexing** High Performance Computing (HiPC), 2009 International Conference on 2009 Narang, A.; Swaminathan, K.; Agrawal, P.
6	**Towards real-time parallel processing of spatial queries** Parallel Processing, 2003. Proceedings. 2003 International Conference on 2003 Haibo Hu; Manli Zhu; Dik-Lun Lee
7	**A scalable lightweight performance monitoring tool for storage clusters** Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on 2015 Bauer, D.; Feridun, M.

Tabla 2 - Resultados búsqueda IEEE

1.8.4. Evaluación de la calidad de los estudios

Se consideró que los estudios que se encontraron en la base de conocimiento de la IEEE fueron lo suficientemente evaluados y considerados una fuente sólida que ayudó a fundamentar la investigación presente. Además, se afinaron los filtros de búsqueda de manera que se pudo tener mayor certeza y mejores coincidencias con respecto a la necesidad.

1.8.5. Revisión de la selección

La selección de los estudios fue sometida a la revisión del conocimiento con el que se contaba acerca del tema y de la necesidad que se tenía para el desarrollo de la investigación.

1.8.6. Extracción de información

Se definió una estructura a manera de tabla donde se rescataron los puntos clave que dieron una idea clara de los puntos que abarcó cada artículo o estudio que la búsqueda sobre la base de datos de conocimiento retornó.

A continuación, se presenta la información extraída de cada estudio primario.

Identificación
Título: The Performace Survey of in Memory Database
Publicación: Parallel and Distributed System (ICPADS), 2015 IEEE 21st International Conference.
Autores: Yinfeng Wang.; ShenZhen Inst. Of Inf. Technol, Shenzhen, China
Referencia:
Descripción
Área: Bases de Datos en Memoria
Resumen: La continua necesidad de rendimiento para el procesamiento de big data y aplicaciones críticas hace que sea necesario un esquema de alta disponibilidad, alto volumen y alta escalabilidad.
Aspectos a destacar
Se propone un análisis profundo donde se evalúa la velocidad y volumen con el fin

de observar el rendimiento de las transacciones en tiempo real.

Se demuestra que el uso de arquitectura NoSQL es la más apta para el desarrollo de estos procesos.

Se realizan comparaciones entre los distintos motores de bases de datos en memoria que se ofrecen en el mercado.

TABLE 1. COMPARISON OF IMDB SYSTEM FEATURES

IMDB Name	SQL 92 Standard	Concurrency	Data Size	Storage Type	System Architecture	Availability	Scalability
eXtremeDB	SQL-89	multi-version concurrency control(MVCC) 0.7 times/ms	Billion	No Translation	Embedded /Shared memory	Two phase synchronous mode	One pair of N memory mirroring
Oracle TimesTen	√	2.5 ms read 10 ms write Transaction isolation Locks,MVCC	Billion	Relational model based on row	C/S, Embedded	Transaction logging and database checkpoint	X/Open XA Specification Java Transaction API
Altibase	√	MVCC	Billion	Multiple data formats	C/S	Copy, Switch, Write Ahead Logging	Star copy 1:32
SQLite	√	The entire database file for read / write lock	Million	Typelessness Five kinds of genetic types	Embedded	Lasting ACID	None
Berkeley DB	√	Support thousands of concurrent threads	Maximum 256TB	Any type of key / value pairs	Embedded	Two lock technology and write-ahead logging policy	None

Tabla 3 - The Performace Survey of in Memory Database

Identificación
Título: Simulation of distributed real-time transactions
Publicación: Parallel and Distributed Real-Time System, 1994 Proceeding of the Second Workshop
Autores: Van der Stok, P.D.V. Dept. of Math. & Comput. Sci, Eindhoven Univ. of Technol., Netherlands
Referencia:
Descripción
Área: Bases de Datos en Memoria
Resumen: Los objetos son almacenados en memoria de dos maneras, la primera de manera sintáctica en donde tiene un objeto en común de las cuales se invocan, la segunda de manera de tiempo en común.
Aspectos a destacar
Uso de algoritmos entre los cuales se encuentran control de concurrencia optimista, indicador de tiempo y bloqueos de base.

El rendimiento de los algoritmos es analizado de manera independiente.

Tabla 4 - Simulation of distributed real-time transactions

Identificación
Título: A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services
Publicación: Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference
Autores: Jing Zhang Dept. of Comput. Sci., Hefei Univ. of Technol., Hefei, China
Referencia:
Descripción
Área: Bases de Datos en Memoria
Resumen: Se utiliza las bondades del sistema de archivos Hadoop, donde se encuentran servicios que acceden directamente la memoria y otros que se encargan de acceder el disco local. Los archivos son cargados a memoria para que puedan ser accedidos por los servicios cliente.
Aspectos a destacar
Los archivos son replicados en nodos para tener mayor robustez y que se alivie la carga de trabajo. Se menciona el rendimiento de tener archivos en memoria.

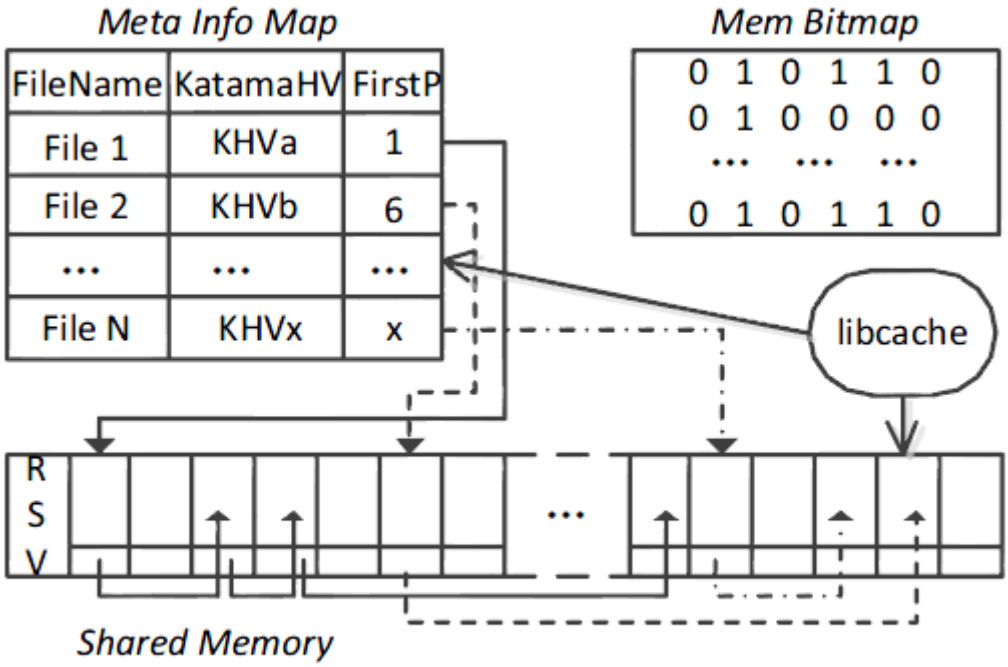


Figure 3. Shared memory management of HDCache service

Tabla 5 - A Distributed Cache for Hadoop Distributed File System in Real-Time Cloud Services

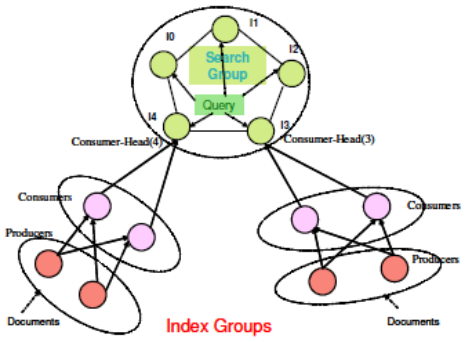
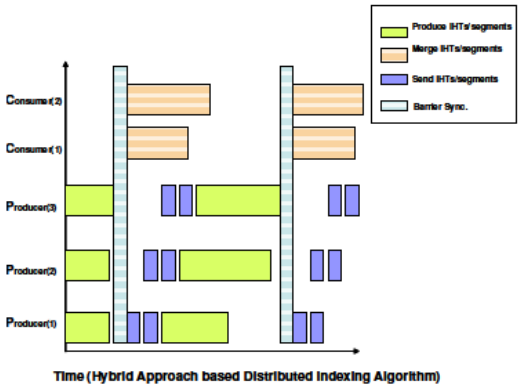
Identificación
Título: Performance optimizations for distributed real-time text indexing
Publicación: High Performance Computing (HiPC), 2009 International Conference on 2009
Autores: Narang, A.; Swaminathan, K.; Agrawal, P.
Referencia:
Descripción
Área: Algoritmos para la optimización de índices
Resumen: Indexación en tiempo real y la búsqueda de grandes cantidades de datos que circulan a gran velocidad. La estructura de datos y el algoritmo basado en el grupo distribuido (referido como Pghtl) en tiempo real para la indexación de texto en memoria.
Aspectos a destacar
Indexación de texto distribuido basado en un grupo de algoritmo híbrido. Optimizaciones de comunicación para la indexación distribuida. Indexación rendimiento y escalabilidad.  

Tabla 6 - Performance optimizations for distributed real-time text indexing

Identificación
Título: Towards real-time parallel processing of spatial queries
Publicación: Parallel Processing, 2003. Proceedings. 2003 International Conference on 2003
Autores: Haibo Hu; Manli Zhu; Dik-Lun Lee

Referencia:

Descripción

Área: Aplicaciones de base de datos espaciales en tiempo real

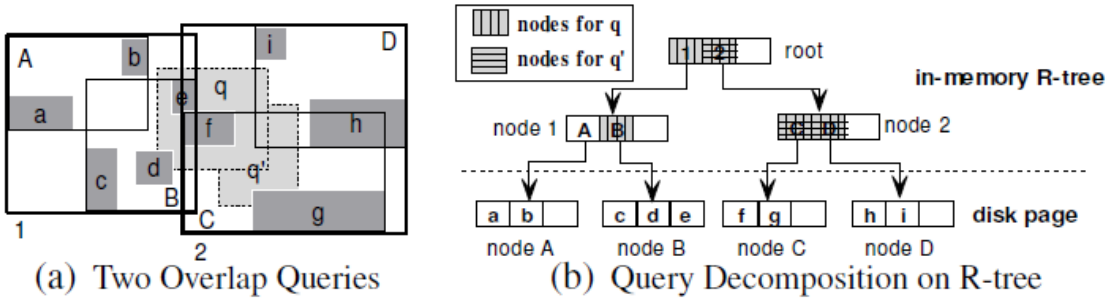
Resumen: Investigación de múltiples consultas espaciales (en concreto, las consultas de ventanas) pueden ser paralelizados, descompuestos, programados y procesados bajo una carga de trabajo en tiempo real con el fin de mejorar el rendimiento del tiempo de ejecución.

Aspectos a destacar

R-tree y el índice de Datos Espaciales.

Sistemas modelos de bases de datos espaciales de la vida real.

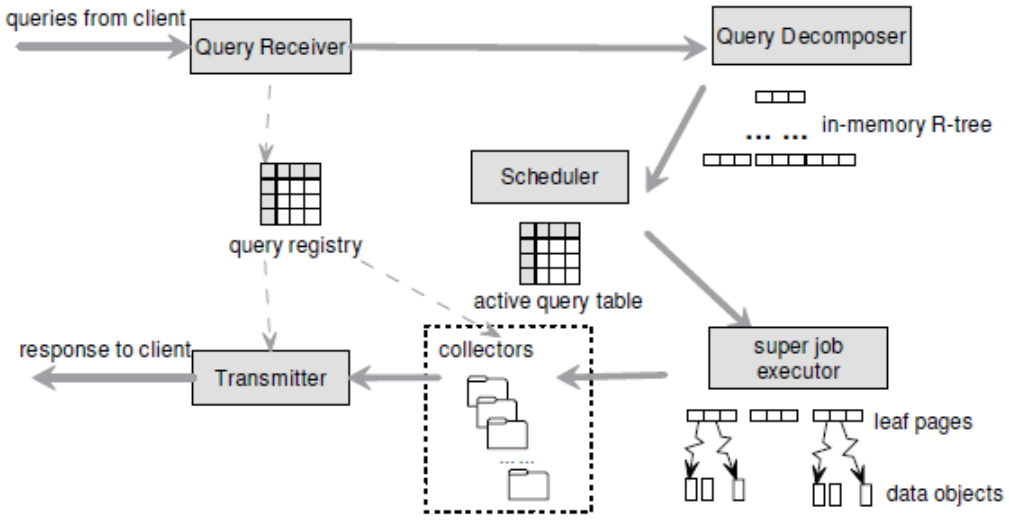
Procesamiento de consultas espaciales en tiempo real.



In-memory R-tree and Query Decomposition

R-tree y consultas activas en memoria

Arquitectura del Sistema de Base de datos espaciales



Spatial Database System Architecture

Programación de algoritmos dinámicos

Algorithm 1 Runtime Scheduler for Immediate Mode

Input: *queue*: queue of jobs to be executed
table: the active query table
Procedure:

```
1: while SYSTEM_UP do
2:   if queue is not empty then
3:     job = queue.popFirst(); job.execute();
4:     table.removeFinishedQueries();
5:   while table.size < S AND exist(nextQuery) do
6:     table.insert(nextQuery); queue.insertJobs(nextQuery);
7:   if table is changed OR system supports deadline then
8:     update weight(q) according to Equation 6
9:     update sat(p)/selec(p) for all jobs in queue
10:    re-order all jobs according to Policy 1
```

Algorithm 2 Runtime Scheduler for Collecting Mode

Input: *queue*: queue of jobs to be executed
table: the active query table
Procedure:

```
1: while SYSTEM_UP do
2:   while queue is not empty do
3:     job = queue.popFirst(); job.execute();
4:   while table.size < S AND exist(nextQuery) do
5:     table.insert(nextQuery); queue.insertJobs(nextQuery);
6:   if system supports deadline then
7:     query.removeMissedQueries();
8:     if table has urgent queries then
9:       nextQuery = table.MostUrgentQueries();
10:      queue.insert(nextQuery.jobs);
11:  if queue is empty then
12:    update  (sum_{j=1}^{intersec(q_i)} sat(p_{i,j})) / (sum_{j=1}^{intersec(q_i)} selec(p_{i,j}))  for all queries
13:    nextQuery = table.findBestQueries();
14:    queue.insert(nextQuery.jobs);
```

Tabla 7 - Towards real-time parallel processing of spatial queries

Identificación

Título: A scalable lightweight performance monitoring tool for storage clusters
Publicación: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on 11-15 May 2015
Autores: Bauer, D.; Feridun, M.
Referencia:
Descripción
Área: Sistema de Gestión (PMMS)
Resumen: El Sistema de Gestión (PMMS) es una herramienta de monitorización del rendimiento versátil que recoge cientos de miles de mediciones por segundo y suministra esta información en series de tiempo con un rendimiento de casi tiempo real. En su núcleo se encuentra una base de datos en memoria que escala a través de grandes grupos de varios cientos de nodos. Los datos se recopilaron mediante sensores que recogen y envían periódicamente sus datos métricos a los colectores PMMS.
Aspectos a destacar
Herramienta de monitorización Bases de datos en memoria Escalabilidad Colectores PMMS

Tabla 8 - A scalable lightweight performance monitoring tool for storage clusters

Capítulo 2. Marco Conceptual

2.1. ¿Qué es una base de datos en memoria?

Una base de datos en memoria se caracteriza porque su principal fuente de almacenamiento no depende de discos duros, ya que se encuentra en su mayoría establecida en la memoria del computador, por lo tanto es justificable su mayor capacidad de

procesamiento en cuanto a tiempos de repuesta y manipulación de data en comparación con los motores de bases de datos tradicionales.

2.2. Historia

Jim Gray fue pionero en las tecnologías de bases de datos y principal actor en el desarrollo de la tecnología de transacciones a nivel computacional, lo cual dio origen a las ventas en línea además de cajeros automáticos. Jim conceptualizó la tecnología de MMDB aproximadamente hace 30 años, y aseguró que su inclusión y aceptación en las tecnologías empleadas en el manejo de datos se realizaría en los años 2000. Fue en 1980 que su aparición se hizo presente en telecomunicaciones, pero únicamente empleada en componentes internos.

IBM fue el primero en proponer una herramienta basada en la tecnología de motores en memoria, lo cual fue en el año 1978. Por otra parte, se encuentran las bases de datos tradicionales, las cuales se basan en el almacenamiento de datos en disco, además cumplen con un modelo de datos relacional, donde se abstrae un concepto de la vida cotidiana y se genera un modelo del tipo entidad relación. También existe el modelo orientado a objetos, capaz de caracterizar mediante atributos a un objeto cotidiano. Pese a la buena representación de aspectos cotidianos, las bases de datos tradicionales presentan limitaciones, ya que comúnmente se utilizan solamente en entornos transaccionales y analíticos.

A continuación se presenta un cuadro comparativo donde se pueden observar las diferencias entre las MMDB y las Bases de Datos Tradicionales

Bases de Datos en Memoria	Bases de Datos Tradicionales
La data es almacenada en memoria	Toda la data es almacenada en disco y es necesario almacenar datos en memoria cuando es necesario
La data es persistente o volátil dependiendo del producto	La data siempre es persistente

El tamaño de la base de datos depende de la cantidad de memoria	Es virtualmente ilimitada
Estructuras de datos especializados e índices que asumen que la data siempre se encuentra en memoria	Estructuras de datos por lo general en árboles B diseñados para el almacenamiento óptimo en discos.

Tabla 9 - Diferencias entre las MMDB y las Bases de Datos Tradicionales

2.3. Control de concurrencia

García-Molina (1992), en su documento de bases de datos en memoria principal, hace referencia a la eficiencia que se presenta durante el acceso a esta, con respecto al acceso a disco, de modo que se tiene mayor probabilidad de que las transacciones se completen más rápidamente.

2.4. Memoria Caché

Es una memoria que se encuentra situada entre el procesador y la memoria principal y que cuenta con un acceso muy rápido. De acuerdo con UPIICSA (2016) esta memoria puede ser de 5 a 6 veces más veloz que la memoria RAM, y su función principal es servir de puente entre los procesadores cada vez más rápidos y la memoria principal, lo cual afecta directamente el rendimiento y la velocidad de acceso y ejecución de instrucciones.

2.5. Paginación en Memoria

La paginación es una manera de deshacerse del problema de la fragmentación que ocurre en la memoria, es decir que partes distintas de información pueden estar localizadas en distintos lugares dentro de la memoria, a estos fragmentos se les conoce como páginas.

2.6. Máquinas Multi-Core

Son equipos que cuentan con más de un núcleo en cada procesador, con lo cual se persigue aumentar el rendimiento de la ejecución de instrucciones disminuyendo el tiempo de respuesta, debido al mayor poder de procesamiento que tienen, para lograr mayor concurrencia en la ejecución de procesos.

2.7. Hardware

El concepto de hardware se refiere a toda la parte física con la cual cuenta un equipo de cómputo, formado por todos los componentes electrónicos, cables y circuitos de luz, entre otros, los cuales hacen posible el funcionamiento del equipo.

2.8. ¿Qué es un ETL?

Según Anguiano (2004), la definición de ETL por sus siglas en inglés, Extract Transform Load, es la extracción de información, un proceso en el que la información puede provenir de distintas fuentes de datos. Por ejemplo, se puede realizar extracción de información a través de conexiones ODBC para unificar la información en un repositorio único. Transform se refiere al proceso de transformar la información extraída antes de ser almacenada en el repositorio único, acá se pueden hacer conversiones de datos y cálculos previos. Load es el proceso que ejecuta la carga de los datos ya listos y transformados hacia el repositorio único.

2.9. ¿Qué es una llave Primaria?

Según Kwok (2010), es una clave única que permite identificar un valor dentro de un set de registros en una tabla de base de datos, dentro de su función está la de evitar que aparezca un conjunto de valores más de una vez, dentro de un conjunto determinado de columnas.

2.10. OLTP

Con su abreviatura en inglés On-Line Transactional Processing o Procesamiento de Transacciones en Línea, son bases de datos que procuran que cada una de las operaciones realizadas se completen de manera atómica, con lo cual puedan asegurar una consistencia entre toda la información que es almacenada y gestionada. Estas operaciones son de inserción, actualización y eliminación de registros. Dentro de sus características principales se pueden mencionar las siguientes:

- El acceso a los datos comprende la lectura y escritura de manera ya optimizada por el motor.
- Por lo general no existe un histórico de datos, simplemente se posee la última versión de cada registro de datos.

2.11. OLAP

Con su abreviatura en inglés On-Line Analytical Processing o Procesamiento Analítico en Línea, son bases de datos encargadas de realizar estudios sobre datos de grandes cantidades de volumen, con lo cual puedan retornar información útil con la cual se tomen decisiones por parte de sus usuarios o encargados. Dentro de sus características principales se pueden mencionar las siguientes:

- El acceso a los datos se realiza principalmente sobre funciones de lectura, utilizando operadores de agregación o agrupación principalmente.
- Tienen la particularidad de que están diseñadas para soportar grandes cantidades de datos a manera de histórico.
- Son alimentadas principalmente por procesos de Extracción, Transformación y Carga de datos (ETLs).

Capítulo 3. Marco Metodológico

3.1. Tipo de investigación

Para el presente proyecto se utilizó un tipo de investigación evaluativa. Se tomó como punto de referencia la opinión del autor Alvira (1985), quien explica en detalle que este tipo de investigación es la acumulación de información acerca de una intervención sobre su funcionamiento y sobre sus efectos y consecuencias de una manera más formalmente. Es por ello que se eligió este tipo de investigación, debido a su naturaleza, ya que se evaluaron diferentes sistemas de bases de datos en memoria, se observó su comportamiento y se realizó una evaluación de rendimiento ante las diferentes situaciones en el momento que se aplicaron cargas masivas y ejecuciones de consultas.

3.2. Alcance Investigativo

Para este trabajo se abordó un alcance de investigación descriptiva. Según Tamayo (2004), este tipo de investigación comprende la descripción, registro, análisis e interpretación de la naturaleza actual y la comprensión o procesos de los fenómenos, realizando conclusiones de peso acerca de cómo algo o alguien se conduce o funciona en el presente.

Es importante recalcar que la investigación descriptiva se ocupa de las realidades de los hechos, caracterizándose por presentar una interpretación correcta de lo que está ocurriendo. En este trabajo se tomó como punto de partida la evaluación de algunos sistemas de bases de datos en memoria, y se da una explicación de su funcionamiento con la finalidad de evaluar los procesos de carga y ejecución de consultas en cada motor.

3.3. Enfoque de la investigación

Para la presente investigación se hizo un abordaje cualitativo, para definir el enfoque. La investigación cualitativa engloba diferentes disciplinas, perspectivas, métodos y estrategias para la recolección de los datos que se van a caracterizar, lo cual conllevó a

desarrollar un orden conceptual que permita comunicar los resultados obtenidos al grupo de personas interesadas en el área de esta investigación.

3.4. Diseño

Estudio de casos: Se utilizó esta metodología debido a que se eligieron motores de bases de datos en memoria para la ejecución de las pruebas de rendimiento en cargas de información y ejecución de consultas.

Investigación evaluativa: Se utilizó esta metodología porque se evaluaron diferentes motores de bases de datos en memoria usando diferentes escenarios y tipos de información cuyo objetivo fue evaluar el rendimiento en cargas de información y ejecución de consultas.

3.5. Población y muestreo

La población y muestreo que se consideró para ejecutar la investigación involucró la recolección de una cantidad de datos lo suficientemente representativa, de manera que se pudo realizar cargas y procesamiento de los mismos por los motores de bases de datos que se utilizaron. Se utilizó información transaccional de varios años, donde se involucraron tipos de datos de los más usados comúnmente, de modo que se pudo obtener resultados lo más acordes con la realidad.

3.6. Estrategia de desarrollo de la propuesta

Se crearon diferentes repositorios de datos con los distintos escenarios más comunes a los cuales se enfrentan las organizaciones en el procesamiento de datos, con el objetivo de someterlos a los distintos motores de bases de datos en memoria. De este modo se obtuvo resultados lo más ajustados a la realidad posibles.

Capítulo 4. Análisis del diagnóstico

4.1. Actualidad de las bases de datos en memoria

Actualmente los sistemas de bases de datos relacionales (RDBMS) se encuentran optimizados para realizar toda la gestión de almacenamiento y recuperación de datos accediendo el disco duro de los equipos. Este siempre ha sido uno de los temas que más ha preocupado a los desarrolladores de estos motores, debido a que se hace indispensable recurrir a estrategias que hagan posible un mayor rendimiento independientemente de la volumetría de datos que los motores procesen en cada transacción. Con esto las bases de datos en memoria vienen a proponer una solución al problema mencionado, aprovechando la continua mejora y posibilidad de adquisición de equipos con mayor capacidad de direccionamiento y uso de memoria RAM.

Con respecto a este tipo de bases de datos, se pueden distinguir factores clave que es importante mencionar, con lo cual se pueda dar a conocer su distinción con respecto a rendimiento por sobre los motores tradicionales, a saber:

- **Velocidad:**

Las bases de datos tradicionales han sido diseñadas para los sistemas donde la transaccionalidad y atomicidad de la información es una de las principales situaciones a garantizar, contrario que con las bases de datos en memoria este punto pueda tomarse con menos importancia siempre y cuando se tengan tiempos de respuesta mucho más reducidos y se afecte positivamente el rendimiento.

Se conoce entonces que la velocidad con la cual se accede un disco duro normal promedia en los 5 milisegundos, mientras que el acceso a memoria RAM es de 80 nanosegundos; si se realiza el cálculo correspondiente se puede apreciar una diferencia de hasta 100 mil veces más rápido. Igualmente utilizando discos de estado sólido se puede obtener un rendimiento de hasta 100 veces más veloz. Esto viene a dar una ventaja muy significativa en cuanto a la mejora en la respuesta de los sistemas de bases de datos en

memoria a la hora de realizar consultas y retornar la data para el despliegue al usuario final de la aplicación.

- **Durabilidad:**

Una de las primeras preguntas y dudas con respecto al tema de bases de datos en memoria es qué sucede en el caso de que se corte el suministro de energía al equipo donde se encuentra en ejecución el motor. Es que mientras se tiene la data en memoria existen procesos que escriben la información en almacenamiento no volátil, lo cual garantiza una recuperación al 100% del motor en caso de que se produzca una pérdida de energía al computador.

4.2. OLAP en memoria

La optimización que se puede realizar hoy día en las bases de datos para que tengan un rendimiento adecuado es variada con respecto a las tareas o información que se debe almacenar y recuperar por las aplicaciones (OLTP). El principal problema con este tipo de bases de datos es que no son adecuadas para realizar análisis de datos, por lo cual se emplea otro tipo de estructura multidimensional, la cual permite responder consultas en fracciones de tiempo muy por debajo a las que se obtendrían haciendo uso de OLTP.

Es por lo anterior que no existe una estructura hasta el momento que logre satisfacer las necesidades de OLTP y OLAP juntos, situación que se puede mejorar haciendo uso de las bases de datos en memoria las cuales permiten reducir la creación de cubos ya pre-calculados. De este modo es posible realizar las consultas de manera inmediata, al contar con la posibilidad de poder eliminar completamente el uso de cubos y estructuras de datos almacenadas en discos duros, los cuales son costosos en tiempo para su actualización. Lo que se persigue entonces es poner a disposición del usuario final el consumo de información actualizada en tiempo real, lo que significa trasladar de manera instantánea los datos que son generados en OLTP a OLAP, para ello es necesario reducir drásticamente el tiempo que involucra realizar esta operación, y utilizar las bondades que ofrecen las estructuras en memoria.

4.3. OLTP en memoria

Una de las bases de datos tradicionales más conocidas con el nombre de Microsoft SQL Server propone en su versión de 2012 la integración de una tecnología con el nombre de xVelocity, que viene a plasmar su orientación al manejo y administración de la información en memoria, siendo una base de datos transaccional. Se propone como una mejora al motor VertiPaq incluida en la versión de SQL Server 2008 R2, el cual es un motor que almacena la información en columnas en memoria con el que se obtiene un rendimiento mucho mayor sobre las consultas ejecutadas, haciendo uso de compresión de datos, almacenamiento en caché algoritmos de agregación y detección de datos en paralelo.

xVelocity para su versión de SQL Server 2012 ha sido mejorado para el soporte de consultas de inteligencia de negocios en ambientes corporativos, el cual compone dos escenarios posibles:

- Es posible trabajar utilizando distintos tipos de orígenes de datos, realizar limpiezas sobre los datos, construir reportes y visualizarlos. Esto permite a los usuarios no verse limitados con respecto a la cantidad de registros que pueden ser consumidos por Excel, sino que pueden usar varios millones de registros y obteniendo respuestas en segundos.
- Los desarrolladores de inteligencia de negocios pueden crear proyectos de Analysis Services Tabular teniendo varios orígenes de datos y grandes cantidades de volúmenes de datos administrados en particiones sobre el motor de xVelocity. Cuando el proyecto es publicado en el servidor de Analysis Services, las herramientas como Excel o Power View se comunican con el modelo resolviendo solicitudes en cuestiones de segundos utilizando el motor de xVelocity.

Otro motor de base de datos popular como es Oracle ya ha implementado esta tecnología en su versión 12c, liberada en julio de 2014 con el nombre de Oracle Database In-Memory. Es así como entonces este producto busca cuatro puntos específicos:

- Combinar bases de datos OLTP y OLAP utilizando el mismo producto y eliminar procesos de extracción, transformación y carga cuando sea posible, con lo cual contrarrestar tiempos de carga costosos.
- Contar con información actualizada inmediatamente.
- Reducir el esfuerzo de administración y mantenimiento.

Uno de los aspectos que se menciona sobre esta nueva propuesta de Oracle Database In-Memory es sobre el rendimiento que este ofrece en materia de respuesta a consultas realizadas por los usuarios finales de las aplicaciones, consultas que normalmente tomaban tiempo muy significativo. Con esta nueva tecnología es posible satisfacer esas solicitudes en cuestión de instantes, sin necesidad de comprometer ni restringir ningún tipo de funcionalidad ni configuración; esto debido a que su implementación no requiere ningún tipo de cambio a nivel de aplicaciones finales, ya que los ajustes son propios del motor como tal.

La propuesta entonces radica en que no es necesario mantener toda la base de datos en memoria, ya que puede ser únicamente que se requiera algunas tablas o estructuras que se alojen en ella y que sean puntos clave para aumentar el rendimiento de las aplicaciones que las consumen. De igual manera no es requerido aumentar la memoria del servidor donde reside el motor de Oracle ya que esto mismo es configurable de acuerdo con los requerimientos del cliente y el consumo de data por las aplicaciones. Dentro de las estructuras que pueden ser colocadas en memoria se encuentran las siguientes:

- Tablas
- Particiones
- Subparticiones
- Vistas materializadas

4.4. Consideraciones para implementar bases de datos en memoria

Si se va a realizar una implementación de una base de datos en memoria, ya sea migrarla completamente o fracciones de la misma, entiéndase por esto tablas o vistas materializadas, se debe tomar en cuenta la memoria disponible en el servidor donde se encuentra instalado el motor, tomando como referencia también el crecimiento que puede tener la misma en el transcurso del tiempo utilizando como referencia estimaciones sobre el volumen. Con esto también se menciona la necesidad de realizar un análisis previo de si efectivamente la tabla en discusión va afectar de manera positiva el rendimiento de la aplicación.

Es posible la aplicación y uso de herramientas que ayudan a determinar si la migración de una tabla a memoria tendrá un rendimiento lo suficientemente significativo como para realizar ese proceso de migración. Para el caso de motores de bases de datos como SQL Server, se cuenta con el recopilador de rendimiento de transacciones, con el cual se puede analizar la carga de trabajo e indica qué tablas y procedimientos almacenados obtendrán una mejora de transformar a OLTP en memoria la arquitectura tradicional de la base de datos.

Por otra parte, también SQL Server pone a disposición un asesor de optimización de la memoria, que permite identificar qué aspectos o atributos de la tabla son incompatibles o podrían causar inconvenientes con la migración. Este asesor permite al administrador conocer el impacto y con ello tomar la decisión de realizar la operación de cambio a OLTP en memoria.

Se debe tomar en cuenta que el rendimiento de un motor de bases de datos depende de varios factores, los cuales este asesor de optimización puede ignorar y sobre los cuales es posible que sea necesario un estudio más profundo realizado con otras variables para asegurar una migración mucho más certera.

4.5. Los sistemas sin bases de datos en memoria

Antes de las bases de datos en memoria, los motores de bases de datos transaccionales venían a ser algo así como un repositorio de información, donde solamente se almacenaban y quedaban registrados todas las actividades y procesos a nivel de

transacciones. La implementación de consultas y almacenes de datos para la toma de decisiones implicaba el desarrollo de procesos de extracción, transformación y carga que generalmente corren mediante procesos nocturnos o en momentos donde las aplicaciones no estén consumiendo en su mayor parte los servicios ofrecidos por el motor de base de datos, para no degradar el rendimiento de las aplicaciones.

Actualizar grandes volúmenes de información en bases de datos muy grandes es todo un desafío, ya que implica un costo en procesamiento y tiempo por lo general muy alto. Muchas empresas ya han experimentado muchos de estos problemas descritos. Ante la necesidad de encontrar soluciones alternas, han decidido implementar un modelo de sistemas con base en aplicaciones que consuman los servicios que ofrecen actualmente los motores de bases de datos en memoria, y se obtiene así resultados satisfactorios acordes con las necesidades.

Las bases de datos en memoria han venido a agilizar los procesos de negocios. Las grandes empresas buscan cada vez más apoyo de las tecnologías de la información para sacar mayor ventaja competitiva, está claro que cuanto más rápido se tenga la información a mano, más rápido se podrán analizar los resultados para tomar decisiones que a la postre hasta lleven a la determinación de cambiar el rumbo estratégico con el que venía trabajando la compañía.

Antes los tiempos de reacción eran mayores, por lo que las grandes empresas definían estrategias de mercado muy diferentes a las actuales. La necesidad de competir y sacar ventaja ante los rivales, así como solventar problemas muy particulares, es lo que ha llevado a las bases de datos a evolucionar ante un mercado tan cambiante a nivel mundial. Ahora todo se necesita de inmediato, las grandes compañías requieren procesar y ver información casi que en tiempo real. Es por ello que las bases de datos en memoria se han convertido en un excelente aliado para la optimización de procesos a nivel computacional, pues permiten a las empresas obtener información procesada en tiempos muy pequeños, si se compara con el desempeño que normalmente tienen los motores de bases de datos tradicionales, basados en el almacenamiento primario en disco duro.

4.6. ¿Quiénes utilizan bases de datos en memoria?

Grandes compañías alrededor del mundo han implementado y están utilizando arquitecturas de bases de datos en memoria, lo interesante es que la técnica ha sido aplicada tanto a modelos de bases de datos relacionales como no relacionales. La necesidad de implementar este tipo de tecnologías de la información, ha tenido una constante común en la mayoría de las diferentes compañías que lo han hecho, la cual radica en la necesidad de obtener una ventaja con respecto a la velocidad con que se obtiene y se procesa la información.

Dentro de las compañías que más utilizan bases de datos en memoria se encuentran las de redes de telecomunicaciones, que utilizan aplicaciones empresariales, software y sistemas embebidos. Dentro de algunos líderes de mercado que las utilizan, se encuentran los siguientes:













Se continuará ahora compartiendo una vivencia que experimentó una de las compañías mencionadas, la cual enriquece el tema tratado en este proyecto, de acuerdo con Wolter (2016) en el relato de experiencia vivida con la base de datos mongoDB.

OTTO es la principal tienda en línea de venta al por menor en Alemania, para artículos de moda y estilos de vida. El sitio anterior de OTTO era estático, lento y caro de cambiar. OTTO tenía un sistema grande con un gran equipo, solamente pequeños fragmentos del sitio se podían modificar para personalizar la experiencia del cliente. Temas como la fluctuación de precios competitivos eran críticos y no tenían la capacidad de obtener los precios más relevantes de cara al cliente, por lo que el tiempo de reacción era una sentencia a muerte.

Con el modelo de datos no relacional que posee mongoDB se eliminó esa complejidad, OTTO solo tiene de uno a dos segundos para acceder a los perfiles de los clientes en tiempo real y decidir cómo reaccionar. Adicionalmente, las actualizaciones en listas de precios de los artículos que antes tardaban alrededor de 12 horas, ahora se realizan en tan solo 15 minutos. Gracias a mongoDB, ahora OTTO puede moverse muy rápido dentro de la memoria y la capacidad en disco ha llevado a mejorar significativamente los tiempos de respuestas.

De acuerdo con lo investigado para el desarrollo de este proyecto, no se encontró un caso real en Costa Rica de alguna empresa que utilice las bases de datos en memoria para solventar problemas de desempeño o mejorar los tiempos de respuestas de sus aplicaciones. Sin embargo el caso de OTTO viene a dar un buen precedente del tipo de problemas que se pueden solucionar en sistemas tanto transaccionales como de consultas, con la finalidad de mejorar los tiempos de reacción ante un mercado tan competitivo y cambiante como en el que se vive actualmente.

4.7. La selección de los motores de bases de datos en memoria

En este proyecto se investigó acerca de varios motores de bases de datos en memoria que se ofrecen en la actualidad, muchos de los cuales son propietarios, sin embargo, también hay de tipo software libre. Algunos de los propietarios ofrecen versiones de pruebas que pueden ir desde los treinta hasta los noventa días.

Debido a la naturaleza de este proyecto, la idea principal fue analizar algunos de los motores de bases de datos seleccionados y observar su comportamiento con respecto a la eficiencia en que se ejecutaban las consultas para evaluar el rendimiento y la capacidad de devolver la información solicitada, basados en el tiempo de respuesta que tardaba el motor en mostrar la información solicitada. En principio muchos de los motores seleccionados, parecían atractivos para este proyecto, sin embargo, al investigar más sobre las bondades y características propias de cada uno de ellos, se llegó a determinar que muchos fueron desarrollados para solucionar problemas de alguna situación muy particular, por lo que, para el desarrollo de este proyecto, no aportaban mayor beneficio. Por otra parte, otros motores sí se ajustaban para poder utilizarlos y realizar evaluaciones de ellos, acorde con la idea principal que se desarrolló en este proyecto, por lo que se incursionó más en la investigación acerca del funcionamiento y la arquitectura de algunos de ellos, para entender su funcionamiento y sacar provecho de las propiedades que ofrecían.

Sin embargo, aun teniendo claro cuáles de los motores seleccionados cumplían con la naturaleza de este proyecto, venía la primera etapa, la cual consistía en entender el funcionamiento y preparar los ambientes en que corre cada motor, realizar las configuraciones e instalación. Algunos de los motores seleccionados fueron muy complejos de instalar, y otro tanto más compleja fue la carga de datos y ejecución de consultas, por lo que en principio se descartó su uso para este proyecto, ya que estaba tomando demasiado tiempo la implementación y puesta en marcha, para poder utilizarlos. Otros motores eran menos complejos de implementar, ya que para realizar cargas de información y ejecución de consultas requerían de menos programación para lograrlo, por lo que se eligió trabajar con este tipo de motores de bases de datos.

Dado que el propósito de este proyecto fue evaluar el rendimiento de los motores de bases de datos, para dar al lector una idea de cómo iniciar con una implementación de esta naturaleza, no quisimos incursionar con motores de bases de datos en memoria complejos de configurar e instalar, a fin de transmitir al lector una idea clara para que pueda tomar una decisión acerca de qué tipo de arquitectura en memoria podría eventualmente implementar para dar solución a un problema muy particular que tenga. Al principio se tuvo malas experiencias con motores complejos de implementar, como por ejemplo, Tarantool o MonetDB, que utilizan un API para la ejecución de consultas las cuales se deben programar en lenguajes específicos, como Lua en el caso de Tarantool, para luego compilar el programa para poder ejecutarlo. Es por ello que este tipo de motores de bases de datos en memoria se descartaron para este proyecto.

Capítulo 5. Propuesta de solución

En este proyecto se instalaron y configuraron algunos motores de bases de datos para luego cargar un volumen de datos representativo que permitiera realizar consultas y evaluar el rendimiento de cada uno de ellos. A continuación, se explican los detalles respecto a la instalación y configuración de los motores de bases de datos en memoria evaluados.

5.1. SolidDB

SolidDB es un producto de IBM, es de tipo relacional con todas las funciones en memoria, además utiliza el lenguaje estándar de SQL, es multiplataforma. Dentro de los sistemas operativos en que corre se encuentran:

AIX: (Advanced Interactive eXecutive), un sistema operativo Unix propiedad de IBM.

HP-UX: También Unix desarrollado y soportado por Hewlett-Packard.

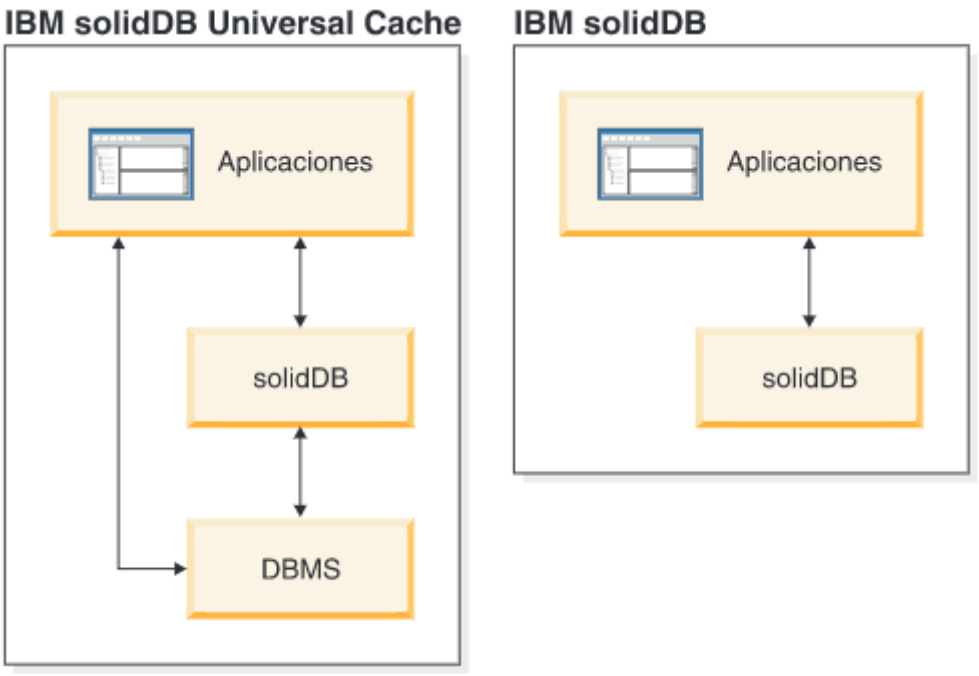
Linux: De tipo software libre, desarrollado en lenguaje C.

Windows: Desarrollado por Microsoft.

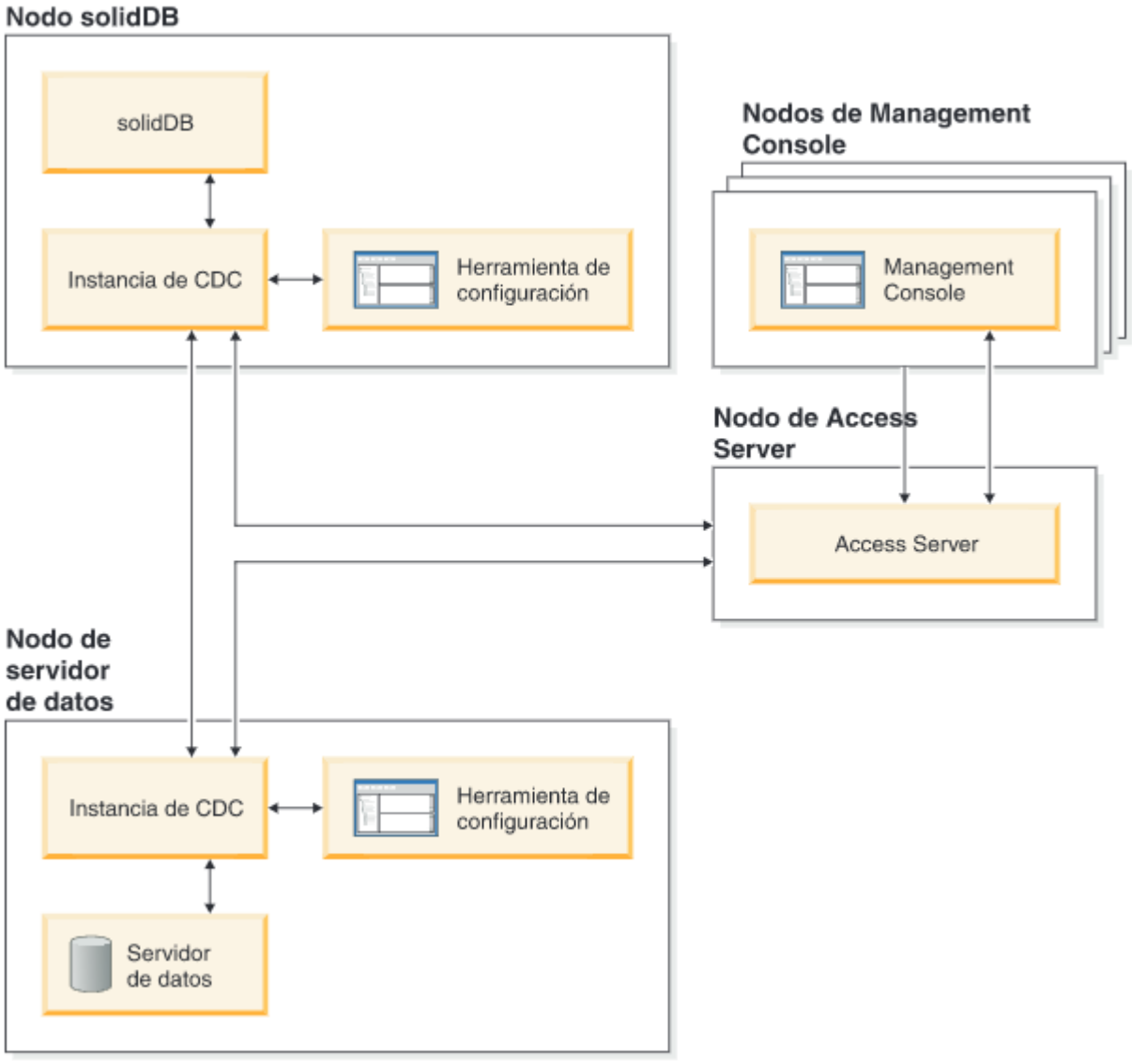
Solaris: Desarrollado por Sun Microsystems, actualmente propiedad de Oracle Corporation.

5.1.1. Arquitectura de SolidDB

Básicamente SolidDB se compone de dos productos: IBM SolidDB Universal caché y IBM SolidDB. El primero es un software de memoria caché, en memoria relacional, y el segundo es una base de datos en memoria relacional, en donde se pueden tener tablas en memoria o en disco duro sobre la misma instancia. A continuación se muestra una visión general de la arquitectura de SolidDB.


IBM solidDB Universal Cache
Aplicaciones
solidDB
DBMS
IBM solidDB
Aplicaciones
solidDB

Seguidamente se ilustra de una manera muy general la arquitectura de SolidDB Universal Cache de una configuración típica y sus componentes principales:



Ahora se va a describir a grandes rasgos un poco acerca de cada uno de los componentes que conforma la arquitectura de SolidDB.

SolidDB: Es la base de datos front-end, y se encarga de duplicar datos desde y hacia. Puede ser la base de datos origen o destino, o ambas, dependiendo de la configuración.

Servidor de datos: Es la base de datos back-end, y se encarga de duplicar datos desde y hacia. Puede ser la base de datos origen o destino, o ambas, dependiendo de la configuración.

Instancia de InfoSphere CDC: Es una instancia de tiempo de ejecución del motor de InfoSphere CDC para un DBMS determinado.

Herramienta de configuración: Es una consola GUI que se utiliza para configurar y crear instancias de InfoSphere CDC.

Access Server: Es un proceso que permite a los usuarios de Management Console acceder a instancias de InfoSphere CDC y las configuren.

Management Console: Aplicación interactiva con una GUI y se puede utilizar para configurar y supervisar las réplicas de los datos.

- **Características:**
  1. Modelo Cliente/Servidor.
  2. Soporte simultáneo para varios protocolos de red y tipos de red.
  3. Drivers de conexión ODBC y JDBC.
  4. Tablas en memoria.
  5. Tablas basadas en disco.
  6. Características avanzadas como triggers y procedimientos almacenados.
  7. Acceso a la memoria compartida (SMA) y a la biblioteca enlazada (LLA).
  8. Replicación de datos.
  9. Alta disponibilidad (HotStandby).

- **Ventajas:**
  1. Cuando las tablas residen en memoria se tiene baja latencia de consultas y alto rendimiento.
  2. Las tablas a nivel de base de datos son configurables, dependiendo de la necesidad se pueden parametrizar para que sean en memoria, para que sean persistentes o transitorias.
  3. Las consultas pueden abarcar tanto tablas en disco como también en memoria, mediante las operaciones de (unión o join) estándar de SQL.

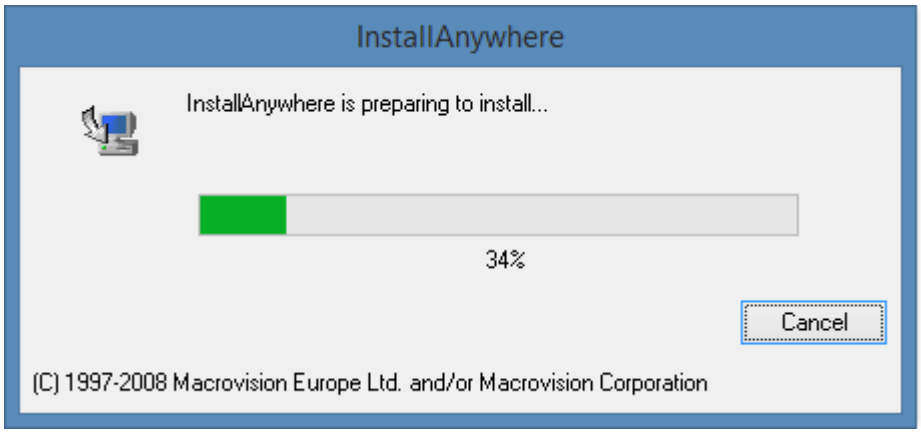
4. Gracias a la característica de acceso a la memoria compartida, es posible enlazar aplicaciones con el servidor de SolidDB directamente, sin necesidad de establecer comunicación a través de protocolos de red como TCP/IP.
5. La replicación de los datos puede realizarse con base en tres tecnologías de SolidDB de tres formas: Réplica avanzada, Réplica de InfoSphere CDC y Alta disponibilidad (HotStanby)

5.1.2. Instalación de SolidDB

IBM cuenta con varios manuales en línea acerca de SolidDB, para este proyecto se trabajó puntualmente con la versión IBM SolidDB 6.5. Por tanto, lo que se describe a continuación, relacionado con el proceso de instalación, así como algunos inconvenientes que se tuvieron al momento de instalarlo, está relacionado puntualmente a la versión descrita en este documento. IBM cuenta con un sitio para la descarga de SolidDB y todos sus componentes, para cada uno de los sistemas operativos a los que el motor de base de datos soporta.

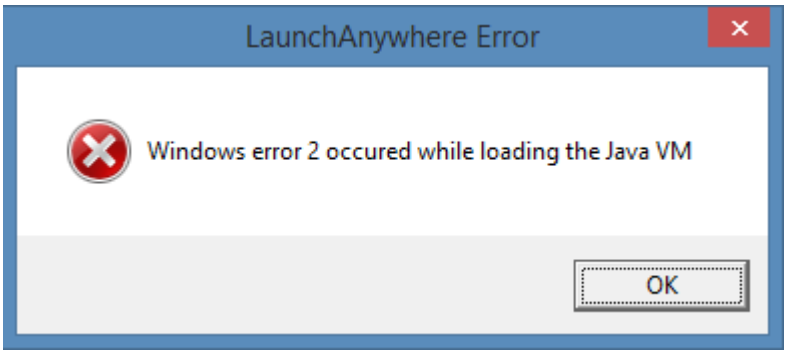
Para efectos de este proyecto, y la ejecución de las pruebas necesarias, tanto de cargas masivas de datos, así como para los análisis, se utilizó la versión de SolidDB para Microsoft Windows en la versión de 64 bits. Esta versión utiliza la plataforma virtual de Java, basada en la arquitectura descrita por Oracle en Java Platform Standard Edition 8 Documentation (2016). Es por ello requerido tener instalada la máquina virtual de Java.

Luego de descargar el instalador, se procedió a realizar la instalación, con un usuario de tipo administrador del equipo. IBM por lo general proporciona dentro de sus paquetes de instalación un programa interactivo que cumple la función de guiar al usuario durante el proceso de instalación, el cual se puede ejecutar con doble clic desde la interface gráfica, o desde una consola de comandos. A continuación se ilustra la forma gráfica:



Sin embargo, se debe tener en consideración que, para esta versión en particular que se instaló, es común encontrar un error de ejecución, relacionado a la variable de entorno, PATH de Java, propia de la configuración.

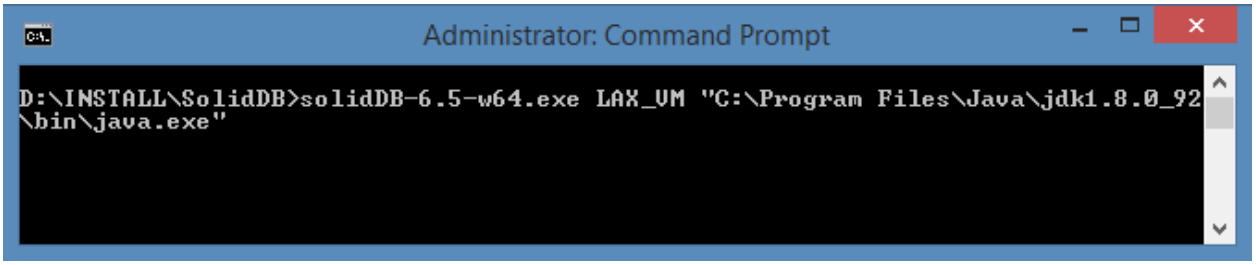
Lo que sucede con la versión de este instalador de SolidDB, es que el programa interactivo InstallAnywhere utiliza una ubicación de Java específica para la variable de entorno PATH, específica en la ruta "C:\ProgramData\Oracle\Java\javapath", la cual no coincide con la ruta PATH creada al momento de instalar la versión de la máquina virtual de Java. Al estar la máquina virtual de Java instalada en una ruta diferente a la predefinida por InstallAnywhere, se genera un error al momento de intentar cargar la máquina virtual de Java, durante la ejecución del instalador de SolidDB. El error es tal como se muestra en la siguiente figura:

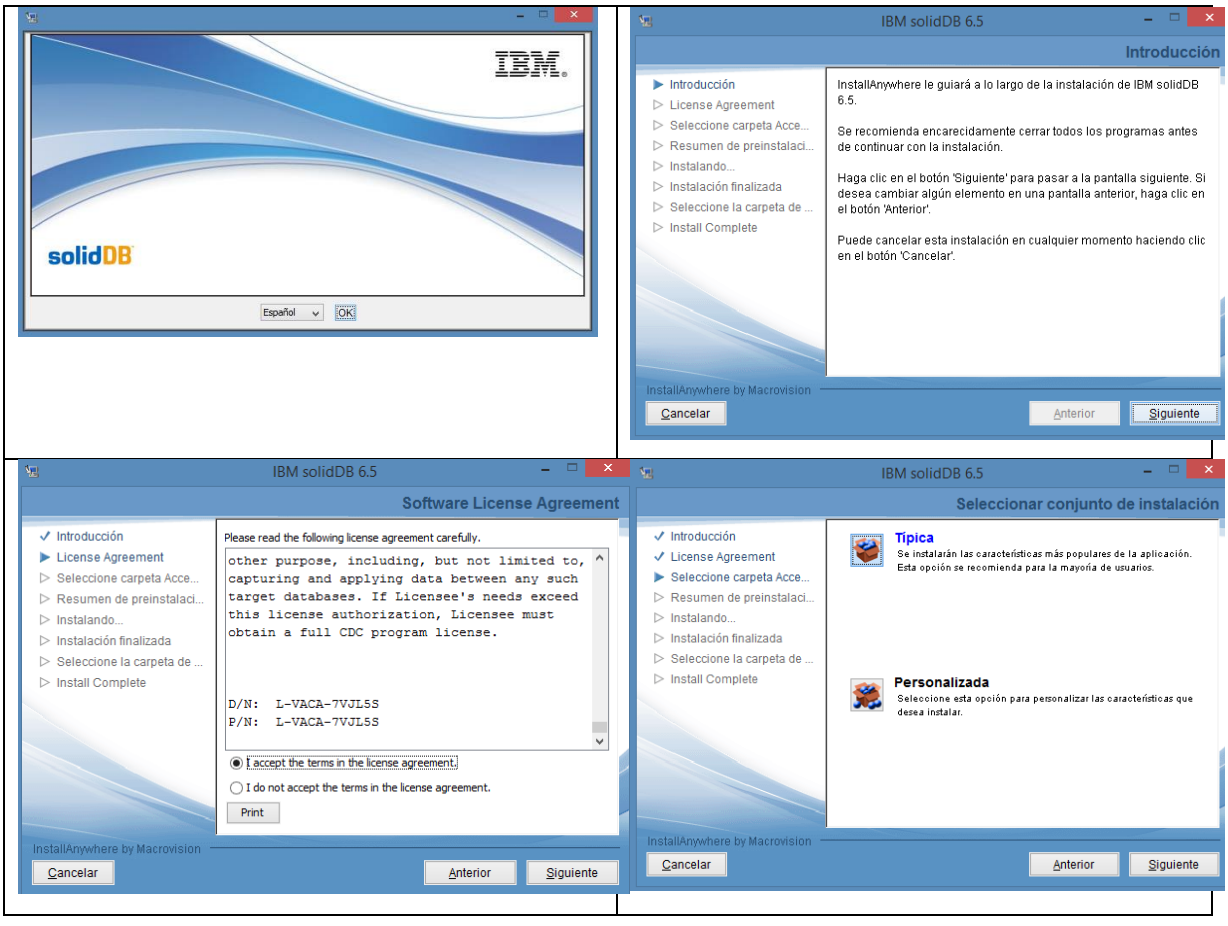


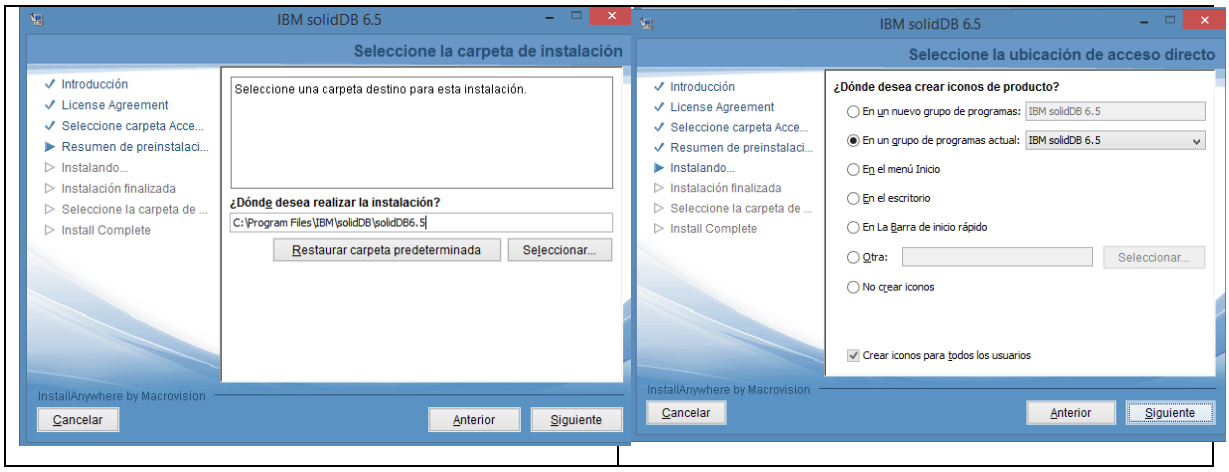
La solución al problema fue bastante simple, en vez de usar la instalación gráfica, se utilizó la instalación a través de la línea de comandos, por medio de esta se pudo especificar mediante el comando "LAX_VM" la ubicación física del directorio donde se encontraba el

programa “java.exe”, de acuerdo con la versión de la máquina virtual instalada. Burch (2016) en el Workaround explica cómo realizar la instalación por línea de comandos. A continuación, se muestra la ejecución del comando desde el Command Prompt de Windows como usuario administrador del equipo.

D:\INSTALL\SolidDB>solidDB-6.5-w64.exe LAX_VM "C:\Program Files\Java\jdk1.8.0_92\bin\java.exe"







Una vez instalado el paquete, queda instalado el motor de base de datos SolidDB, una consola de administración llamada “solsql SQL Editor” para administrar tareas propias relacionadas con el motor de base de datos, tales como creación y modificación de objetos, creación de usuarios, desarrollo de consultas.

Otra herramienta importante integrada dentro del paquete de instalación es “solload”, de mucha utilidad para realizar cargar masivas de datos, lo cual permite importar datos hacia una tabla dentro de la base de datos de SolidDB.

5.1.3. Configuración de SolidDB

Cuando se instala SolidDB, se crea un directorio con un archivo de configuración llamado “solid.ini”. En este proyecto se utilizó una versión de evaluación del producto, por lo que dicho archivo de configuración estaba ubicado en el siguiente directorio: C:\Program Files\IBM\solidDB\solidDB6.5\eval_kit\standalone.

Es importante tener en consideración que estos parámetros pueden ayudar a mejorar el rendimiento del motor de base de datos, pero por otra parte, si los parámetros se configuran mal, el rendimiento se reducirá considerablemente. Muchos de los parámetros tienen que ver con las características propias de la memoria, como por ejemplo, si se desea que al crear las tablas, estas queden almacenadas por defecto en la memoria, por lo tanto, es indispensable prestar atención al consumo de la memoria.

Para efectos de simplificar las pruebas realizadas, se limitó a configurar solamente las variables relacionadas con la administración de la memoria, ya que la idea acá siempre fue probar las funciones de almacenamiento en memoria y documentar el rendimiento para su posterior análisis. Es importante tener en consideración que los cambios que se le realicen al archivo de configuración "solid.ini" no tendrán efecto alguno hasta la próxima vez que inicie el servidor de SolidDB, por lo que se recomienda reiniciar el servicio de SolidDB para que la nueva configuración se aplique.

Los parámetros relacionados con la memoria se encuentran en la sección [MME] dentro del archivo de configuración. De acuerdo con MME section (2016), a continuación se describen los más relevantes:

Sección [MME]	**Descripción**	**Valor de fábrica**	**Modo acceso**
ImdbMemoryLimit	Especifica el límite máximo que utiliza el servidor para almacenar tablas e índices en memoria. Valor en cero "0" significa sin límite. ImdbMemoryLimit=1073741824 ImdbMemoryLimit=1048576kb ImdbMemoryLimit=1024MB ImdbMemoryLimit=1GB	0 Unidad: 1 byte k=KB m=MB g=GB	RW
ImdbMemoryLow Percentage	Indica qué porcentaje de memoria se puede utilizar, antes de que el servidor limite su capacidad de inserción sobre las tablas almacenadas en memoria. Los valores permitidos son entre 60% y 99%	90	RW
ImdbMemoryWarn ingPercentage	Establece un límite de advertencia para el tamaño de la memoria. Se	80	RW

	debe expresar como un porcentaje del parámetro ImdbMemoryLimit. Este parámetro se aplica solo sobre la memoria principal de SolidDB. No se toma en cuenta el almacenamiento en disco.		
LockEscalationEnabled	Activa o desactiva el bloqueo de tablas. Mientras un usuario modifica una tabla, otros son incapaces de usar esa misma tabla, incluso si desean utilizar diferentes filas.	no	RW/StartUp
LockEscalationLimit	Si LockEscalationEnabled se establece en “si”, entonces este parámetro indica la cantidad de filas que deben ser bloqueadas en una tabla, antes de que el servidor escale el nivel de bloqueo de fila a nivel de tabla.	1000	RW/StartUp
MaxCacheUsage	Limita la cantidad de memoria caché utilizada para una tabla almacenada en disco. El Valor MaxCacheUsage = 0 significa ilimitado, por lo que el uso de la caché es IndexFile.CacheSize / 2.	8MB	RW/StartUp
MaxTransactionSize	Define el tamaño máximo apropiado para una transacción en Bytes. Si el tamaño de la transacción supera el valor establecido, la operación falla con el error de base de datos de solidDB 16509, transacción MME, tamaño máximo excedido. Valor 0 es	0	RW

	ilimitado.		
ReleaseMemoryAtShutdown	Al establecer este valor en “yes” cuando se cierra el sistema, el servidor libera la memoria utilizada por las tablas almacenadas en memoria.	no	RW/StartUp
RestoreThreads	Este parámetro define la cantidad de hilos máximos utilizados, mientras se restaura la base de datos en memoria, al momento de inicializar la base de datos. Posibles valores entre 1 y 65536. Por defecto se toma el valor del parámetro “General.MultiprocessingLevel”.	Igual que: General.MultiprocessingLevel	RW/StartUp

5.2. H2

Es una propuesta de base de datos en memoria iniciada en mayo de 2004, su primer producto publicado fue en diciembre de 2005 como una versión estable para su utilización. Su creador fue Thomas Mueller, quien también es el creador de Hypersonic SQL en 2001. Esta base de datos dio origen a su versión de Hypersonic 2, con la cual dio paso a la creación de H2, que a pesar de su creador, no comparte código con sus orígenes. La principal característica de H2 es que se encuentra desarrollada en JAVA, y esto se debe a lo siguiente:

- El uso de JAVA permite integrar las aplicaciones de manera muy sencilla y rápida.
- Es soportado por una mayor cantidad de plataformas.
- Las funciones y los triggers son ejecutados de manera muy rápida.

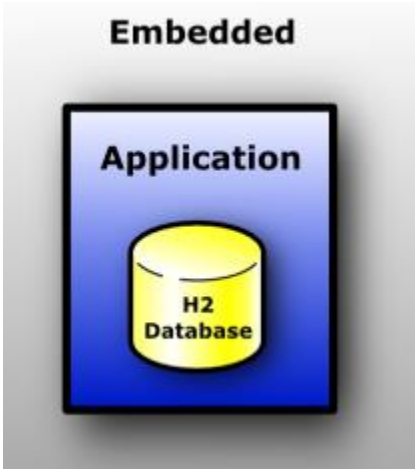
Los desarrolladores de H2 mencionan la rapidez con que se desarrolla en JAVA comparado con C o C++, además de la portabilidad que tiene a diferentes plataformas. Se logra así una mayor utilización de la herramienta, al no depender directamente de un entorno específico de servidores ya disponibles por las compañías.

5.2.1. Arquitectura

La arquitectura en la cual puede operar la base de datos H2 puede ser comprendida en tres modos:

- **Modo embebido**

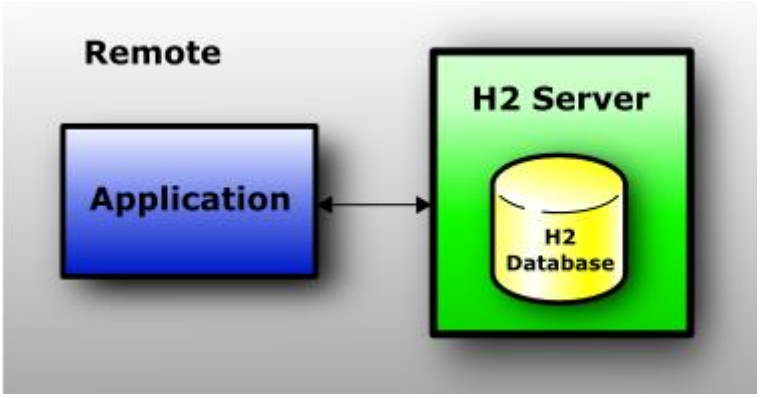
En este modo tanto la aplicación como el motor de base de datos se encuentra en el mismo servidor de máquina virtual de JAVA, donde la conexión se realiza utilizando un JDBC. Este modo permite que sea el más fácil y rápido de implementar, pero la desventaja es que la base de datos solamente se puede abrir en una máquina virtual. En este, como en todos los modos siguientes, tanto los datos en memoria como los persistentes son soportados. A continuación la ilustración de este modo:



- **Modo servidor**

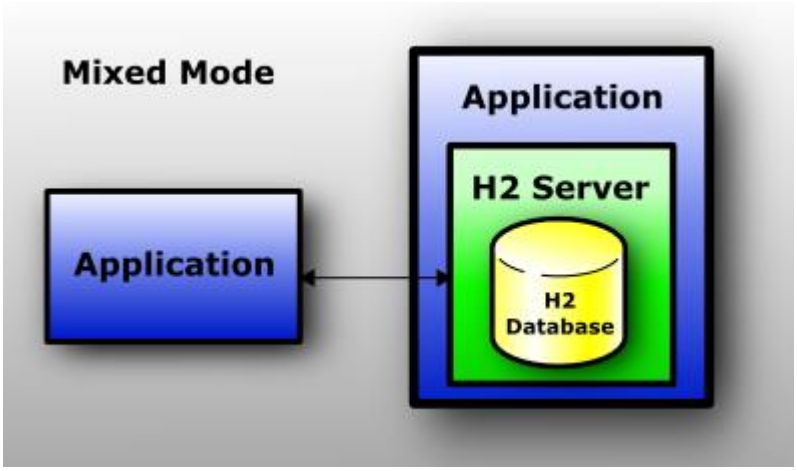
Este modo es el conocido como cliente/servidor, donde el cliente se comunica con el motor remotamente utilizando un JDBC que se conecta mediante TCP/IP, esto permite que se pueda tener varios clientes conectados a la base de datos. De igual manera el

procesamiento de las consultas y demás se realizan internamente. Este modo es más lento que el anterior mencionado, ya que la data debe ser enviada por la red desde y hacia el cliente. La ventaja es que no existe cantidad de conexiones abiertas.



- **Modo combinado**

Como su nombre lo indica es una combinación del modo embebido y cliente/servidor que se mencionaron, de modo que la primera aplicación que hace uso del motor de base de datos lo realiza en el servidor, y se inician procesos para que respondan a las solicitudes que hagan otras aplicaciones fuera del él. Sin embargo, las solicitudes de la aplicación embebida son atendidas y resueltas de manera más rápida, mientras que las conexiones que se realizan de manera remota son más lentas.



5.2.2. Características

Dentro de las principales características de H2 se encuentran las siguientes:

- Herramienta muy rápida de código abierto y uso de Java Data Base Conectivity API.
- Consola de aplicación por medio de buscador web.
- Soporte de SQL estándar.
- Características de seguridad fuertes.
- Puede ejecutarse utilizando versiones distintas de manera concurrente sin producirse ningún problema.
- Puede operar utilizando memoria o acceso a disco duro, aunque sea para el almacenamiento de tablas.
- Manejo de transacciones.
- Soporte para múltiples conexiones.

5.2.3. Instalación

5.2.3.1. Requerimientos

La instalación de este motor de base de datos requiere que se tenga un equipo con las siguientes características mínimas:

- Windows XP o Vista, Mac OS X, Linux
- Java 6 o superior
- Sistema de archivos en NTFS o FAT32

5.2.3.2. Consola de H2

Es necesario contar con un navegador web para acceder a la consola del motor, la cual permite la ejecución de las sentencias SQL.

5.2.3.3. Plataformas Soportadas

H2 es un motor de base de datos desarrollada en JAVA, lo cual permite que pueda ser ejecutado e instalado sobre gran variedad de plataformas. Según el desarrollador ha sido probado utilizando Java 6 y sobre las plataformas de Windows 8 y Mac OS. Sin embargo en otros sistemas operativos que ejecuten Java, como son Windows XP, Windows Vista, Windows 7, Ubuntu y otros, puede ser instalado sin problema alguno.

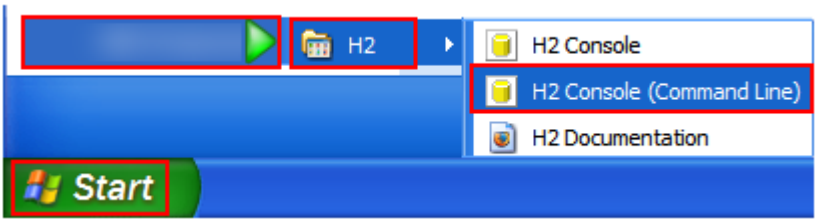
5.2.3.4. Instalación de H2

La instalación del motor es el paso más sencillo, ya que únicamente se debe descargar el paquete de instalación y ejecutarlo seleccionando la ruta o carpeta donde van a ser depositados los archivos necesarios para H2.

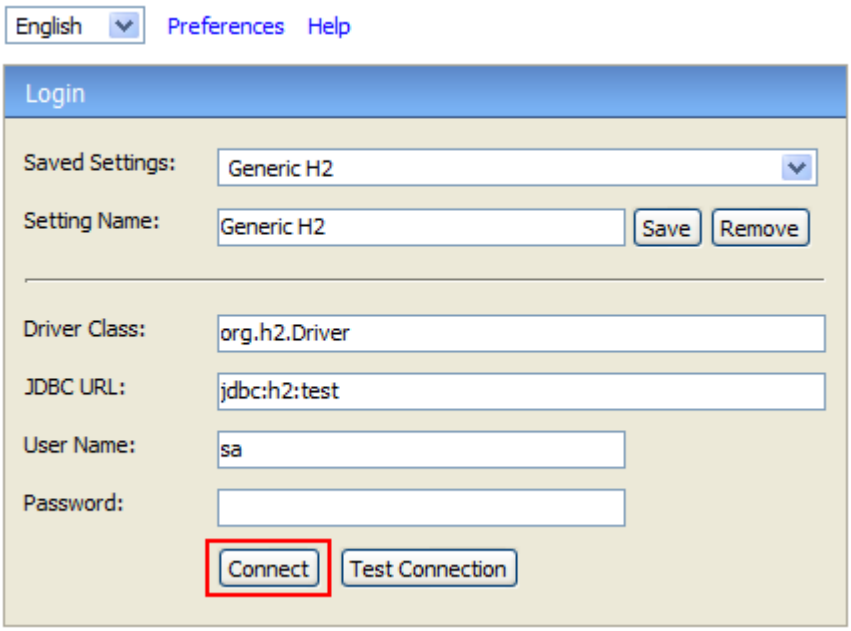
Una vez instalado se debe apreciar una estructura de archivos como la que se muestra a continuación:

Carpeta	Descripción
bin	Archivos binarios
docs	Documentación
docs/html	Páginas HTML
docs/javadoc	Archivos de JAVA
ext	Dependencias externas
service	Herramientas para los servicios
src	Archivos de origen
src/docsrc	Documentación de archivos de origen
src/installer	Instaladores y scripts
src/main	Código del motor de base de datos
src/test	Código de prueba
src/tools	Herramientas para comunicación y conexión a la base de datos.

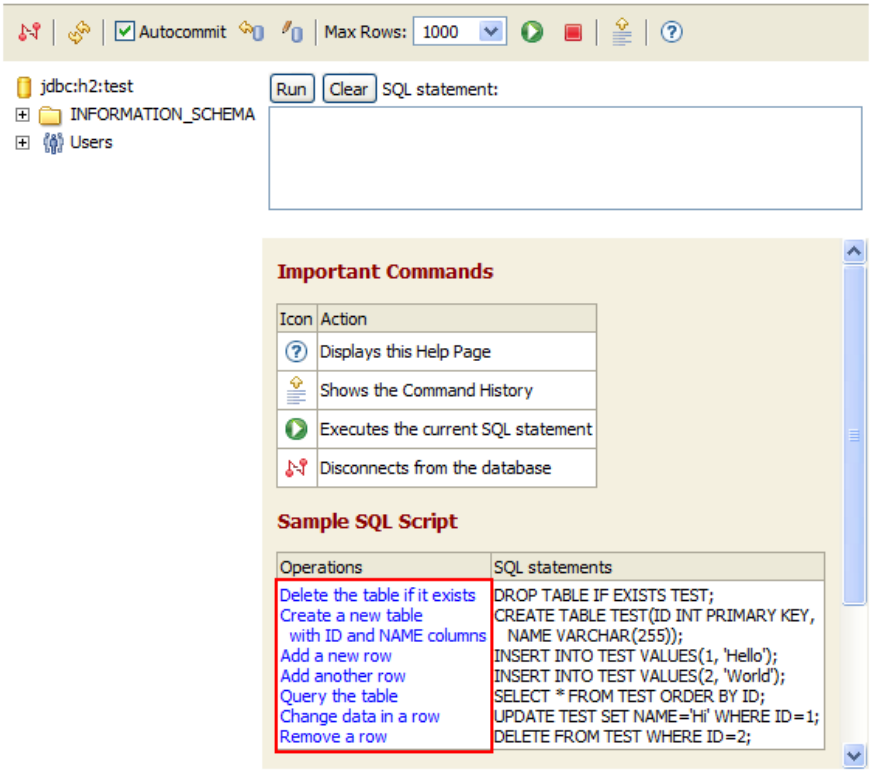
Para acceder al motor de H2 simplemente se debe iniciar la consola del mismo haciendo clic en Inicio, Todos los programas, H2 y finalmente H2 Console (Command Line). Ello se muestra en la figura adjunta:



Una vez ejecutada la utilidad se debe abrir un explorador web con la ruta siguiente: http://localhost:8082, con la cual se puede ingresar al ambiente gráfico del motor. Este ambiente se muestra de la siguiente manera:



De esta manera se puede observar la utilidad donde se realiza toda la gestión de la base de datos y la ejecución de sentencias SQL, como se muestra en la figura siguiente:



5.3. Derby DB

Derby DB es un subproyecto de la Fundación de Software de Apache. Su primera versión hace su aparición en 1997 bajo el nombre de JBMS, pero fue hasta el año 2004 cuando fue Apache adquirió los derechos del código fuente y lo llamó Derby DB. Es una base de datos relacional de código abierto. Dentro de las principales características que posee este motor están las siguientes:

- Sus archivos para ejecución son muy livianos, aproximadamente 2.6 megabytes para el motor y el controlador JAVA.
- Está enteramente basado en JAVA.
- Provee ensamblados que permiten utilizarla de manera embebida en aplicaciones desarrolladas en JAVA.
- Puede ser configurada en una arquitectura cliente servidor.
- Es fácil de instalar, compilar y usar.

5.3.1. Arquitectura

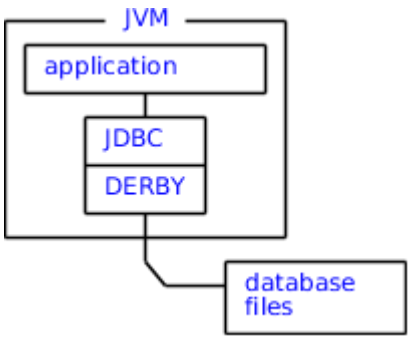
De acuerdo con Apache Derby (2016), básicamente Derby DB pone a disposición tres utilidades con las cuales se puede gestionar la base de datos:

- ij: Que es una especie de compilador de scripts y sentencias SQL que serán ejecutadas por el motor.
- dblock: Es una utilidad para extracción de datos del motor.
- sysinfo: Es una utilidad para desplegar versiones y rutas a los archivos del sistema.

Derby DB utiliza los estándares de SQL, por lo que provee la funcionalidad esperada de todo motor de base de datos tradicional. Esto es tanto a nivel de sentencias SQL, gestión de transacciones, concurrencia, triggers y respaldos. Todo esto permite que la migración hacia y desde este motor sea totalmente transparente para todo desarrollador de aplicaciones si este tiene a bien cambiar de motor.

De acuerdo con Embedded Derby (2016), cuando una aplicación tiene acceso a DerbyDB mediante el controlador JDBC embebido el motor no se ejecuta en un proceso separado, por lo tanto, no hay procesos independientes de la base de datos para iniciar y detener. En vez de procesos separados, el motor de DerbyDB se ejecuta bajo un mismo proceso padre dentro de la misma máquina virtual de Java (JVM) que está ejecutando la aplicación. De esta manera Derby se convierte en una solicitud más, al igual que cualquier otro archivo JAR utilizado por la aplicación.

Seguidamente se muestra la arquitectura de DerbyDB:



5.3.2. Características

Dentro de las principales características de DerbyDB se encuentran las siguientes:

- Arquitectura cliente servidor.
- Seguridad de autenticación de usuarios.
- Autenticación LDAP.
- Soporte de SQL estándar.
- Almacenamiento de datos en memoria.

5.3.3. Requerimientos

Dado que Derby es un motor de base de datos escrito completamente en Java, es capaz de correr en cualquier máquina virtual de Java que esté certificada, los requerimientos son los siguientes:

- Tener instalado el ambiente virtual de Java (JDK), versión 1.4 o superior.
- Tener instalada y configurada la carpeta de binarios de Apache Derby.
- Tener instalada una consola de comandos del sistema operativo.

5.3.4. Instalación

Para la instalación de Derby DB se debe seguir una serie de pasos, con los cuales se puede obtener una versión lista para su uso en un ambiente controlado. De esta manera, se debe seguir las actividades siguientes:

Se debe verificar la existencia del Java Development Kit (JDK), ya que es requerida una versión 2 o posterior del mismo. Si no se cuenta con la instalación del mismo se debe descargar y proceder como corresponde para ejecutar satisfactoriamente Derby DB. Una vez instalado se debe configurar la variable de ambiente JAVA_HOME a la raíz de la carpeta de instalación del JDK, para ello se abre una consola de ***command prompt,*** y se digita la siguiente sentencia como se muestra en seguida:

C:\> set JAVA_HOME=C:\jdk1.7

Una vez realizado lo anterior, es tiempo de descargar el paquete de instalación de Derby DB ubicado en el sitio: http://db.apache.org/derby/derby_downloads.html. Es necesario descargar la versión estable, ya que en algunas ocasiones las versiones betas pueden tener errores que afecten algunas funcionalidades del producto.

Para este proyecto se utilizó la última versión estable que había disponible en su momento (db-derby-10.10.2.0). Una vez descargado el archivo Zip de la versión, se creó una carpeta llamada “Apache” en la raíz del disco duro C:\ de la siguiente manera: C:\Apache, se copió dentro el archivo comprimido “db-derby-10.10.2.0-bin.zip” y se descomprimió para armar toda la estructura de directorios de archivos, propia del motor de base de datos. Queda entonces de la siguiente manera: “C:\Apache\db-derby-10.10.2.0-bin”

Una vez extraído el paquete de instalación hay que configurar la variable de entorno de instalación para DerbyDB, a través del command prompt de Windows con el siguiente comando:

set DERBY_INSTALL=C:\Apache\db-derby-10.10.2.0-bin

Luego con el Command Prompt, se debe posicionar en el directorio bin de la siguiente manera y ejecutar los siguientes pasos de configuración:

Command Prompt -> cd C:\Apache\db-derby-10.10.2.0-bin\bin

Ejecutar el archivo “setEmbeddedCP.bat” que se encuentra dentro del directorio bin

Command Prompt -> java org.apache.derby.tools.sysinfo

Command Prompt -> java org.apache.derby.tools.ij

En este punto ya debe estar iniciado el compilador de scripts del DerbyDB, ahora hay que conectarse a la base de datos, mediante el siguiente comando:

ij> connect 'jdbc:derby:memory:MyDbTest;create=true';

A partir de este punto ya el motor estará iniciado y listo para recibir código del tipo SQL, con el cual se pueden crear tablas y realizar consultas a la base de datos.

5.3.5. Consola del motor

Derby trabaja amparado totalmente a la máquina virtual de Java, para acceder al motor se necesita una consola de comandos, como por ejemplo el Command Prompt de Windows. En la consola se puede iniciar el motor de base de datos y se ejecuta el intérprete de sentencias SQL. El intérprete se encarga de interactuar con el motor de base de datos, realizando las tareas solicitadas a través de la línea de comandos.

5.3.6. Plataformas soportadas

Dado que Derby es un motor de base de datos escrito completamente en Java, es capaz de correr en cualquier máquina virtual de Java certificada.

5.4. SQLite

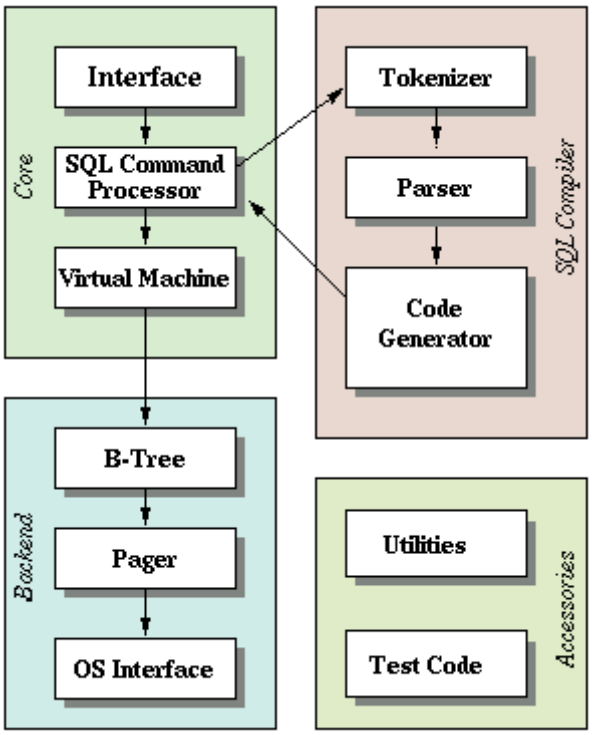
De acuerdo con About SQLite (2006), SQLite es una biblioteca compacta que contiene un motor de base de datos autónomo. El código de SQLite es de dominio público, por lo que es libre para el uso de cualquier propósito, ya sea comercial o privado. A diferencia de la mayoría de motores SQLite, no posee un servidor de procesos separados. El proyecto SQLite se inició a finales del año 2000, con la intención de apoyar los procesos de tomas de decisiones.

5.4.1. Arquitectura

Según Architecture of SQLite (2006), el motor funciona a través de la compilación de texto SQL haciendo uso de la máquina virtual de Java. Dentro de la arquitectura del motor existe un programa que actúa como un compilador para la conversión de texto SQL a código de bytes.

SQLite tiene una interface de lenguaje C, en la que se encuentran los archivos de origen main.c, legacy.c y vdbeapi.c. Dentro del compilador existe un módulo denominado Tokenizer, el cual se encarga de evaluar las sentencias SQL. Luego un analizador asigna un significado en función del contexto, se crea el contexto gramatical de la sentencia definida por la sentencia SQL que entiende SQLite y se almacena en el parser. Después de que el analizador reúne el análisis sintáctico, el generador de código se ejecuta para analizar el árbol de instrucciones y generar el código de bytes que contienen la instrucción SQL, se crea el programa, y es ejecutado por la máquina virtual.

Una base de datos creada en SQLite se almacena en una estructura de árboles tipo B-Tree y se encuentra guardada en el archivo btree.c. Este módulo del B-Tree se encarga de solicitar información al disco en páginas de tamaño fijo, por defecto de 4096 bytes, la paginación en caché permite la reversión y confirmación atómica de las transacciones y se encarga de bloquear los registros en la base de datos. A continuación se muestra una imagen que ejemplifica el diseño de la arquitectura de una base de datos SQLite.



5.4.2. Características

De acuerdo con Features of SQLite (2006), dentro de las principales características de SQLite se encuentran las siguientes:

- Transacciones atómicas, consistentes, aisladas y durables (ACID).
- Soporte estándar de SQL con capacidades avanzadas como índices parciales.
- Soporte de almacenamiento para grandes bases de datos, incluso terabytes y cadenas de gigabytes.
- Fuentes de dominio público, por lo que se puede utilizar para cualquier propósito.

Dentro de los usos sugeridos de SQLite se pueden mencionar los siguientes:

- Internet de las Cosas
- Almacenamiento de sitios Web
- Sustituto de un RDBMS con fines de demostración o para la pruebas

- Análisis de datos
- Como un servidor caché de datos

5.4.3. Instalación

Para la instalación de SQLite se debe ingresar a la página oficial del producto, en el apartado de descargas (http://www.sqlite.org/download.html) y buscar el archivo que posee los binarios compilados para la versión correspondiente, específicamente el que se encuentra con nombre de ***sqlite-tools-win32-x86-*.zip***. El mismo debe ser descargado al equipo en una ruta fácil de localizar, como por ejemplo: **C:\sqlite**. Es importante mencionar que esta ruta puede variar conforme las consideraciones del usuario y las prestaciones del equipo.

Una vez copiado el archivo comprimido en la ruta especificada, se deben extraer todos los archivos, los cuales corresponden a los siguientes nombres:

- sqlite3_analyzer.exe
- sqlite3.exe
- sqldiff.exe

Una vez realizada la extracción satisfactoria de todos los archivos mencionados, se debe ingresar a la utilidad de la línea de comandos o command prompt, se navega hasta la carpeta donde se encuentran los archivos y se ingresa el nombre de sqlite3.exe. Una vez ocurrido esto se debería desplegar un mensaje como el que se muestra a continuación:

```
C:\>sqlite3
SQLite version 3.8.5 2014-05-29 12:36:14
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

5.4.4. Requerimientos

No se especifica un tipo de requerimiento especial para la ejecución de este motor de base de datos. Por su simplicidad y debido a que es una herramienta desarrollada en C hace que sea una utilidad poco demandante de recursos computacionales para su ejecución. Sí se debe tomar en cuenta que se debe descargar los archivos correspondientes a cada sistema operativo donde se va ejecutar.

5.4.5. Consola de SQLite

Tal y como se muestra en Command Line Shell For SQLite (2006), SQLite proporciona una consola de comandos muy sencilla, llamada sqlite3, para Windows, que permite ejecutar instrucciones SQL e interactuar con el motor de base de datos. Se accede escribiendo sqlite3 en el símbolo del sistema. Para finalizar la consola se debe escribir el carácter de fin de línea, por lo general se puede introducir con la combinación de teclas (Ctrl+D).

Ejemplo del uso de la consola:

```
$ sqlite3 ex1
SQLite version 3.8.5 2014-05-29 12:36:14
Enter ".help" for usage hints.
sqlite> create table tbl1(one varchar(10), two
smallint);
sqlite> insert into tbl1 values('hello!',10);
sqlite> insert into tbl1 values('goodbye', 20);
sqlite> select * from tbl1;
hello!|10
goodbye|20
sqlite>
```

5.4.6. Plataformas soportadas

SQLite es un motor de base de datos multiplataforma, dentro de los sistemas soportados se encuentran Android, * BSD, iOS, Linux, Mac, Solaris, VxWorks, y Windows (Win32, Windows CE, WinRT).

5.5. Pruebas de rendimiento

Seguidamente se presentan pruebas de rendimiento con respecto a los motores de bases de datos en cuestión, los cuales se orientaron a procesos de carga masiva de datos y ejecución de consultas. Con estas pruebas se obtuvieron resultados comparables entre sí logrando de esta manera una perspectiva más real con respecto a las bases de datos tradicionales. Para ello se ha propuesto una serie de aspectos a medir, los cuales son los siguientes:

- **Cantidad de hilos de ejecución**

  Este aspecto comprende los hilos consumidos por el motor de base de datos, incluyendo procesos asociados a la ejecución de cada sentencia evaluada.

- **Tiempo de ejecución**

  Este aspecto comprende el tiempo consumido por el motor de base de datos para completar la operación evaluada.

- **Uso de memoria**

  Este rubro comprende el consumo y liberación de memoria durante y una vez finalizada la operación evaluada.

- **Uso del procesador (Ciclos de reloj)**

  Esta característica comprende el consumo y liberación de uso del procesador y una vez finalizada la operación evaluada, la cual se basa en la cantidad de ciclos de reloj ejecutados.

- **Lectura y Escritura en Disco Físico**

  Este aspecto comprende el uso del disco duro para almacenamiento y recuperación de datos a petición del motor.

- **Lectura y Escritura en bytes de Memoria**

Este aspecto comprende la lectura y escritura en bytes que son realizados por la ejecución de las sentencias en el motor.

5.5.1. Elección del equipo para ejecución de pruebas

La adecuada selección del equipo de hardware para la ejecución de las pruebas de rendimiento tiene un importante impacto sobre los resultados que se pueden obtener a la hora de instalar y aplicar las sentencias por cada uno de los motores. Es por ello que nos dimos a la tarea de analizar los recursos disponibles para determinar su adecuada aptitud y competencia, con lo cual se obtuvieran resultados lo más cercanos a la realidad que enfrentan día con día las organizaciones.

Se tomaron en cuenta los siguientes aspectos para determinar las características requeridas:

- Prestaciones exorbitantes o muy bajas no representarían en gran mayoría los equipos que tanto empresas grandes como pequeñas poseen.
- Debido al origen de las pruebas y el aspecto evaluado, es necesaria una prestación adecuada en lo que se refiere a memoria RAM por el equipo seleccionado.
- El espacio disponible en disco duro debe ser suficiente, ya que es necesaria la carga de archivos de datos considerablemente grandes para las cargas a realizar.
- Se requirió de acceso local al equipo para tener escenarios más acertados, de tal manera que permitiera la gestión directa de los archivos de carga y de los motores a evaluar.

En seguida, se muestra el detalle de las características de hardware y software que poseía el equipo en el que se realizaron las pruebas:

Recurso de Hardware/Software	Características	
Fabricante	**Modelo**	OptiPlex 980
	Fabricación	Dell Inc.
	Número de Serie	D3P2MM1
Tarjeta Madre	**Nombre del Producto**	0D441T
	Fabricación	Dell Inc.
	Número de Serie	..CN7360406800QT.
	Versión	A01
Detalle del BIOS	**Nombre**	Phoenix ROM BIOS PLUS Version 1.10 A02
	Fabricación	Dell Inc.
	Número de Serie	D3P2MM1
	Fecha de lanzamiento	29/04/2010
	Versión SMBIOS	A02
Memoria	**Capacidad**	16 GB
	Fabricación	80CE000080CE
	Forma-factor	DIMM
	Velocidad	1333
Procesador	**Nombre**	Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz
	Frecuencia y velocidad del Reloj	inventoring 2794 MHz
	Velocidad máxima del Reloj	2794 MHz
	Reloj Externo	133 MHz
	Multiplicador	21
	Descripción	Intel64 Family 6 Model 30 Stepping 5
	Fabricación	GenuineIntel
	Estado CPU	CPU habilitado

	Designación del zócalo	CPU
	Tamaño de la cache L2	1024 KBytes
	Velocidad de la cache L2	Ningún detalle
Disco Duro	**ID del Producto :**	WDC WD1600HLFS-75G6U1
	Número de Serie	W -DXW1G3C903223
	Tipo de Interface:	SCSI
	Fabricación	(Standard disk drives)
	Tipos de medios:	Fixed hard disk media
	Particiones:	4
	Tamaño:	149.01 GB
Sistema Operativo	**Nombre**	Microsoft Windows 7 Professional
	Fabricación	Microsoft Corporation
	Versión	6.1.7600
	Organización	Microsoft
	Nivel de encriptación	256 bits
	Lenguaje OS	English (United States)
	OS type:	WINNT
	Arquitectura de sistema operativo:	64-bit
	Tipo OS	Estación de Trabajo

5.5.2. Recopilación de datos

Otro aspecto importante para las pruebas a realizar es la elección y recopilación de los datos que van a ser procesados por los motores de bases de datos en memoria. Básicamente se procuró contar con 2 características como son:

- **Cantidad de registros**

  En este aspecto se procuró establecer una cantidad de registros de datos lo suficientemente representativa, y con la cual se pueda tener un escenario lo más cercano a la realidad, con respecto a cargas y consultas.

- **Tipos de datos**

  En este aspecto se determinó la necesidad de involucrar el uso de tipos de datos más usados en los distintos motores más comunes utilizados por las organizaciones, esto hace que se ponga a prueba también el procesamiento de la información de manera más acorde con la realidad.

5.5.2.1. Formato de recopilación de datos

La recopilación de datos para pruebas involucra la creación de algún tipo de proceso encargado de extraerlos a un formato de archivo que pueda ser procesado por cada uno de los motores de bases de datos que van a ser evaluados. Se tiene en consideración aspectos como la comunicación que se puede realizar con cada motor elegido, lo cual es una limitante con respecto al uso de procesos de carga utilizando paquetes de extracción, transformación y carga. Con esto se determina que para lograr una uniformidad con respecto al procesamiento de la información es necesaria la creación de archivos con delimitadores de columnas, es decir, formatos de archivos planos que puedan ser procesados de manera transparente por cada motor.

Uno de los formatos más soportados por cada uno de los motores involucrados en la investigación corresponde al archivo CSV (Comma Separated Values), en donde como su nombre lo indica cada columna es delimitada por un carácter de coma “,”. Esto hace posible que el motor identifique el dato tomando como punto de referencia el inicio y fin establecido por el carácter “,” con ello estableciendo una relación entre las columnas pertenecientes a cada tabla con respecto a la disposición de los campos del archivo. Con esto se tiene un escenario sencillo con respecto al procesamiento de los datos provenientes de orígenes distintos que van a ser cargados en cada motor evaluado.

5.5.2.2. Definición y estructura de las tablas utilizadas

Básicamente se trabajó con dos tablas para almacenar la información de los archivos CSV. Dicha información se tomó de una base de datos Microsoft SQL Server, de un ERP descontinuado que actualmente se utiliza de solo consulta, pero que contenía un histórico transaccional significativo, lo cual permitió tomar un buen volumen de datos para realizar las pruebas de esta investigación. Para tales pruebas se redujo la cantidad de columnas de ambas tablas, esto con la finalidad de dejar la información más significativa, para evaluar los rendimientos descritos en el alcance de este proyecto.

La primera tabla se llamó F4111M, perteneció a los movimientos de inventario de una compañía comercializadora de electrodomésticos, mientras que la segunda se llamó F0911M y correspondía a las transacciones de movimientos contables. Dichas tablas se crearon con la misma estructura en cada uno de los motores de bases de datos en memoria evaluados.

Seguidamente se muestra la estructura de cada una de las tablas, con sus correspondientes columnas y tipos de datos utilizados, los cuales se heredaron de la base de datos original, definidos para el sistema ERP que las utilizaba.

Inventario (F4111M)	
Columna	**Tipo dato**
ILLITM	char(25)
ILMCU	char(12)
ILKCO	char(5)
ILDOC	float
ILDCT	char(2)
ILICU	float
ILDGL	numeric(18, 0)
ILTRUM	char(2)
ILTRQT,	float
ILUNCS,	float
ILPAID	float

Contabilidad (F0911M)	
Columna	**Tipo dato**
GLKCO	char(5)
GLDCT	char(2)
GLDOC	float
GLDGJ	numeric(18, 0)
GLPOST	char(1)
GLICU	float
GLICUT	char(2)
GLANI	char(29)
GLPN	float
GLFY	float
GLCRCD	char(3)

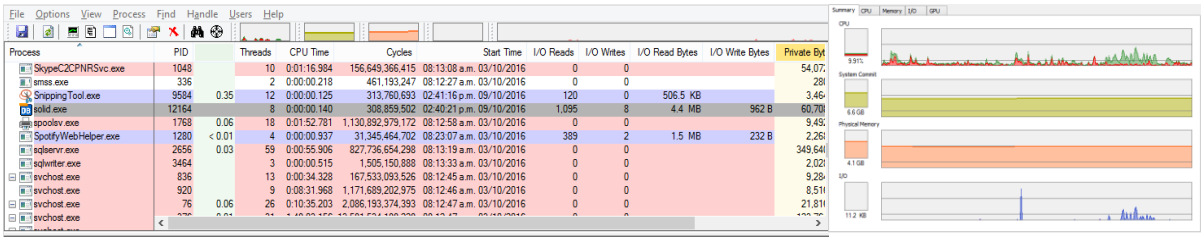
GLAA	float

5.5.3. Herramienta de monitoreo

Inicialmente se trató de utilizar la herramienta de administración de tareas nativa de Windows, sin embargo, por sus características resultó no ser apta para esta investigación, ya que la información que se detalla resulta limitada para el nivel de detalle con el que se deseaba monitorear los recursos. Al investigar en internet se encontró una serie de herramientas que cubrían las necesidades y brindaban el nivel de detalle requerido.

Para esta investigación se utilizó la herramienta de monitoreo de procesos llamada "Process Explorer" en su versión v16.12, que de acuerdo con Russinovich (2016), es una herramienta bastante completa para monitorear ejecución de procesos individuales. Con Process Explorer se puede monitorear información relevante, relacionada con el uso de los recursos que está empleando cada uno de los procesos que se ejecutan. Además permite guardar un archivo con dicha información, de esta manera se logró extraer el detalle de consumo de los recursos para los procesos ejecutados, según fuera el motor de bases de datos evaluado.

Elegimos esta herramienta por lo fácil de utilizar y por la facilidad de guardar el consumo de recursos, indispensable para el análisis posterior de cada motor de base de datos. Process Explorer está compuesto básicamente de dos pantallas, la principal es la que muestra el detalle de uso de recursos por cada proceso del sistema, la segunda es una pantalla que muestra de forma gráfica el uso de memoria, CPU, escrituras y lecturas de disco duro.



5.5.4. Definición de pruebas de rendimiento

Los escenarios de pruebas de rendimiento seleccionados corresponden a tareas que en ocasiones y dependiendo del negocio de la organización deben ser ejecutadas diariamente, pero que debido a su volumen de datos pueden sacrificar y comprometer la disponibilidad de los equipos que ejecutan dichos motores. Es por ello que por lo general se crean procesos calendarizándolos de manera tal que no afecten de manera negativa la operatividad de la organización, sin embargo, esto podría interferir con la toma de decisiones por parte de los jerarcas de las empresas al no contar con datos lo suficientemente actualizados, debido a la limitante de contar con datos actualizados casi de manera inmediata y con respuestas rápidas a consultas complejas.

De este modo se definieron dos ámbitos sobre los cuales definir escenario de pruebas de rendimiento, los cuales son:

- **Carga de datos**

Una de las tareas más complejas y pesadas a las cuales se puede enfrentar un esquema de base de datos es la carga de información, esta puede comprender el procesamiento de grandes cantidades de registros diarios que son necesariamente requeridos replicar en estructuras que se encuentren disponibles de manera casi inmediata para su consulta. Debido a esto y a la necesidad de contar con información actualizada, es que se elige este ámbito como un escenario de prueba importante y sujeto a estudio.

- **Consultas**

Otra de las tareas complejas y en las cuales en ocasiones los motores de bases de datos tradicionales no han podido satisfacer de manera completa, es la respuesta a consultas realizadas empleando estructuras que cruzan tablas que contienen grandes cantidades de registros. En situaciones particulares donde es requerido el cruzamiento de históricos grandes, se hace casi imposible una respuesta inmediata a la solicitud enviada por el usuario. Debido a esto se define la consulta como otro ámbito importante y sujeto a estudio.

5.5.4.1. Diseño Experimental

Debido a que en este proyecto se utilizó un diseño experimental basado en un tipo de investigación evaluativa, se ha estudiado el comportamiento de cada motor de base de datos tomando en consideración una serie de variables las cuales guardan una estrecha relación con la administración empleada por cada motor para hacer uso de los recursos de hardware. La finalidad del experimento es poder comparar los resultados obtenidos mediante la observación de las variables controladas.

Dentro del diseño experimental los factores de tratamiento que se definen pueden ser clasificados en dos grupos, que corresponden a cualitativos o cuantitativos. El primero de ellos se refiere a una característica específica mientras que el segundo grupo lo determina la posibilidad de llevar acabo algún tipo de medición.

Para la investigación realizada, tomando como base la definición de diseño experimental, se trató entonces de identificar para la evaluación de cada motor de base de datos factores principales que pueden afectar el rendimiento de cada uno de ellos. Es así que en el caso particular, la variable de interés se refiere meramente al desempeño que se obtenga con cada medición de cada una de las operaciones a realizar por el motor, y los factores influyentes son los siguientes:

Dentro de los factores cualitativos se encuentran:

1. Motor de base de datos: La arquitectura creada por el desarrollador del motor de base de datos puede variar significativamente, este es un factor de gran importancia debido al nivel de conocimiento que pueda tener el desarrollador con respecto al manejo y depuración de sentencias de SQL.
2. Replicación a disco duro: Los procesos de replicación en disco duro para cada una de las operaciones ejecutadas por el motor de base de datos, factor de importancia debido a que puede interferir con las mediciones con respecto al rendimiento del equipo.

3. Bloques de lectura: Funcionalidad de lectura de archivos, factor de importancia debido a cómo se realiza el acceso al archivo de datos para la carga de información y ejecución de consultas en cada uno de los motores.
4. Uso de la memoria: La forma en que el motor administra y consume la memoria disponible en el equipo donde se ejecuta.

Con la definición de este diseño experimental elaborado para la investigación se logró detectar pequeños aspectos o factores que influyen en la medición obtenida con respecto al rendimiento de cada motor, teniendo una variable de respuesta basada en el tiempo de ejecución que consumió cada motor de base de datos evaluado para completar una tarea específica (carga masiva de datos y ejecución de consultas).

5.5.4.2. Aspectos a considerar para evitar contaminación y sesgos

La ejecución de las pruebas de rendimiento de cada uno de los motores de bases de datos seleccionados demandó la necesidad de definir minuciosamente las operaciones a realizar, para la preparación del ambiente sobre el cual se fuese a procesar cada instrucción evaluada del motor. La necesidad de contar con un ambiente lo más intacto posible responde a poder generar resultados lo más limpios y reales posibles, es decir, obtener resultados sin que previas tareas ejecutadas por el equipo interfirieran en cada una de las iteraciones ejecutadas. Para ello entonces se definieron las siguientes operaciones a realizar previas a cada procesamiento de cada uno de los archivos con sus respectivas instrucciones para el motor correspondiente:

1. Desactivar y dejar en modo manual todos los servicios no necesarios para la operación del equipo, para lo cual contar con la mayor cantidad de memoria disponible para la ejecución de los procesos de carga y rendimiento.
2. Preparación de los scripts a ejecutar sobre cada motor con sus respectivos archivos de prueba.
3. Instalación y configuración de los motores a evaluar de modo que se encuentren ya disponibles para su evaluación, evitando de esta forma consumir recursos innecesarios del equipo previo a cada prueba ejecutada.

4. Reinicio del equipo sobre el cual se ejecuta el motor a evaluar, esto con la finalidad de poder liberar toda la memoria posible, dejando lo más intactos posibles todos los procesos que no deban ser iniciados por el equipo.
5. Definir los pasos a realizar para ejecutar cada motor, acceder los archivos y scripts de prueba, de modo que se sigan los mismos pasos y secuencias para evitar así consumos innecesarios de memoria que puedan alterar los resultados de la evaluación.
6. Establecer los pasos necesarios para la recopilación de la información sobre cada ejecución, de modo que no se utilicen herramientas distintas para el almacenamiento de los resultados obtenidos con cada prueba.

Con las operaciones anteriores entonces se logró definir una serie de aspectos que son muy importantes tomar en cuenta ante toda evaluación de rendimiento, con respecto al uso de los recursos del equipo sobre el cual se ejecuta la herramienta. Es por ello que estas consideraciones van a garantizar que se tengan valores lo más acordes con la realidad, evitando así que las comparaciones no contengan un sesgo importante debido a que no se mantuvieron las precauciones respectivas que afecten directamente la prueba de evaluación.

5.5.4.3. Prueba de rendimiento: Carga de Datos

La prueba de rendimiento correspondiente a la carga masiva de datos tiene como finalidad medir todos los rubros mencionados anteriormente, por lo que fue realizada definiendo dos características específicas:

- Carga de dos tablas específicas.
- Uso de dos archivos con formato CSV de aproximadamente 10 000 000 registros cada uno, para su procesamiento.

De esta forma, entonces, se generaron los archivos con la cantidad de registros mencionados, los cuales fueron cargados al equipo donde se realizó la instalación de los motores de bases de datos en memoria.

5.5.4.3.1. Justificación del uso del dataset

Muchas veces uno de los principales problemas a la hora de realizar pruebas de rendimiento con bases de datos suele ser encontrar un dataset de información acorde al proceso o modelo de negocios que se desea probar.

Como bien es sabido, existen numerosos datasets en internet disponibles para descargar, muchos de ellos avalados por la comunidad y algunos con la opción de obtenerlos incluso gratis. Sin embargo, para el desarrollo de este proyecto se utilizó un set de datos propio, seleccionado de un repositorio de datos transaccionales bastante amplio, el cual contenía información de más de 10 años, 100% real de una empresa transnacional. Básicamente se decidió realizarlo de esa manera debido a que necesitábamos un gran volumen de datos y para los efectos de las pruebas de rendimiento aplicadas en este proyecto; quisimos basarnos en un amplio espectro de tiempo, el cual tuviera contenido de varias compañías, para hacerlo aún más complejo al momento de realizar las mediciones de rendimiento para las ejecuciones de las consultas. Realmente dentro de los criterios de búsqueda realizada no se encontró un set de datos que cumpliera con estas características requeridas, de varios años de información transaccional y que a la vez fuera información de muchas compañías contenidas en el mismo dataset, es por ello que finalmente se decidió utilizar un set de datos propio.

5.5.4.3.2. Preparación y ejecución de la prueba

Para la preparación y ejecución de la prueba de carga masiva de datos se siguieron los siguientes pasos, los cuales deben ser aplicados a cada uno de los motores de bases de datos de igual manera, a fin de obtener resultados bajo escenarios lo más similares posibles y evitar de esta forma efectos negativos o que ocasionen ruido a la evaluación:

- Ejecución del motor de base de datos.
- Escritura del script de carga de datos, usando como referencia cada uno de los archivos ubicados en el equipo que contiene la información de carga.
    - El script debe incluir:
        - Fecha de inicio de la ejecución de la sentencia

        - Sentencia de carga del archivo especificado
        - Fecha de fin de la ejecución de la sentencia
    - Ejecución del monitor de recursos e identificar los procesos a dar seguimiento para la toma de datos correspondientes.
        - Toma de datos del estado inicial del equipo.
    - Puesta en ejecución del script en el motor de base de datos correspondiente.
    - Toma de datos del equipo una vez finalizado el script de carga masiva de datos.
    - Recopilación de toda la información generada por la carga para su representación tabular y gráfica.

Se hace la aclaración de que las pruebas ejecutadas durante el desarrollo de este proyecto fueron sobre los cuatro motores de bases de datos que se han venido mencionando a lo largo del presente documento. Sin embargo, antes de mencionar algún resultado a cerca de cualquier motor, se quiere dejar en claro que los resultados para el motor de base de datos H2 son aproximaciones con lo que pudo haber ocurrido en la realidad, esto debido a que H2 no soportó el volumen de datos que se utilizó para evaluar las cargas de información en cada uno de los motores en cuestión. Aunque se trató de cargar un set de datos ligeramente inferior al original, los resultados no fueron satisfactorios.

H2 tiene una limitación importante en cuanto a la cantidad de memoria que utiliza para gestionar la información. Para el caso de las cargas de datos en H2, se cargaron solamente la mitad de los registros que se utilizaron con los demás motores, mientras que, para las pruebas de consultas, solamente se utilizó un cuarto de la cantidad de registros utilizados con los demás motores.

Una vez aclarado este punto, lo que se hizo realmente entonces con H2 fue tratar de calcular una aproximación en términos de tiempo y recursos utilizados, con el fin de homologar los resultados para poder realizar el análisis comparativo con respecto a los demás motores evaluados.

Con el fin de representar de una forma más simple los resultados obtenidos, se propone el uso de una tabla comparativa que se muestra a continuación:

Prueba De Rendimiento		Carga masiva de datos			
Cantidad Registros	10 000 105	Motor de Base de Datos			
Rubro Evaluado		Solid DB	H2	Derby DB	SQLite
Cantidad de Hilos de Ejecución (Threads)		30	69	22	3
Tiempo de Ejecución (s)		416	67	90	86
Uso de Memoria (Gb)		2.08 Gb	4.64 Gb	4.28 Gb	0.87 Gb
Uso del Procesador (Ciclos de Reloj Expresado en Millones)		1.282.957	333.105	279.688	237.267
Cantidad de Lecturas a Disco Duro (I/O Read)		211.406	110.772	110.362	208.225
Cantidad de Escrituras a Disco Duro (I/O Write)		107.945	566	6	0
Cantidad de Bytes Leídos de Disco Duro (I/O Read Bytes)		0.79 Gb	0.49 Gb	1.38 Gb	1.13 Gb
Cantidad de Bytes Escritos en Disco Duro (I/O Write Bytes)		3.10 Gb	580,2 Kb	730 B	0 B

Prueba De Rendimiento		Carga masiva de datos			
Cantidad Registros	10 000 133	Motor de Base de Datos			
Rubro Evaluado		Solid DB	H2	Derby DB	SQLite

Cantidad de Hilos de Ejecución (Threads)	29	69	22	3
Tiempo de Ejecución (s)	463	78	104	145
Uso de Memoria (Gb)	3.82 Gb	9.29 Gb	4.29 Gb	1.72 Gb
Uso del Procesador (Ciclos de Reloj Expresado en Millones)	977.393	376.506	297.933	256.411
Cantidad de Lecturas a Disco Duro (I/O Read)	395.448	135.726	234.709	456.898
Cantidad de Escrituras a Disco Duro (I/O Write)	203.711	634	6	0
Cantidad de Bytes Leídos de Disco Duro (I/O Read Bytes)	3.20 Gb	1.96 Gb	1.7 Gb	1.7 Gb
Cantidad de Bytes Escritos en Disco Duro (I/O Write Bytes)	1.7 Gb	1320.8 Kb	730 B	0 B

Prueba De Rendimiento		**Consulta cruzada de inventarios**			
Cantidad de Tablas	**2**	**Consulta cruzada de 10.000.105 x 10.000.133 registros**			
Rubro Evaluado		**Motor de Base de Datos**			
		Solid DB	**H2**	**Derby DB**	**SQLite**
Tiempo de Ejecución (s)		10	NA	420	50
Uso del Procesador (Ciclos de Reloj Expresado en Millones)		26.576	NA	1.171.962	135.185
Cantidad de Lecturas a Disco Duro (I/O Read)		2	NA	84	168.169

Cantidad de Escrituras a Disco Duro (I/O Write)	0	NA	0	351.449
Cantidad de Bytes Leídos de Disco Duro (I/O Read Bytes)	0.4 KB	NA	0	0.7 GB
Cantidad de Bytes Escritos en Disco Duro (I/O Write Bytes)	0	NA	0	1.3 GB

5.5.4.3.3. Gráficas de rendimiento

5.5.4.3.3.1. Carga Masiva de Datos

A continuación se presenta una serie de gráficos donde se puede observar de manera más representativa los resultados obtenidos, con esto se puede tener una mejor percepción del rendimiento de cada motor una vez sometido a la prueba de carga masiva de datos. Los gráficos entonces se orientan en cada uno de los rubros evaluados y mencionados en la tabla anterior. Se dispone de una gráfica por cada bloque de datos cargados por tabla, para observar el comportamiento de una mejor manera con respecto a cada ejecución realizada de manera independiente.

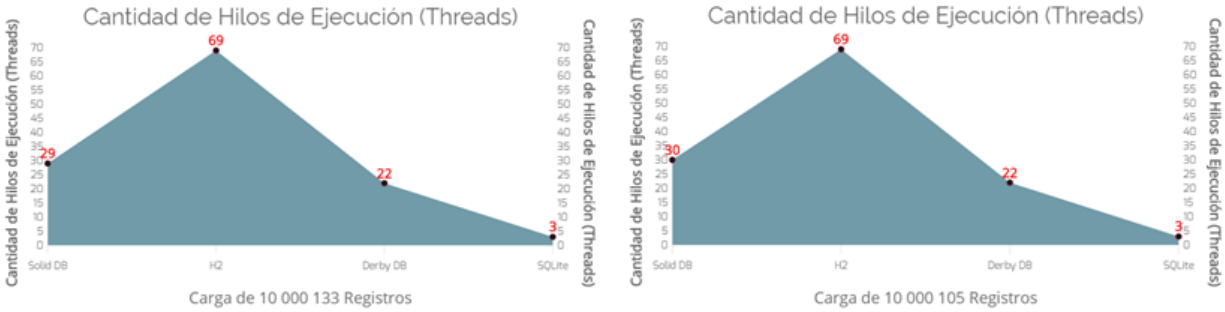


Ilustración 2 Cantidad de Hilos de Ejecución (Threads)

Las gráficas anteriores muestran la cantidad de hilos de ejecución empleados por cada uno de los motores de bases de datos. Se puede observar que para el caso de SQLite se

toman menor cantidad de hilos para realizar la tarea de carga, mientras que H2 es el que consume más recursos.

Observemos a continuación el comportamiento con respecto a los tiempos de ejecución para la carga de los datos de cada una de las tablas.

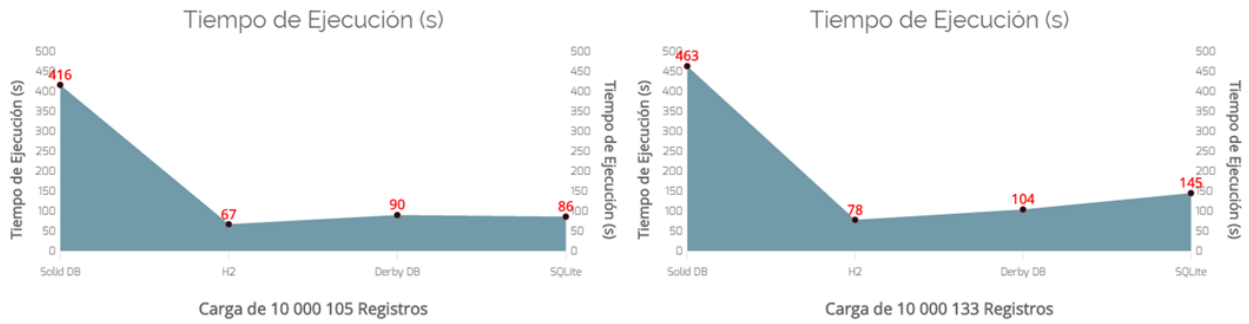


Ilustración 4 Tiempo de Ejecución

La gráfica anterior incluye la información en cuanto a tiempos de ejecución para realizar la carga de los registros de las tablas definidas. Es posible observar que Solid DB tiene el pico en cuanto a tiempo utilizado para realizar la carga, posicionándose por debajo de todos los motores H2, que registra el tiempo de menor magnitud, lo cual lo convierte para esta prueba en el motor más eficiente. Sin embargo, es importante recordar que este tiempo para H2 es solamente una aproximación.

Observemos a continuación el comportamiento a modo de gráfico con respecto al uso de memoria por los distintos motores en el momento de la carga de datos:

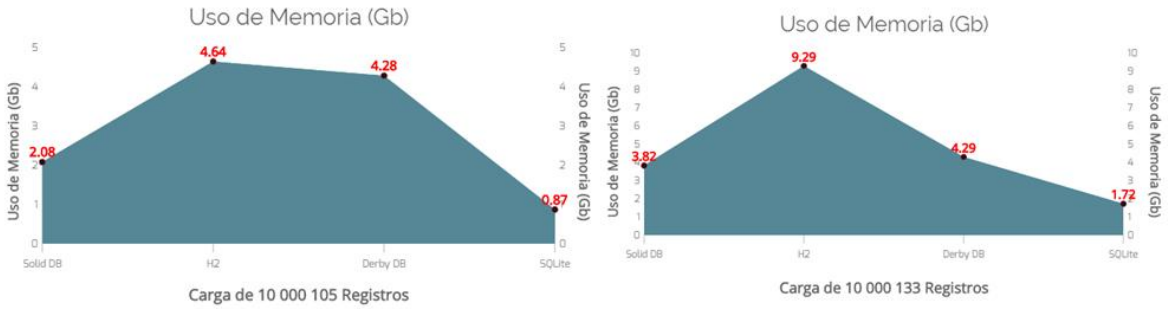


Ilustración 5 Uso de Memoria (Carga de Datos)

Con respecto al uso de la memoria por cada uno de los motores, uno de los rubros más importantes a observar y analizar es el consumo utilizado por H2, el cual es bastante similar al motor Derby DB. Continuamos observando un comportamiento más a favor sobre SQLite que, si bien es cierto no realiza replicación en disco duro, posee un consumo mucho menor en recursos del equipo en cuanto a memoria se refiere.

Observemos el comportamiento en el uso de ciclos de reloj empleados por cada uno de los motores para esta tarea de carga masiva de datos.

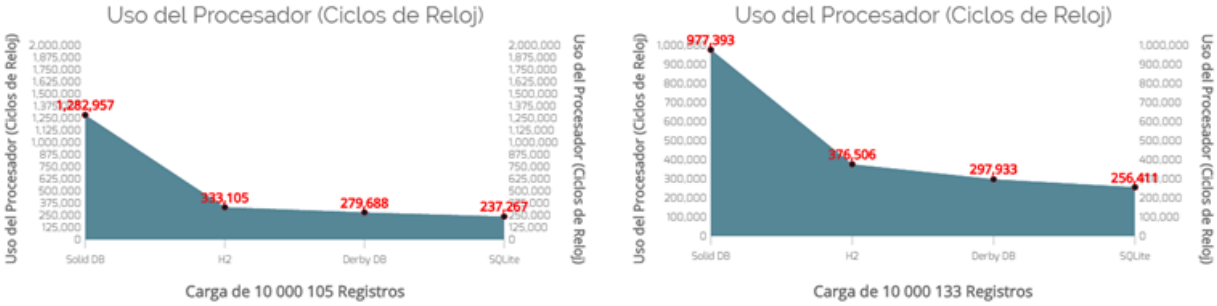


Ilustración 6 Uso del Procesador Ciclos de Reloj (Carga de Datos). Métricas expresadas en Millones.

Como es posible observar en el gráfico anterior, el uso del procesador con respecto a ciclos de reloj es llevado a un límite por Solid DB, y registra una disminución considerable y por debajo de los demás motores el que corresponde a SQLite. La característica de replicación de datos en disco duro que posee Solid DB por sobre los demás motores evaluados puede ser el disparador en cuanto a ciclos de reloj utilizados para este proceso de carga, por lo cual el equipo es llevado a un límite en su rendimiento.

Observemos a continuación el gráfico de cantidad de lecturas a disco duro por cada uno de los motores con respecto a esta carga de datos masiva:

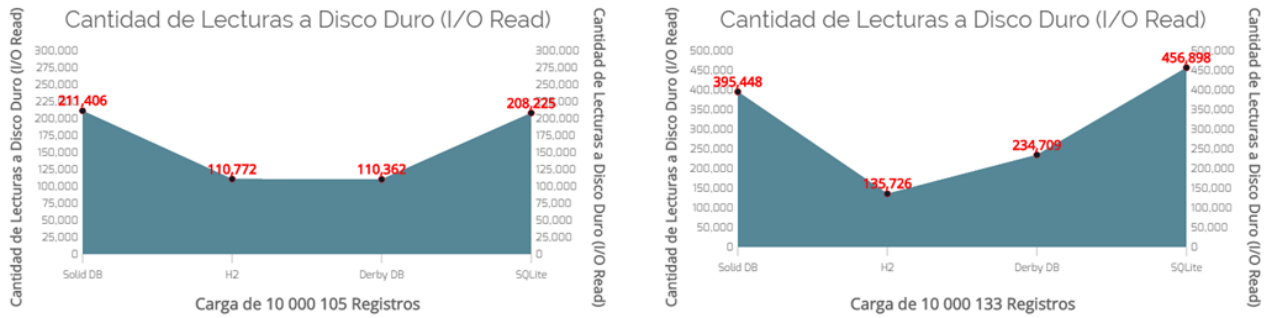


Ilustración 7 Lectura de Disco Duro (Carga de datos)

En el gráfico anterior podemos observar la cantidad de lecturas de disco duro realizadas por los diferentes motores para esta prueba de carga masiva de datos. Se puede apreciar que en este caso Derby DB lleva la delantera con la menor cantidad de accesos para lectura en la carga del primer conjunto de datos y posteriormente llevar el registro a memoria. Para este rubro de rendimiento Solid DB pierde su puesto al quedar en tercer lugar con respecto a sus homólogos.

Observemos a continuación el gráfico de cantidad de escrituras a disco duro por cada uno de los motores con respecto a esta carga de datos masiva:

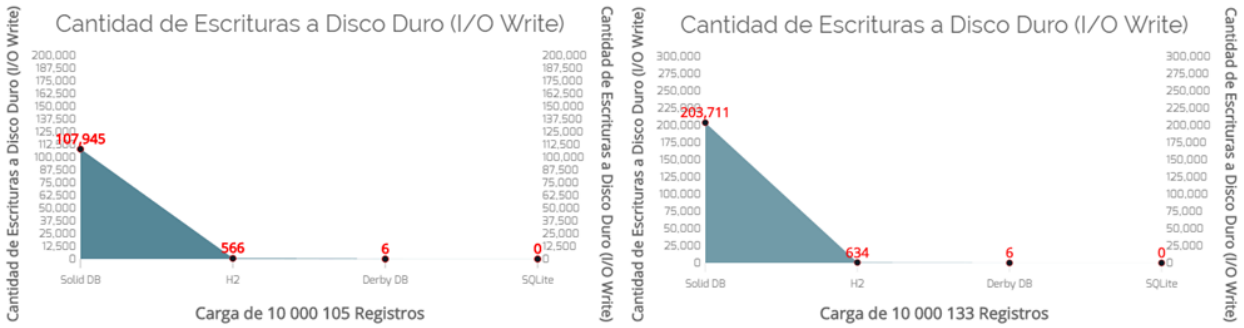


Ilustración 8 Cantidad Escritura Disco Duro

Para el gráfico anterior se observa la cantidad de escrituras en disco, lo cual es un punto muy a favor de SQLite, el cual no realiza ninguna escritura, independientemente de la cantidad de registros que se encuentren cargando durante el proceso. Seguidamente se encuentra Derby DB, que también deja al descubierto su buen rendimiento en este aspecto evaluado.

5.5.4.3.3.2. Consulta Cruzada de Inventarios

Con respecto a las consultas realizadas una vez cargadas las tablas con los 10 millones de registros cada una aproximadamente, se puede a continuación observar unas gráficas que permiten apreciar más claramente el rendimiento de cada motor de base de datos a la hora de responder a las sentencias de recolección de datos. La consulta empleada para la medición de tiempos se puede apreciar a continuación:

```
SELECT      GLICUT,
            ILDGL,
            GLANI,
            GLCRCD,
            SUM(GLAA) /100 AS GLAA
FROM F4111M
INNER JOIN F0911M
        ON ILKCO = GLKCO
        AND ILDCT = GLDCT
       AND ILDOC = GLDOC
       AND ILICU = GLICU
WHERE ILKCO= '95000'
AND ILDOC = '27890'
AND ILDCT = 'RI'
AND ILICU = 377727
AND GLPOST = 'P'
GROUP BY GLICUT, ILDGL, GLANI, GLCRCD;
```

La consulta entonces combina las dos tablas ya cargadas, retornando una columna calculada y realizando un agrupamiento para aplicar la instrucción de sumatoria. Se realizó una consulta que involucrara varios millones de registros para observar el tiempo de procesamiento de la data que reside en memoria, y de esta forma tener resultados lo más acordes con la realidad, que se asemejen a los volúmenes de datos que las diferentes organizaciones se enfrentan día con día. Es importante recordar que, en muchas ocasiones, es necesario valerse de procesos nocturnos donde se pre-calcula la información con el fin de mejorar de cara al usuario los tiempos de respuesta de las consultas. Con los motores de bases de datos en memoria se puede solventar gran parte de este tipo de problemas de lentitud que se presentan en la actualidad.

Seguidamente, se observa el gráfico correspondiente a los tiempos que fueron consumidos por cada motor a la hora de satisfacer la sentencia mencionada anteriormente:

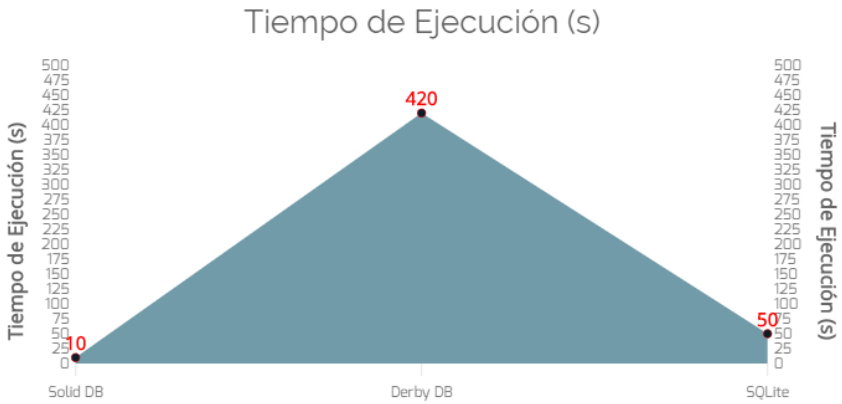


Ilustración 9 Tiempo Ejecución Consulta

El gráfico anterior deja en claro la respuesta de Solid DB con una marcada diferencia con respecto a Derby DB, sin embargo y no muy atrás se encuentra SQLite, el cual entregó un rendimiento muy satisfactorio con respecto a Derby DB. Aunque Solid DB quedó un poco atrás en tiempo consumido para las cargas masivas de datos, se puede ver su rendimiento con respecto a la satisfacción a las solicitudes de datos que son ejecutadas. Se debe tomar en cuenta que uno de los aspectos más importantes a verificar con este proyecto es el tiempo involucrado por cada uno de los motores investigados, a fin de observar un comportamiento mucho más satisfactorio y favorable para las organizaciones a la hora de enfrentarse a escenarios donde el trabajo con cantidades considerables de información se hace un poco engorroso y complejo.

Posteriormente, se observa un gráfico donde se despliega el uso del procesador con respecto a ciclos de reloj para satisfacer la consulta mencionada:

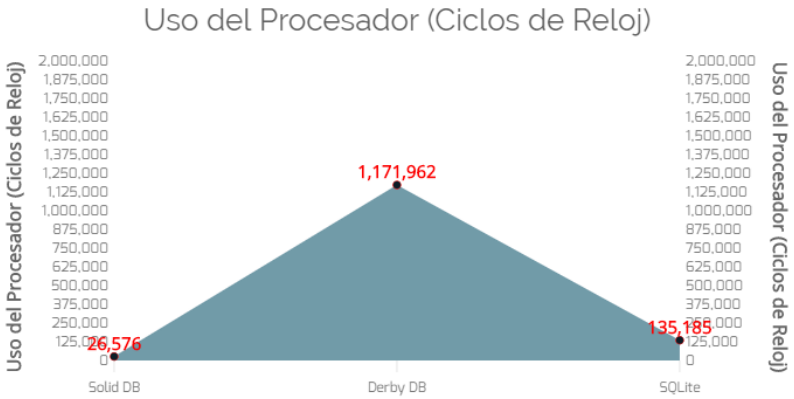


Ilustración 10 Uso del Procesador (Ciclos de Reloj). Métricas expresadas en Millones.

Para lo que corresponde al uso del procesador en cuanto a ciclos de reloj, se vuelve a posicionar Solid DB con métricas más eficientes con respecto a los demás motores evaluados. Es posible observar que la diferencia es considerablemente representativa, pues consumió en 100 mil millones de ciclos de reloj menos que SQLite y más de 1 billón de diferencia con respecto a Derby DB, que como vemos es el que más recursos ha consumido para satisfacer la consulta planteada y ejecutada.

Seguidamente, se muestra un gráfico que refleja la cantidad de lecturas en disco duro ejecutadas por los motores:

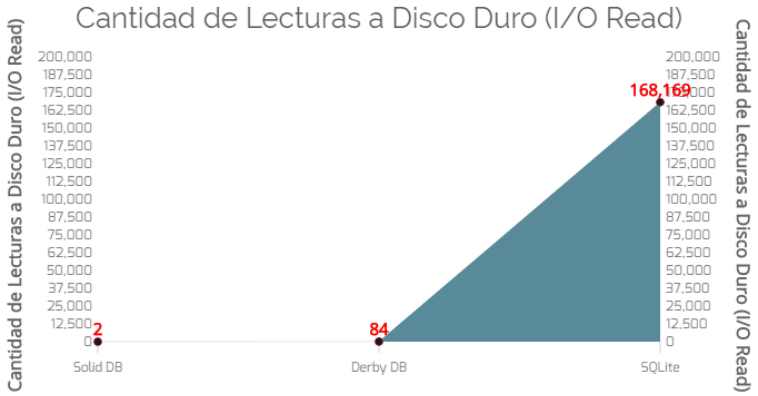


Ilustración 11 Cantidad Lecturas a Disco Duro

La gráfica anterior vuelve a posicionar a Solid DB por sobre los demás motores de bases de datos evaluados, donde SQLite es el que presenta la mayor cantidad de lecturas a

disco duro. Es importante recalcar que es un comportamiento un poco inesperado debido a que es un motor que se encuentra totalmente en memoria, por lo que no existe justificación lógica del consumo y lecturas realizadas a disco duro.

El gráfico siguiente refleja el comportamiento con respecto a la escritura de datos en disco duro por cada uno de los motores:

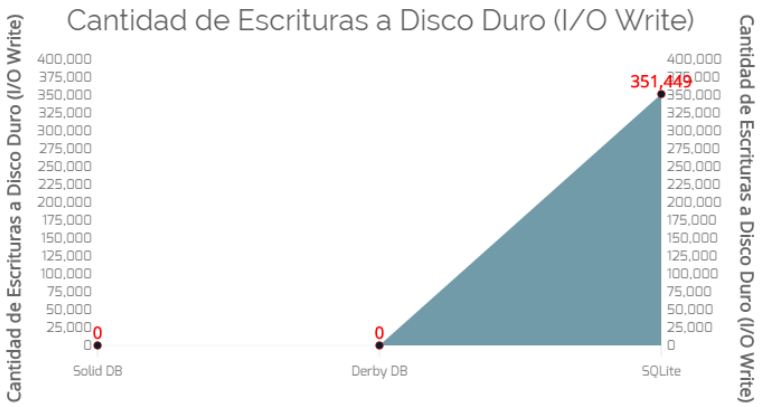


Ilustración 12 Cantidad Escrituras a Disco Duro

Con respecto a las escrituras en disco duro, tanto Solid DB como Derby DB se encuentran en las mismas posiciones, situación contraria a SQLite que vuelve con un comportamiento inesperado realizando escrituras en disco duro de información que debería estar alojada completamente en memoria.

Una vez observados los comportamientos de cada uno de los motores evaluados en esta prueba de consulta de datos, se puede observar que Solid DB marca una diferencia muy pronunciada a la hora de recuperar información y satisfacer sentencias ejecutadas por el usuario.

5.5.4.4. Análisis y consideraciones de los resultados obtenidos

Las distintas pruebas de rendimiento aplicadas a los motores evaluados pueden entregar una noción un poco más clara y acertada con respecto a una implementación real en alguna organización que se mantenga interesada de poder mejorar el rendimiento en las

aplicaciones que posee actualmente, y que estén presentando alguna deficiencia en este tema. Es importante mencionar que los resultados obtenidos pueden variar dependiendo del equipo y las bondades que este ofrezca en cada organización, por lo que debe tomarse en cuenta este aspecto ante cualquier implementación que se desee ejecutar.

Es de vital importancia conocer el volumen de registros con los cuales se trabaja el día a día en la organización, así como la cantidad de tablas a las cuales se va a enfrentar los procesos de carga y consulta, como se trató de representar en el proyecto y las pruebas ejecutadas. De cierta forma este documento ya presenta las herramientas básicas, así como las mediciones y cómo deben realizarse para verificar el rendimiento de otros motores a los cuales el lector desee enfrentarse para su análisis.

Basados en los resultados obtenidos en este proyecto, se puede destacar el desempeño que ha presentado el motor de bases de datos SolidDB para ejecutar las distintas tareas a las cuales fue sometido. Ello deja una buena percepción de rendimiento para su implementación en un ambiente más controlado y real. Cabe destacar su potencial con respecto a la ejecución de sentencias de consultas, lo cual lo hace un candidato muy apto en cuanto a escenarios donde el tiempo de respuesta sea uno de los aspectos más perseguidos por la organización.

Por otra parte, las características de replicación en disco duro lo convierten en una herramienta que además de ser ágil y veloz, sea además segura en términos de integridad de información ante situaciones como fallos en los sistemas y/o agentes externos, como por ejemplo la falta de suministro de energía eléctrica. Así mismo, la cantidad de variables de configuración permite que pueda ser instalado y personalizado de acuerdo con las necesidades de la organización.

5.5.4.5. Plantilla para evaluación de rendimiento

A continuación, se deja a disposición una plantilla sugerida para realizar pruebas o evaluaciones de rendimiento a motores de bases de datos en memoria, la cual puede ser empleada en posteriores análisis que apoyen la toma de decisiones por parte de la

organización que desee involucrar el uso de una tecnología en memoria para satisfacer las necesidades de rendimiento que demandan sus aplicaciones.

Equipo utilizado para las pruebas	
Recurso de Hardware/Software	**Características**
Fabricante	• *Fabricante del equipo*
Tarjeta Madre	• *Tarjeta madre del equipo*
Detalle del BIOS	• *Descripción del BIOS*
Memoria	• *Memoria disponible en el equipo*
Procesador	• *Cantidad, marca y detalles generales del procesador*
Disco Duro	• *Capacidad del disco duro*
Sistema Operativo	• *Sistema operativo del equipo*

Motor de Base de datos	
Característica	**Descripción**
Descripción	• *Una breve descripción del motor de base de datos.*
Arquitectura	• *Características importantes de su arquitectura.*
Requerimientos	• *Requerimientos mínimos para su ejecución*
Instalación	• *Generalidades sobre la instalación del motor*

Script de prueba	
Característica	**Descripción**
Descripción	• *Breve descripción del script y su funcionalidad*
Tipo	• *Script de carga, consulta, ejecución etc.*
Formato	• *CSV (Comma-Separated Values), Sentencias (insert, update, delete)*
Cantidad Registros	• *Cantidad de registros (para el caso que aplique).*
Fuente	• *De dónde provienen los datos y su manera de recopilación.*

Prueba De Rendimiento	
Cantidad Registros	**Motor de Base de Datos**
	Nombre del motor de base de datos
Rubro Evaluado	
Cantidad de Hilos de Ejecución (Threads)	• *Se debe indicar la cantidad de hilos de ejecución empleados para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*
Tiempo de Ejecución (s)	• *Se debe indicar el tiempo de ejecución para completar la tarea, debe representarse en una unidad de medida de fácil comprensión. Se recomienda el uso de segundos como unidad de medida.*
Uso de Memoria (Gb)	• *Se debe indicar el uso de memoria para completar la tarea, debe representarse en una unidad de medida de fácil comprensión. Dependiendo del consumo se debe cambiar la unidad de medida.*

Uso del Procesador (Ciclos de Reloj Expresado en Millones)	• *Se debe indicar el uso del procesador para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*
Cantidad de Lecturas a Disco Duro (I/O Read)	• *Se debe indicar la cantidad de lecturas a Disco Duro para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*
Cantidad de Escrituras a Disco Duro (I/O Write)	• *Se debe indicar la cantidad de escrituras a Disco Duro para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*
Cantidad de Bytes Leídos de Disco Duro (I/O Read Bytes)	• *Se debe indicar la cantidad de bytes leídos de Disco Duro para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*
Cantidad de Bytes Escritos en Disco Duro (I/O Write Bytes)	• *Se debe indicar la cantidad de bytes escritos de Disco Duro para completar la tarea, debe representarse en una unidad de medida de fácil comprensión.*

Capítulo 6. Conclusiones y Recomendaciones

6.1. Conclusiones

1. Actualmente existe una amplia gama de productos en el mercado, que funcionan bajo el esquema de tecnologías de almacenamiento de datos en memoria, y los hay muy variados. Algunos de estos productos han sido creados para solucionar problemas muy específicos, otros han sido desarrollados para solventar problemas comunes. Algunos son del tipo de datos SQL, otros son No SQL, los hay con licencias de software libre, otros son del tipo propietarios, para lo cual se necesita el pago de una licencia para poder hacer uso ininterrumpido, después de terminado el periodo de pruebas.

   Muchos de estos motores de bases de datos en memoria tienen la característica de soportar múltiples sistemas operativos, esto los convierte en plataformas más robustas, que a la vez permiten una mejor flexibilidad al momento de migrar los datos de un sistema operativo a otro. Esto no quiere decir que el proceso sea transparente, bien es sabido que la portabilidad en términos de migración de datos siempre es costosa, sin embargo al haber motores de bases de datos con características multiplataforma, de alguna manera, el proceso de migración se agiliza un poco.

   Por la naturaleza de este proyecto, para la selección de los motores evaluados, se tomó en cuenta que fueran arquitecturas de bases de datos que soportaran información del tipo transaccional, con datos del tipo SQL, debido a que aún en la mayoría de las empresas se utilizan sistemas que generan información transaccional (registros de datos). Existen opciones interesantes, algunos motores de bases de datos permiten replicar la información a disco duro, al mismo tiempo que mantienen la información disponible cargada en memoria, disponible para su utilización en cualquier momento. Algunos implementan variantes que permiten definir mediante variables de configuración parámetros para indicar por ejemplo la cantidad de memoria máxima a utilizar, o para disparar mensajes de alerta cuando la cantidad de

memoria utilizada está próxima a llegar a su límite, esto permite personalizar el uso de la memoria y optimizar su uso, para no dejar escaso de recursos al sistema operativo.

Una de las características más interesantes encontradas en la elaboración de este proyecto, es la posibilidad que ofrecen algunos motores de poder definir, al momento de crear una tabla, si se desea crear en disco duro, o directamente en la memoria. Esto sin duda permite optimizar aún más la utilización de los recursos, ya que perfectamente bajo previo análisis, se pueden elegir las tablas más críticas en función de la demanda de uso, y definirlas bajo el esquema de almacenamiento en memoria. De esta manera se puede mejorar el rendimiento de las transacciones y de las consultas.

2. Para la elaboración de este proyecto se han elegido cuatro plataformas de motores de bases de datos, aunque todos trabajan bajo el concepto de almacenamiento en memoria, el funcionamiento interno de cada uno de ellos difiere ligeramente. Por ejemplo, en términos de almacenamiento de datos, SolidDB y H2 ofrecen la posibilidad de guardar la información en memoria o en disco duro. En el caso de SolidDB, cuando se define almacenamiento en memoria, de manera automática la información también se replica el disco duro, ya que SolidDB crea por defecto un archivo con una base de datos física. Por otra parte H2 funciona un poco distinto, al momento de crear conexión con la base de datos se puede especificar si se desea trabajar con almacenamiento en memoria o en disco duro, además se pueden definir algunos parámetros de configuración para el manejo de la memoria; se puede especificar que al cerrar la conexión con la base de datos se libere la memoria que ha sido utilizada, o también se puede especificar para que se mantenga la información en memoria luego de terminar la conexión. En este último caso la información se mantiene, siempre y cuando la máquina sobre la cual está corriendo el servicio no se apague o se reinicie.

   Para los otros dos motores SQLite y DerbyDB estudiados en este proyecto, podemos decir que tienen un manejo de la información completamente en memoria, ambos realizan una liberación de la memoria luego de cerrar el servicio que

administra el motor, por tanto la información se pierde de manera inmediata ya que no se almacena en disco duro.

Es importante tener en cuenta que todos los motores evaluados en este proyecto utilizan una plataforma basada en las características de Java, esto los convierte prácticamente en motores multiplataforma, gracias a la portabilidad que ofrecen los entornos de Java. Además, todos estos motores están basados en el estándar SQL, por lo que las consultas y definiciones de objetos son compatibles entre cada uno de ellos. Por ejemplo, para la definición de tablas y las consultas en esta investigación, solamente se creó un script con dichas definiciones, esos mismos scripts fueron los que se ejecutaron en cada motor para crear los objetos y realizar la ejecución de las consultas, para todos los motores funcionó de manera satisfactoria.

La consola que trae consigo cada uno de los motores evaluados está basada en línea de comandos, a excepción de H2, que integran una interface gráfica basada en un navegador web. Sin embargo, en el caso de los demás motores no tiene mayor complejidad para su uso, y más bien consiste en una herramienta muy simple que se caracteriza por su sencillez, debido a que las opciones son muy básicas. Vale la pena buscar algún IDE que permita interactuar de una manera más gráfica con el motor de base de datos, a fin de poder visualizar más fácilmente los objetos creados en la base de datos.

3. Los distintos motores de bases de datos elegidos para realizar las pruebas de rendimiento fueron seleccionados tomando como referencia un aspecto importante que en la mayoría de las organizaciones poseen, esto es, la instalación sobre un sistema operativo no muy complejo de mantener y de configurar. Es por ello que se procuró la operatividad de cada motor sobre el sistema operativo Windows, sin embargo, y como se mencionó anteriormente, cada uno de los motores tiene su conjunto de archivos de instalación para distribuciones de Linux, y en algunos casos sobre Mac OS.

   Se pudo observar de esta manera que todos los motores, exceptuando a H2, que como fue mencionado presentó problemas a la hora de realizar las cargas masivas, se encuentran bien estables con respecto al sistema operativo donde fueron

instalados y configurados. Esto deja en claro que, en un ambiente con características y bondades similares al equipo seleccionado, no habría inconveniente alguno para la correcta operación y ejecución de las distintas operaciones a realizar.

Otro aspecto a considerar y de manera muy importante es las prestaciones con respecto a memoria del equipo a utilizar, ya que como se pudo observar con las pruebas de rendimiento, la utilización de la memoria crece significativamente debido a la naturaleza de las herramientas investigadas. De esta manera, entonces, una vez elegido el motor es necesario realizar estimaciones con respecto al consumo de este recurso para así preparar y contar con un equipo con las prestaciones requeridas para el tipo de procesamiento de información que se realizará por parte de la organización.

Por otra parte, la necesidad de contar con una distribución de archivos, particiones y demás del equipo a utilizar no es tan importante, debido a que la utilización de almacenamiento no volátil (disco duro) no se ve tan afectada como sí sucede con las bases de datos tradicionales, las cuales por lo general almacenan archivos de data, metatadata y log. Esto hace indispensable contar con bondades y prestaciones suficientes de disco duro en el equipo de instalación.

4. La recolección y volumen de los datos para la aplicación de pruebas de rendimiento en estos tipos de motores, los cuales se basan en el uso de memoria para su procesamiento, debe ser lo suficientemente representativa, esto debido a que al ser motores con un procesamiento mucho más veloz con respecto a los tradicionales, las cargas y consultas no hubiesen sido posibles de medir debido a la rapidez con la que se concluyen los procesos que se ejecutaron. En Costa Rica no muchas organizaciones procesan cantidades de datos con un volumen similar al visto en el capítulo anterior, sin embargo, los escenarios definidos involucraron realizar un estimado de registros a emplear, de tal manera que no se llegara a extremos, esto es, ni tan poca cantidad que no permitiera su medición ni tanto como para que se perdiera de vista la realidad de las empresas de nuestro país.

   Verificar de manera correcta el equipo empleado para la aplicación de las pruebas es de extrema importancia, esto porque los motores deben liberar los recursos por

completo entre un escenario y otro. Es decir, cada vez que es ejecutado un proceso al finalizar este, el siguiente motor a evaluar debe disponer del equipo como si no se hubiese ejecutado ninguna sentencia, a fin de obtener resultados íntegros utilizando escenarios lo más puros y limpios posibles que no afectarán ni alterarán ningún dato de salida.

Los datos de salida en toda prueba realizada deben ser almacenados de manera inmediata una vez finalizada cada ejecución de sentencia, esto con la finalidad de observar el estado final real en el cual se encuentra el equipo, y compararlo con el estado inicial previo a la ejecución. Esto es un aspecto del cual se debe tomar control, ya que el monitoreo de los recursos indicará, como bien se conoce, el rendimiento entre cada motor evaluado. De este modo, entonces, es indispensable contar con herramientas que entreguen un detalle adecuado del estado de los recursos, con la finalidad de poder recopilarlos y realizar todo el análisis correspondiente.

5. La naturaleza de la investigación demanda realizar toda una estructuración detallada de los resultados obtenidos de las pruebas aplicadas a cada uno de los motores, con la finalidad de poder graficar de una manera más representativa y visual que permita apreciar de mejor modo el rendimiento presentado con cada sentencia ejecutada. Es de esta forma que se recopilaron los datos en tablas unificando valores de medición obtenidos de las herramientas de monitoreo usadas antes, durante y después de la ejecución de las pruebas de rendimiento, procurando observar las variables más representativas y de interés particular con respecto al consumo de recursos del equipo donde se encuentre instalado el motor.

   Es bien sabido que este es uno de los aspectos más perseguidos, después del rendimiento con respecto al tiempo de respuesta ante cada una de las sentencias ejecutadas. Aunque la velocidad de respuesta sea muy efectiva, en ocasiones el consumo de recursos se hace exponencial, lo que hace casi imposible la liberación del equipo para la ejecución de otras tareas en paralelo, y deja casi fuera de línea su disponibilidad para otras operaciones que son vitales también para la organización en su diario quehacer. De esta forma, es de vital importancia contar con un

panorama claro y bien ilustrado de los resultados, con el cual se pueda apreciar el contraste de un motor con otro.

En ocasiones es necesario realizar una homologación de datos que permita obtener un estimado de los resultados que puedan ser generados, es decir, para aquellos casos donde las pruebas no puedan completarse por aspectos técnicos fuera del alcance, como errores en la herramienta, debe realizarse una proyección o cálculo de los valores que probablemente pueden ser obtenidos. Esto hace que para el caso particular no se tenga que desechar la prueba, sino que más bien pueda incluirse haciendo la salvedad del problema obtenido, a fin de tener en cuenta que son valores calculados y no reales que puedan afectar la toma de alguna decisión.

6.2. Recomendaciones

Recordemos que el propósito de este proyecto siempre fue dar al lector una idea, a grandes rasgos, del funcionamiento de los motores de bases de datos evaluados. Por tanto para realizar una implementación de algún motor de bases de datos en memoria (ya sea alguno de los evaluados en este proyecto, o cualquier otro de los tantos que existen en el mercado), recomendamos encarecidamente realizar un análisis basado en el volumen de datos que se requieren procesar. La mayoría de las veces los sistemas generan información transaccional que no es tan necesaria para el análisis de procesos relacionados con el negocio, como por ejemplo logs de transacciones, donde se guarda información de quién ingresó al sistema, o quién modificó un registro, esta quizás sea información muy importante para auditoría de los sistemas, pero no tanto para tomar decisiones de negocio. Optimizar la información es importante, un análisis de este tipo permitirá clasificar mucha información, lo que sin duda reducirá la cantidad de espacio que finalmente será utilizado por la memoria.

Una vez identificada la información con la que se trabajará, es sumamente importante realizar pruebas de rendimiento con el motor de bases de datos seleccionado, recordemos que cada ambiente es único. Para las pruebas recomendamos utilizar como base las plantillas para la evaluación de rendimiento que se han definido en este proyecto, son una base con una serie de aspectos y características que nos tomó tiempo ajustar y seleccionar, es algo que ya está probado y sabemos que funciona, además son fáciles de modificar. Se pueden incluir nuevas características, lo cual permite personalizar la información que se desea recolectar.

Muchos de los motores de bases de datos en memoria investigados para el desarrollo de este proyecto tienen características de configuración mediante parámetros de iniciación, por lo que durante el arranque de los servicios se establecen dichas configuraciones. Recomendamos investigar ampliamente estas propiedades antes de crear alguna configuración adrede, si bien es cierto son propiedades que ayudan a optimizar el uso de los recursos de hardware y definir umbrales para emisión de alertas antes de que los recursos

lleguen a un punto límite, si no se tiene un amplio conocimiento de lo que se está haciendo se puede convertir en un arma de doble filo y fácilmente puede ocasionar que el rendimiento del motor se vea afectado significativamente.

Algunos motores no ofrecen la característica de replicación en disco duro, por lo que se debe tener especial cuidado, ya que la pérdida de información sensible puede perjudicar el desempeño de los procesos de negocio. Para este tipo de motores sin replicación en disco, se recomienda por ejemplo para un modelo de reportes, que se alimente por ejemplo de una base de datos tradicional, de tal forma que cuando el modelo de datos en memoria inicie, lea la información transaccional y sea cargada en memoria; de esta manera se puede agilizar algunos de los reportes más críticos. Esto por supuesto trae consigo trabajo extra de configuración y de programación que se debe realizar previamente a la implementación, ya que se deben diseñar todos los procesos de carga de información y replicación a memoria para mantener la integridad de la información generada por los sistemas transaccionales.

Se recomienda el uso de H2 en casos donde se utilice volúmenes de datos pequeños, como se mencionó en el capítulo anterior, por las pruebas realizadas en este proyecto, pareciera que H2 tiene una limitante con respecto a la manera en que administra la memoria. Después de varios intentos de subir la información, mediante carga masiva, no fue posible cargar el set de datos que se tenía previsto para la ejecución de las pruebas. Aún con un set de datos más pequeño que el que se cargó en el resto de los motores, resultó ser un motor poco eficiente con las consultas.

SolidDB es un motor de base de datos relacional en memoria capaz de satisfacer demandas de rendimiento de aplicaciones en tiempo real, manteniendo la confiabilidad, por otra parte ha sido reconocido por poder suministrar datos con extrema velocidad. Existen muchas implementaciones de SolidDB en diversos nichos de mercado, entre los más destacados se encuentran redes de telecomunicaciones, aplicaciones empresariales, software y sistemas integrados. Algunas compañías líderes de mercado como Siemens, CISCO, HP, Nokia y Alcatel las utilizan para aplicaciones críticas.

Por otra parte, H2 y DerbyDB son motores de bases de datos en memoria capaces de procesar de manera eficiente volúmenes de datos, existen implementaciones de

aplicaciones empresariales y de software que se apoyan en este tipo de motores de bases de datos, como por ejemplo sistemas de marcaciones, sistemas de controles y accesos, para bloqueos de puertas magnéticas y accesos controlados a sitios restringidos. Es un motor ideal para pequeñas aplicaciones web o del tipo cliente servidor, que no necesitan de un gran poder de procesamiento en términos de hardware, y utilizan lenguajes de desarrollo como Java, PHP, Microsoft .Net.

SQLite es una base de datos muy particular, ya que es una base de datos incrustada y sin servidor, es ideal para aplicaciones de teléfonos, aplicaciones móviles y PDA aplicaciones estructuradas para Windows 10 (loT Core), ya que la plataforma universal de Windows (UWP) es compatible y la recomienda para el almacenamiento local en todas las familias de dispositivos de Windows 10. SQLite se puede usar en reproducción multimedia, y aplicaciones de juegos, también puede usarse como formato de archivo para almacenar catálogos de datos, como por ejemplo guardar niveles de juegos, para mantener el estado y luego ofrecer la posibilidad de descargarlo nuevamente a dispositivos móviles y reanudar el juego desde el último nivel guardado. Así mismo, las plataformas principales en las que SQLite se encuentra funcionando son Windows 95, 98, ME, 2000, XP y Vista, Windows CE & Pocket PC, Mac OSX, Linux, OpenEmbedded, PalmOS, Symbian.

Capítulo 7. Reflexiones Finales

Realmente no hay una receta puntual que permita elegir una tecnología en particular, se debe tener muy claro que cada escenario es único, y depende mucho de la lógica de negocio utilizada en el ambiente, para el cual se desea implementar un motor de base de datos con características de almacenamiento en memoria. Sin embargo, la amplia gama de opciones que se ofrecen en el mercado permite tener un abanico de posibilidades en donde se puede elegir alguna de las herramientas que mejor se ajuste a la necesidad puntual que se tiene.

Ciertamente no es fácil elegir una herramienta que se ajuste a una necesidad específica, que permita solucionar el problema. Lo ideal es hacer pruebas en ambientes replicados, ojalá lo más semejante posible a los de producción, para poder acercar las pruebas lo más que se pueda a lo que realmente sucede en la vida real.

Configurar una herramienta desconocida muchas veces representa un reto en términos de implementación, ya que ello casi siempre significa un costo adicional monetario, en cuanto tiempo y recursos utilizados. No obstante, todo este esfuerzo al final se convierte en una inversión de conocimiento adquirido. Solo el tiempo nos irá dando el conocimiento y la experiencia para realizar los análisis correspondientes de los resultados a las pruebas ejecutadas en cada uno de los motores de bases de datos seleccionados, a fin de elegir el que mejor se adapte al problema puntual que se desea solucionar.

Capítulo 8. Trabajos a Futuro

Un proyecto que puede ser considerado, para su ejecución y desarrollo, corresponde a realizar pruebas de rendimiento de igual forma en motores de bases de datos tradicionales, de modo que se logre establecer un contraste y comparación con respecto a los resultados obtenidos de las pruebas en motores de bases de datos en memoria. La experiencia que se posee al trabajar con motores tradicionales de cierta manera deja la noción de un rendimiento muy satisfactorio con respecto a las analizadas en esta investigación, sin embargo, es necesario poseer resultados medibles y no únicamente contar con nociones o perspectivas de un mejor desempeño en cada tarea ejecutada.

Otro aspecto que sería de gran importancia realizar es el desarrollo de aplicaciones, a modo de cliente-servidor, que consuman las bases de datos en memoria evaluadas, de modo que se pueda observar el rendimiento real a la hora de la ejecución de las consultas y demás operaciones de mantenimiento, ya sea sobre volúmenes grandes de registros o datos específicos. Este aspecto sería de gran utilidad debido a que se entraría en un aspecto meramente de comunicación con el motor de base de datos, obteniendo un panorama lo más cercano a la realidad en cuanto a la operación de un sistema con datos más representativos, los cuales están sujetos a análisis por parte de usuarios que esperan resultados de manera inmediata.

Capítulo 9. Referencias

Alvira Martín Francisco. (1985). La investigación evaluativa: Una perspectiva experimentalista. Consultado el 02 de junio de 2016. Revista Española de Investigaciones Sociológicas. Sitio web: http://www.jstor.org/stable/40183087

Anguiano Morales Jorge Daniel. (2014). Característica y tipos de bases de datos. Consultado el 20 de junio de 2016. IBM México. Sitio web: https://www.ibm.com/developerworks/ssa/data/library/tipos_bases_de_datos/

Apache Derby. (2006). Apache Derby Project Charter (2016). Consultado el 15 de agosto de 2016. Sitio Web: https://db.apache.org/derby/derby_charter.html

Apache Derby. (2006). Embedded Derby. Consultado el 15 de agosto de 2016. Sitio Web: https://db.apache.org/derby/papers/DerbyTut/embedded_intro.html

Berkowitz Brian T, Sreenivas Simhadri, Peter A. Christofferson, Gunnar Mein (2002). In-memory database system. Consultado el 10 de julio de 2016. Microsoft Corporation. Sitio Web: https://www.google.com/patents/US6457021

Burch Martin (2016). IBM developerWorks. Troubleshooting CI 12.1.2 - 'Windows error 2 occurred while loading the Java VM' on Java 1.8.0_60 and later. Consultado el 19 julio de 2016. Sitio Web: https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20TRIRIGA1/page/Troubleshooting%20CI%2012.1.2%20-%20'Windows%20error%202%20occurred%20while%20loading%20the%20Java%20VM'%20on%20Java%201.8.0_60%20and%20later

Filein Rómmel (2010). SQLite: La Base de Datos Embebida. Consultado el 26 de febrero de 2017. Sitio Web: https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida#.WM12NG81_IU

García-Molina Hector, Kenneth Salem (1992). Main Memory Database Systems: An Overview. Member, IEEE. Consultado el 10 de junio de 2016. Sitio Web: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=180602

H2 (2016). Advanced. Consultado el 24 de enero de 2017. Sitio Web: http://www.h2database.com/html/advanced.html

H2 (2016). Features. Consultado el 24 de enero de 2017. Sitio Web: http://www.h2database.com/html/features.html

IBM Knowledge Center (IBM solidDB 6.5). Data types (2016). Explanation of column values in the data type tables. Consultado el 15 de junio del 2016. Sitio Web: http://www.ibm.com/support/knowledgecenter/SSPK3V_6.5.0/com.ibm.swg.im.soliddb.sql.doc/doc/data.types.html

Instituto Politécnico Nacional. Fundamentos de la Computación. Consultado el 16 de agosto 2016. Sitio Web: http://www.sites.upiicsa.ipn.mx/polilibros/portal/polilibros/p_terminados/PolilibroFC/Unidad_VI/Unidad%20VI_2.htm

Java Platform Standard Edition 8 Documentation (2016). Oracle Java Documentation. Consultado el 17 de septiembre del 2016. Sitio Web: http://docs.oracle.com/javase/8/docs/

Kwok Leon. (2010). Conceptos básicos sobre DB2: Limitaciones. Consultado el 17 de agosto del 2016. IBM. Sitio Web: http://www.ibm.com/developerworks/ssa/data/library/techarticle/dm-0401melnyk/

Microsoft (2017). Bases de datos SQLite. Consultado el 23 de febrero de 2017. Sitio Web: https://docs.microsoft.com/es-es/windows/uwp/data-access/sqlite-databases

MME section, IBM Knowledge Center (IBM solidDB 6.5). (2016). MME parameters. Consultado el 30 de julio del 2016. Sitio Web: http://www.ibm.com/support/knowledgecenter/SSPK3V_6.5.0/com.ibm.swg.im.soliddb.admin.doc/doc/mme.section.html

Pardo Clemente Ezequiel. (2004). Microinformática de Gestión. Consultado el 17 de setiembre del 2016. Universidad de Oviedo. Servicio de Publicaciones. Sitio Web: https://books.google.co.cr/books?id=RMDWpaaJ0m8C

Russinovich Mark (2016). Windows Sysinternals, Process Explorer v16.12. Consultado el 22 de octubre del 2016. Sitio Web: https://technet.microsoft.com/en-us/sysinternals/bb896653.aspx?tduid=(dceb16761cdb1caf4ea9a1b98a3dc4bb)(256380)(2459594)(TnL5HPStwNw-.PFHfQuff27IPhegd274eg)()

SolidDB. (2016). In-memory database optimized for extreme speed and extreme availability. Consultado el 18 de enero de 2017. Sitio Web: https://teamblue.unicomsi.com/products/soliddb/?cm_mc_uid=04851643030014565387758&cm_mc_sid_50200000=1489856091

SQLite, Small. Fast. Reliable. Choose any three (2006). About SQLite. Consultado el 05 de septiembre de 2016. Sitio Web: https://www.sqlite.org/about.html

SQLite, Small. Fast. Reliable. Choose any three (2006). Architecture of SQLite. Consultado el 10 de septiembre de 2016. Sitio Web: https://www.sqlite.org/arch.html

SQLite, Small. Fast. Reliable. Choose any three (2006). Features of SQLite (2006). Consultado el 10 de septiembre de 2016. Sitio Web: https://www.sqlite.org/features.html

Tamayo y Tamayo Mario. (2004). El Proceso de la Investigación Científica: Incluye evaluación y administración de proyectos de investigación. Consultado el 10 de junio de 2016. México, Limusa. Sitio Web: https://books.google.co.cr/books?id=BhymmEqkkJwC&printsec

Wolter Peter. Head of OTTO's e-commerce Solutions & Technology Division. OTTO Reinvents E-commerce Personalization for More than 2 Million Visitors Per Day. Alemania. Consultado el 26 de agosto 2016. Sitio Web: https://www.mongodb.com/customers/otto

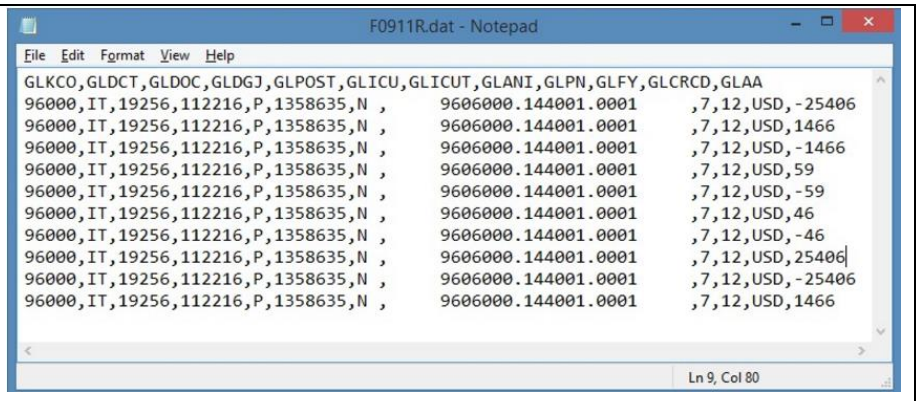
Capítulo 10. Anexos

Seguidamente, se incluye las sentencias SQL utilizadas durante los procesos de carga para cada uno de los motores de base de datos evaluados.

Scripts de Inserts Masivos	
SolidDB	--/**********\ --** filename * delim.ctr --** description * Sample controlfile for SOLID SpeedLoader --** * Uses delimited fields. --** Copyright (C) Solid Information Technology Ltd 1999 --**********/ OPTIONS (SKIP=0,ERRORS=5) -- Skip the first ten records. Stop if errorcount reaches five. LOAD DATE 'YYYY-MM-DD 00:00:00' -- The date format in the data file INFILE 'F0911R.dat' -- The data file INTO TABLE F0911M FIELDS TERMINATED BY X'2C' -- Field terminator is HEX ',' == 2C -- This line could also be: FIELDS TERMINATED BY ','

	OPTIONALLY ENCLOSED BY '[' AND ')' -- Field 2 in row 12 begins with [. (GLKCO, GLDCT, GLDOC, GLDGJ, GLPOST, GLICU, GLICUT, GLANI, GLPN, GLFY, GLCRCD, GLAA) solload -c D:\CargaMasiva "tcpip 2315" dba dba F0911M.ctr
H2	INSERT INTO F0911M SELECT * FROM CSVREAD('F:\CargaMasiva\00 - DATA\F0911R.dat');
SQLite	.mode csv .import "C:/CargaMasiva/00 - DATA/F0911R.dat" F0911M
DerbyDB	CALL SYSCS_UTIL.SYSCS_IMPORT_TABLE (null,'F0911M','C:\CargaMasiva\00 - DATA\F0911R.dat',',','%',null,0);

Estructura del Archivo de Carga

SolidDB	
H2	
SQLite	
DerbyDB	