Universidad de las Ciencias Informáticas

Facultad 3

“Estrategia de Migración hacia Sistemas de Gestores de Base de Datos Libres”.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Antonio Mauri Garcia**.**

Abel Pérez Falcón**.**

Tutor: Ing. Daniel Fernández Arencibia.

Tutor Asesor: MSc. José Raúl Rodríguez Galera.

Co-Tutora: Ing. Dinella Aguilera González.

Co-Tutor: Ing. Disnier Alberto Camejo Domínguez.

Junio 2008

LO QUE PUEDES HACER, O HAS SOÑADO QUE PODRÍAS HACER, DEBES COMENZARLO. LA OSADÍA LLEVA EN SÍ, GENIO, PODER Y MAGIA.

GOETHE

DECLARACIÓN DE AUTORÍA

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

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

_____
Antonio Mauri Garcia
Autor.

_____
Abel Pérez Falcón
Autor.

_____
Ing. Daniel Fernández Arencibia
Tutor.

DATOS DE CONTACTO

Nombre y Apellidos de Tutor: Daniel Fernández Arencibia.

E-mail: darencibia@uci.cu

Curriculum: Ingeniero Informático, graduado de Universidad de las Ciencias Informáticas en el año 2007. Actualmente es profesor adjunto del departamento de Ingeniería de Software de la Facultad #3. Desde el inicio del curso 2007-2008 funge como asesor de la Dirección Técnica de la Infraestructura Productiva en la Universidad de la Ciencias Informáticas.

Nombre y Apellidos de Co-Tutora: Dinella Aguilera González.

E-mail: daguilera@uci.cu.

Curriculum: Profesora graduada de Ing. Ciencias Informáticas en el año 2007. Ha impartido asignaturas como Sistemas de Base de Datos y Ética Informática. Posee categoría docente de instructor recién graduado en adiestramiento. La línea de Investigación a la que pertenece es la Gestión de proyectos.

Nombre y Apellidos de Co-Tutor: Disnier Alberto Camejo Domínguez.

E-mail: dcamejo@uci.cu

Curriculum: Profesor graduado de Ing. Ciencias Informáticas en el año 2007. Asesor de Tecnología Facultad 3.

AGRADECIMIENTOS

Este trabajo pudo ser escrito gracias al apoyo, colaboración y paciencia de muchas personas. Por lo mismo es probable que olvidemos mencionar a algunos de ellos en estas líneas; a ellos les pedimos disculpas.

Agradecemos a la Universidad de las Ciencias Informáticas, por darnos la oportunidad de formarnos en ella durante los últimos cinco años y hacer de nuestros sueños una realidad.

A nuestros padres, amantes de sus hijos, y con esperanza desmedida de la utilidad y significado de la formación académica, nos apoyaron moral y espiritual en todo momento.

A nuestro Tutor e instructor recién graduado Daniel Fernández Arencibia, que con su seriedad científica debidamente matizada de criticidad y humor, a nuestro Tutor Asesor MSc. José Raúl Rodríguez Galera, nuestra Co-Tutora Ing. Dinella Aguilera González y a nuestro Co-Tutor Ing. Disnier Alberto Camejo Domínguez, quienes hicieron de esta experiencia académica algo muy satisfactorio.

A los profesores que han compartido sus experiencias durante estos años. Siempre les estaremos agradecidos.

A todos los que de una forma u otra, nos han ayudado durante la realización del trabajo: amigos, compañeros, en fin gracias a todos.

Con cariño y admiración gracias a todos.

DEDICATORIA

A mis padres por hacer de mi lo que soy. No hay una sola línea de este trabajo donde no estén presentes. Nunca me alcanzará el tiempo para hacer por ustedes lo que han hecho por mí. Siempre han confiado ciegamente en todo lo que he hecho y hacer que se sientan orgullosos ha sido el camino que he tomado desde que tuve conciencia de ello. Son los mejores padres del mundo y saben que nunca les fallaré. Vivan muchos años más que los necesito aún.

A una persona muy importante, quien ha estado conmigo en todo momento, impulsándome a luchar, a no renunciar a mis sueños, ayudándome a confiar y creer en mí, como lo ha hecho y enseñándome el verdadero sentido de lo que es amar.

A mis seres queridos, a todos que con su vida y ejemplo forjaron el amor y los valores que ahora forman parte de mí y que pese a no estar presentes nunca más a mi lado físicamente, me continúan guiando día a día y viven y se enorgullecen de mis triunfos y a ellos les dedico este trabajo.

Antonio Mauri Garcia

DEDICATORIA

A mis padres, gracias por su amor y por ser la luz que me guió durante todos estos años. Espero que se sientan orgullosos de mi como me he sentido yo de ustedes todos estos años por haberme educado con esos principios sin los cuales hoy no podría ser lo que soy.

A mi querida novia Mavis, gracias por soportarme y apoyarme durante estos dos años que han sido maravillosos y por enseñarme lo que es una relación seria basada en el amor.

A mis amigos, en especial mis compañeros de la Vocacional, los mejores amigos que he tenido en mi vida. No importa lo lejos que estemos, siempre seremos uno.

A mis familiares y a todos los que de una forma u otra han estado presente durante esta larga carrera que es la vida.

Abel Pérez Falcón

PROPÓSITO

El documento forma parte de la Solución Informática presentada a la Facultad 3 con el propósito de alcanzar la migración hacia Sistemas Gestores de Base de Datos Libres.

ALCANCE

Está encaminado como futura guía de referencia para Sistemas Informáticos que lleven a cabo la migración hacia Gestores de Base de Datos Libres.

RESUMEN

El presente documento es un reflejo detallado de la propuesta de una Estrategia de Migración hacia Sistemas Gestores de Base de Datos Libres, encaminada a lograr la soberanía tecnológica y económica.

Se lleva a cabo la definición de una estrategia a seguir para puesta en marcha de una correcta migración hacia Sistemas Gestores de Base de Datos Libres.

Se hace alusión a serie de herramientas, propuestas para hacer más robusto el trabajo con el Gestor de Base de Datos PostgreSQL 8.2 y facilitar el trabajo de los administradores de la base de datos.

Se propone además el diseño de mecanismos que garanticen la administración de Sistemas Gestores de Base de Datos Libres en entornos Grid Computing.

PALABRAS CLAVE

Gestores de Base de Datos, software libre, estrategia, migración.

TABLA DE CONTENIDOS

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

El ser humano ha requerido desde tiempos inmemoriales de medios que le permitan realizar cálculos con los más disímiles propósitos y para el procesamiento de la información. La sofisticación de estos medios se ha incrementado desde esos propios momentos en tanto ha ido incrementando el volumen de información a procesar y su complejidad, de acuerdo con sus necesidades siempre crecientes, y como es lógico ha estado estrechamente relacionada y subordinada al progreso de la tecnología. La evolución de estos ha transitado de los medios más sencillos hasta las potentes computadoras de la actualidad, protagonistas de la era de la información. En los momentos actuales, el uso de estas computadoras es uno de los componentes fundamentales del proceso de modernización de la sociedad a nivel universal y del cual no escapa Cuba, insertada ya, con un protagonismo creciente y sólido, en dicho proceso.

En la década del 80 con el lanzamiento al mundo del sistema operativo MS – DOS (Microsoft Disk Operating System) el gigante azul norteamericano se puso a la delantera en la producción de software, posición que se consolidó con la implementación de Windows, sistema operativo con interfaz gráfica, mucho más atractiva que la tradicional "cara" de fondo negro y letras blancas, y línea de comandos de MS–DOS. Cuba asumió esta plataforma informática para el trabajo en todas las tareas en las cuales(al inicio muy pocas, actualmente prácticamente todas) se utilizan computadoras. La utilización de este sistema operativo plantea notables desventajas, dadas primordialmente por las siguientes razones:

- La primera, por ser un software propietario, o sea, que exige el pago para su adquisición y por la licencia de uso.
- En segundo lugar y derivada de esta, la dificultad, en términos de imposibilidad, de contextualizar la mayoría del software producido para este sistema, para su adecuación a las necesidades de una esfera en específico.

- En tercer lugar, por los bugs o errores de programación y vulnerabilidades que posee su núcleo, escondidos en su código y que resulta imposible corregir por cuanto no se cuenta con el código fuente, problemas que se enfatizan sobre todo desde el momento que Cuba se conecta a Internet en octubre de 1996; y la demora en la aparición de los parches o actualizaciones para su corrección.

De acuerdo al inusitado desarrollo tecno-científico que se ha experimentado en los años recientes, el cual ha estado signado bajo la égida de la globalización y la dominación imperialista por parte de los EE.UU. Es indispensable para el resto de los países del mundo diseñar estrategias para romper con la trampa tecnológica que se impone tanto con la fabricación de componentes bajo condiciones de patentes restrictivas para su libre fabricación, y que intentan un total amarre de por vida a la dependencia con ese solo proveedor o grupo de proveedores, como también con la trampa tecnológica con la cual el capitalismo salvaje pretende obligar a todos en su estrategia de plusvalía intensiva a pagar miles y miles de veces por un mismo programa por medio del cobro de licencias por procesador. Incluso potencian estas estrategias comerciales con las que pretenden hacer saltar presupuestos, obligando a pagar todos los años las actualizaciones de licencias, combinando las innovaciones del hardware con la necesidad obligada de utilizar las nuevas versiones de software.

Otro blanco de los problemas con el software propietario lo han sido las Bases de Datos, en lo que respecta a la libertad de información. La regla estipulada por los dueños de software propietario declaraba: «si usted comparte con su vecino, usted es un pirata. Si desea algún cambio, ruéguenos para que lo hagamos nosotros».1

Para un estudiante de Ingeniería Informática lo importante puede ser conocer la gestión de las bases de datos y no el producto comercial. Teniendo un profundo conocimiento de los conceptos asociados a los gestores de bases de datos es relativamente sencillo adquirir los conocimientos

[1] . Declaraciones de STALLMAN que apoyan la difusión del *software* libre, poniéndolo en pugna contra el propietario, aparecen publicadas por JUAN GONZALO en un artículo titulado “Richard Satllman alerta sobre el peligro de monopolios de las bases de datos”, sito en www.abiertos.org/modules.php?name.

necesarios para gestionar adecuadamente una base de datos. En este caso, a la hora de hacer las prácticas siempre será preferible usar una base de datos libre que el estudiante puede instalar en cualquier equipo que desee de forma legal, que usar productos propietarios.

Por lo anteriormente expuesto los autores manifiestan que la ausencia del uso de Sistemas Gestores de Base de Datos Libres que no permite lograr una soberanía tecnológica. Por lo que la introducción de estos gestores en la facultad favorece la soberanía tecnológica y con ello la eliminación del pago por concepto de licencia y soporte debido al uso de estas herramientas.

Como resultado de lo planteado se explicita el siguiente **Problema Científico de investigación:** orientado a ¿Cómo lograr la soberanía tecnológica en la utilización de Sistemas Gestores de Bases de Datos?

El **objeto de estudio:** Sistemas Gestores de Base de Datos.

El **objetivo general** de este trabajo es diseñar una estrategia para alcanzar la migración. Para darle cumplimiento a este objetivo se plantean los siguientes **objetivos específicos**:

- Diseñar una estrategia para la migración hacia Sistemas Gestores de Base de Datos Libres.
- Proponer la integración de nuevas funcionalidades a los Sistemas Gestores de Base de Datos Libres.
- Proponer el diseño de mecanismos que garanticen la administración de Sistemas Gestores de Base de Datos Libres en entornos Grid Computing.

El **campo de acción:** estrategia para la migración hacia Sistemas Gestores de Base de Datos Libres.

Los investigadores plantean como **hipótesis** que si se desarrolla una estrategia para la migración hacia Sistema Gestores de Base de Datos Libres, entonces se logrará la eliminación de pagos por concepto de licencia y soporte, lográndose de esta forma una soberanía tecnológica total.

Se establecen entonces las siguientes **tareas de la investigación**:

- Estudio bibliográfico sobre la administración y configuración de Sistemas Gestores de Base de Datos en Oracle Database, MySQL y PostgreSQL.
- Comparación de las funcionalidades y ventajas de los Sistemas Gestores de Base de Datos Libres y Propietarios para la integración y alcance la migración.
- Sistematización sobre los entornos Grid Computing Flexible.
- Análisis de la Licencia de Software Libre: Licencia Publica General (GPL)
- Diseño de la estrategia para la migración hacia Sistemas Gestores de Base de Datos Libres.

Métodos científicos de investigación

Teórico:

Histórico-Lógico: permitieron el análisis de la trayectoria completa del fenómeno y ponen de manifiesto la lógica interna de su desarrollo, además de hallar el conocimiento más profundo de su esencia.

Análisis: permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio.

Síntesis: Establece mentalmente la unión entre las partes previamente analizadas, lo cual posibilita descubrir sus características generales y las relaciones esenciales entre ellas.

Hipotético Deductivo: A partir de la hipótesis y siguiendo reglas lógicas de deducción se llega a nuevos conocimientos y predicciones, las que posteriormente son sometidas a verificaciones empíricas.

Empírico:

Observación: La observación científica es la percepción planificada, dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado

Entrevistas: La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos o sobre características personales del entrevistado y puede influir en determinados aspectos de la conducta humana por lo que es importante una buena comunicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

Introducción

En el capítulo se hace mención a un grupo de conceptos que serán referenciados en el resto del documento. Se define brevemente el concepto de Base de Datos. Se hace una breve mención acerca de un tema muy polémico e interesante como lo es Software Libre. Además se define brevemente qué es un Software de Base de Datos haciendo un breve enfoque a productos como Oracle Database 10 g, MYSQL y PostgreSQL.

Como elemento esencial en el desarrollo del trabajo se definen un grupo de aspectos relacionados con Software Libre y Software Privativo que permitirán definir una estrategia encaminada a alcanzar una futura soberanía tecnológica en la industria del software cubano.

1.1 Definición de Base de Datos.

El término de Base de Datos fue escuchado por primera vez en 1963, en un simposio celebrado en California EUA; con los progresos en la adquisición y almacenamiento de datos en diferentes ambientes como gobierno, industria y aplicaciones científicas, se arrojan como resultado el surgimiento de enormes bases de datos, cuyo tamaño se incrementa rápidamente, tanto en número de registros como en la dimensionalidad de los mismos, llegando a alcanzar proporciones de almacenamiento del orden de los terabytes. La necesidad de explorar estas bases de datos y extraer información y conocimiento que sea de interés para los propietarios de las mismas, se ha incrementado en la actualidad.

Una Base de Datos o Banco de Datos se puede definir como un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. La utilización de bases de datos como plataforma para el desarrollo de Sistemas de Aplicación en las Organizaciones se ha

incrementado notablemente en los últimos años, esto se debe a las ventajas que ofrece su utilización, algunas de las cuales se comentarán a continuación:

- ✓ **Globalización de la información**: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.
- ✓ **Eliminación de información inconsistente**: Si el sistema está desarrollado a través de archivos convencionales, una cancelación de compra por ejemplo, deberá operarse tanto en el archivo de facturas del Sistema de Control de Cobranza como en el archivo de facturas del Sistema de Comisiones.
- ✓ **Permite mantener la integridad de la información** como cualidad altamente deseable y tiene por objetivo almacenar correctamente la información.
- ✓ **Independencia de datos** como factor esencial en la rápida proliferación del desarrollo de Sistemas de Bases de Datos. La independencia de los datos implica un divorcio entre programas y datos.

Definiciones

Las siguientes son definiciones que se necesitan para manejar con claridad las bases de datos relacionales.

Tupla:

- ✓ Es una hilera o fila en una tabla.

Atributo:

- ✓ Es una columna en una tabla.

Dominio:

- ✓ Es el conjunto de valores de los cuales los atributos obtienen sus valores.

Llave:

- ✓ Es un atributo con una característica de relevancia para identificar la tupla.

Llave primaria:

- ✓ Es una llave con valores únicos, es decir, no ocurren más de una vez en el atributo.

Cardinalidad:

- ✓ Es el número de tuplas en una tabla.

Grado:

- ✓ Es el número de atributos en una tabla.

Relación:

Una definición simple es que se corresponde con una tabla y en ocasiones es preferible pensarlo de esta manera. La definición canónica es que una relación es el producto cartesiano de dos o varios dominios.

Podemos también decir que un dominio es un conjunto de valores escalares del mismo tipo, dónde un valor escalar es la mínima unidad semántica de información en el sentido de que son valores atómicos.

Tabla base:

Es una relación autónoma, a diferencia de las vistas y las tablas intermedias construidas a partir de una consulta.

Vista:

Es una relación virtual, que se construye a partir de tablas base o incluso otras vistas, formada por atributos de estas otras tablas de forma directa o como resultado de una consulta.

1.2 Definición de Sistema de Bases de Datos

Un Sistema de Bases de Datos (SBD) es una serie de recursos para manejar grandes volúmenes de información, sin embargo no todos los sistemas que manejan información son bases de datos.

Un sistema de bases de datos debe responder a las siguientes características:

- ✓ **Independencia de los Datos**. Es decir, que los datos no dependen del programa y por tanto cualquier aplicación puede hacer uso de los datos.
- ✓ **Reducción de la Redundancia**. Llamamos redundancia a la existencia de duplicación de los datos, al reducir ésta al máximo conseguimos un mayor aprovechamiento del espacio

y además evitamos que existan inconsistencias entre los datos. Las inconsistencias se dan cuando nos encontramos con datos contradictorios.

- ✓ **Seguridad**. Un SBD debe permitir que tengamos un control sobre la seguridad de los datos.

1.3 Definición de Administrador de Base de Datos (DBA).

El DBA es la persona encargada de definir y controlar las bases de datos corporativas, además proporciona asesoría a los usuarios y ejecutivos que la requieran.

Las principales funciones del administrador son:

- ✓ **La estructura de la base de datos** en el sentido de determinar qué información va a ser necesario almacenar en la misma, después de haber analizado los requerimientos de los usuarios.
- ✓ **Los estándares** por los que se va a regir la organización en cuanto a documentación de la base de datos, metodologías de diseño de la misma.
- ✓ **La estrategia** de transición del sistema existente al nuevo sistema de información soportado en una base de datos. El DBA deberá decidir sobre la posible puesta en marcha en paralelo del nuevo sistema con el antiguo, las fases de implantación del mismo y los controles necesarios. Todas estas decisiones habrán de tomarse en función de los objetivos marcados y de forma que se cause el mínimo trastorno a los usuarios.
- ✓ **Los permisos de explotación y uso**, es decir, establecer la normativa necesaria para la utilización de la base de datos, el modo de solicitar el acceso a la misma, su actualización, etc.
- ✓ **Los aspectos relativos a la seguridad**, incluidos los procedimientos de control y las auditorías.
- ✓ **Mantenimiento rutinario**. Algunos ejemplos de actividades rutinarias que el administrador de la base de datos debe revisar que se cumplan son:

1- Copia de seguridad periódica de la base de datos, bien sobre cinta o sobre servidores remotos, para prevenir la pérdida de datos en caso de desastres o imprevistos.
2- Asegurarse de que exista suficiente espacio libre en el disco duro para las operaciones normales y aumentar el espacio en el disco en caso de ser necesario.
3- Supervisión del los trabajos que se ejecuten sobre la base de datos y sobre todo asegurarse que el rendimiento no se degrade por tareas muy costosas realizadas por algunos usuarios.

Para que el DBA pueda cumplir con todas estas funciones deberá interactuar con todo el personal de la organización como se muestra en la figura:

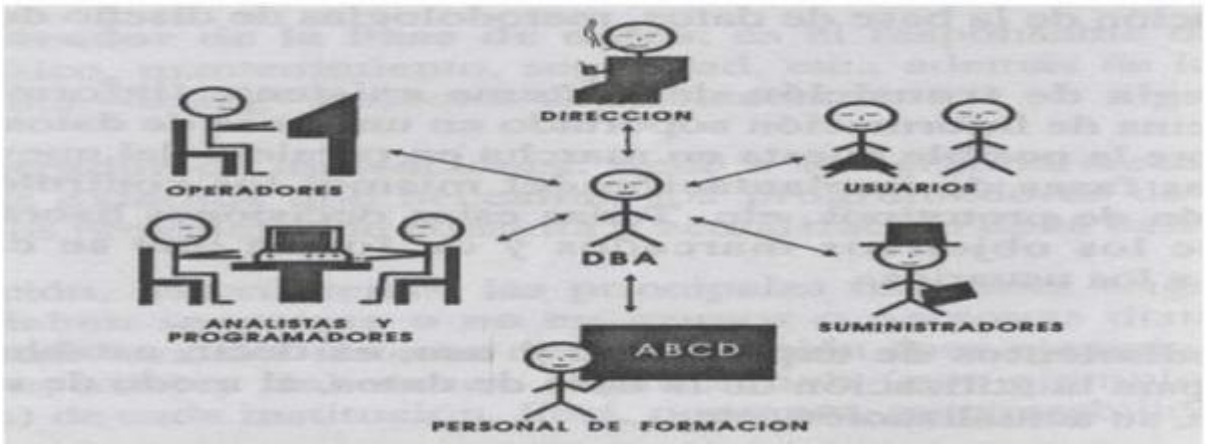

Figura 1 Diagrama de interacción del DBA con los usuarios.

1.4 Definición de Software Libre

La Free Software Foundation (Fundación de Software Libre) es una organización creada en Octubre de 1984 a partir del esfuerzo de Richard Matthew Stallman y otros entusiastas del software libre con el propósito de difundir este movimiento.

El “Software Libre" es un asunto de libertad, no de precio. Para entender el concepto, se debe pensar en “libre" como en “libertad de expresión", lo que ha dado lugar a cierta confusión. “Software Libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar,

cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- ✓ La libertad de usar el programa, con cualquier propósito.
- ✓ La libertad de estudiar cómo funciona el programa, y adaptarlo a necesidades especificas. El acceso al código fuente es una condición previa para esto.
- ✓ La libertad de distribuir copias, con lo que puedes ayudar a tu vecino.
- ✓ La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, se debería tener la libertad de distribuir copias, sea con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y en cualquier lugar. El ser libre de hacer esto significa (entre otras cosas) que no se tiene que pedir o pagar permisos.

La libertad para usar un programa significa la libertad para cualquier persona u organización de usarlo en cualquier tipo de sistema informático, para cualquier clase de trabajo, y sin tener obligación de comunicárselo al desarrollador o a alguna otra entidad específica. Para que las libertades de hacer modificaciones y de publicar versiones mejoradas tengan sentido, se debe tener acceso al código fuente del programa. Por lo tanto, la posibilidad de acceder al código fuente es una condición necesaria para el software libre. Son aceptables, sin embargo, ciertos tipos de reglas sobre la manera de distribuir software libre, mientras no entren en conflicto con las libertades centrales. Por ejemplo, copyleft [“izquierdo de copia"] (expresado muy simplemente) es la regla que implica que, cuando se redistribuya el programa, no se pueden agregar restricciones para denegar a otras personas las libertades centrales. Esta regla no entra en conflicto con las libertades centrales, sino que más bien las protege.

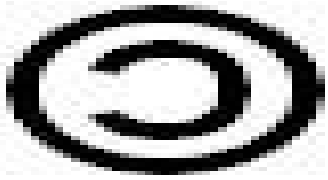

Figura 2 Símbolo del Copyleft

“Software libre'' no significa “no comercial''. Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial. El desarrollo comercial del software libre ha dejado de ser inusual; el software comercial libre es muy importante.

En el proyecto GNU, se utiliza el término “copyleft'' para proteger de modo legal estas libertades para todos. Pero el software libre sin “copyleft'' también existe. Se cree que existen razones importantes por las que es mejor usar copyleft, pero si aparecen programas como software libre sin ser copyleft, se pueden utilizar de todos modos.

Existen varias categorías de Software Libre, entre ellas están:

- ✓ «Software libre»
- ✓ «Código abierto» («Open Source»)
- ✓ «Software de dominio público»
- ✓ «Software protegido con copyleft»
- ✓ «Software libre no protegido con copyleft»
- ✓ «Software cubierto por la GPL»

Entre otras.

1.5 Beneficios del Software Libre

Razones económicas

Ahorros importantes al liberarse del pago de licencias y especialmente por la replicación casi gratuita de aplicaciones comunes a toda la administración pública. El muy bajo costo del software permitirá la ejecución de programas.

Independencia tecnológica

El Estado deja de depender de terceros (a menudo transnacionales) para el diseño, desarrollo y mantenimiento de sus sistemas de información, retomando el control total de sus procesos, en particular de los procesos críticos y de alta importancia estratégica para el país.

Control de la información

El acceso al código fuente, la libertad de inspeccionar el funcionamiento del software, la libertad de decidir la manera en que se almacenan los datos y la posibilidad de modificar cualquiera de estos aspectos queda en manos del Estado, lo cual le permite el control total de la información y por consiguiente el ejercicio de la Soberanía Nacional.

Confiabilidad y estabilidad

El software libre realizado por comunidades está sometido a la inspección de un importante número de personas, este número de verificadores es mucho mayor que el del software propietario. Estas personas identifican los problemas, los resuelven, y comparten las soluciones con los demás. Por tal razón los programas libres de las comunidades gozan de gran confiabilidad y estabilidad.

Seguridad

La información que el Estado maneja generalmente es importante y/o confidencial, puede ser muy peligroso que esta información caiga en manos incorrectas. Por esta razón es imprescindible que el Estado pueda verificar que su software no tenga puertas de entrada traseras, voluntarias o accidentales, y que pueda cerrarlas en caso de encontrarlas; tal control sólo es posible con el software libre.

Desarrollo País

Se genera Transferencia Tecnológica hacia los actores nacionales productores de software, acelerando el Desarrollo Endógeno [2]y reforzando la Soberanía Nacional.

1.6 Licencias de Software. Generalidades

Desde la perspectiva legal, el software puede presentarse en dos estados, ya sea protegido por el Derecho de Autor o pertenecer al dominio público. A partir del Convenio de Berna [3]se adoptó el no requerimiento de formalidades para que se considere una obra protegida, para esto solo se necesita crear. No se debe olvidar que a pesar de ello el registro de la obra brinda una presunción legal de autoría muy útil para resolver casos de plagio o litigios.

Una de las grandes diferencias entre la propiedad tradicional y la intelectual es su duración. Mientras que la primera se extiende ilimitadamente en el tiempo la segunda no. Esta limitación temporal se basa en el criterio de que el autor crea, basándose, utilizando o inspirándose en el conocimiento que lo precede y es justo entonces que su obra en el algún momento abandone el ámbito de su poder para que pase al servicio de la sociedad.

---

[2] "Generación de desarrollo apoyado en la creación de tejidos (cadenas interdependientes) de producción y consumo, que basados en las potencialidades del territorio y en el manejo interno de la tecnología, produce bucles de acumulación por dentro cada vez mayores en los distintos eslabones de cada cadena, produciendo así el incremento sostenido de capitales". Disponible en: http://www.aporrea.org/actualidad/a4851.html.

[3] *Convenio de Berna para la Protección de las Obras Literarias y Artísticas*, 28 de septiembre de 1979. Disponible en: http://www.wipo.int/treaties/es/ip/berne/pdf/trtdocs_wo001.pdf. Visitado el 28 de diciembre de 2007.

Es en esta temporalidad de la protección en la que entra el concepto de dominio público, el cual constituye el status legal que adquiere una obra luego que se ha extinguido el tiempo de duración de los del autor. A partir de ese momento cualquier persona puede disponer libremente de la obra siempre y cuando se respeten determinados derechos morales.

En realidad, los derechos morales duran indefinidamente atendiendo a la concepción continental del derecho de autor, que percibe al creador como ente físico e individual y poseedor de derechos que emanan de su condición, por lo que solo se trasmiten los derechos patrimoniales e incluso después de la muerte dos derechos morales persisten: el reconocimiento de la paternidad y el respeto a la integridad de la obra.

El movimiento del software libre aboga por la plena libertad del usuario a la hora de utilizar el software y entonces la lógica indicaría que la mejor forma de alcanzarlo es que el autor libere el software poniéndolo desde el primer momento a disposición del dominio público. Es aquí donde se palpa una vez más las diferencias entre los subsistemas copyright y el derecho de autor. Mientras en el primero el autor puede renunciar totalmente a sus derechos sobre la obra (porque los derechos morales no se consideran) en el segundo tales derechos morales son irrenunciables y los únicos que podrían cederse serían los derechos patrimoniales.

Si bien bajo el sistema anglosajón es posible dicha práctica, no es compatible con la ideología que proclama el movimiento del software libre. Como cualquier persona puede realizar acciones con el software cuando este se encuentra bajo dominio público puede convertirlo en software propietario, cerrando el acceso al código fuente y liberándolo bajo una licencia restrictiva de uso que eliminaría las libertades que proclama el software libre.

Gramaticalmente una licencia es un permiso o una autorización. MARESCA considera que las licencias de software "no son sino ofertas de acuerdo realizadas por el autor o titular de la obra, que si son aceptadas por un usuario o explotador del software, pasan a convertirse en contratos"[4]. Otros autores[5] la ven como el "instrumento legal que más emplean los

[4] Maresca, *op.cit.*

[5] Malcom Bain, Manuel Gallego Rodríguez, Manuel Martínez Ribas y Judit Rius Sanjuán, *Aspectos legales y de explotación del software libre*, Universidad Abierta de Cataluña, 2004 (edición digital)

desarrolladores de software para distribuirlo a terceros” y luego especifican más su naturaleza jurídica al clasificarlo como “un contrato, al consistir en un acuerdo de voluntades entre proveedor de software y usuario”.

Es necesario partir de una idea básica. Los derechos de autor brindan el control y disposición casi absoluta de la obra al autor, sin embargo este casi nunca crea para su consumo íntimo o personal sino que quiere que su obra esté disponible para otras personas. Sin embargo la utilización que pueden realizar terceros debe respetar sus derechos como autor y por tanto él debe especificar las condiciones de utilización.

Una licencia de software es el instrumento legal a través del cual el autor, titular o proveedor (licenciante) establece las condiciones y los requisitos generales bajo los que se otorga el Software al usuario (licenciatario).

El licenciante en ese documento plasmará los derechos y deberes que tiene el usuario para con él y viceversa. Se tratarán cuestiones como la garantía y también la forma de solucionar conflictos en caso de incumplimiento. Nada de lo que se ha dicho constituye una obligación en el momento de redacción del instrumento, simplemente el licenciante puede ser más o menos explícito con este pronunciamiento. De cualquier forma lo aconsejable es tratar de regular en la mayor medida posible todas las cuestiones legales posibles previniendo así presunciones o libres interpretaciones por parte del usuario. De las definiciones de licencia de software hay un aspecto que no queda despejado. Se trata de establecer la delimitación sobre la consideración de la licencia como oferta o como contrato.

La oferta viene a ser un precontrato o proyecto de contrato, que una vez que el sujeto al que va dirigido lo acepta se constituye como tal. Sin embargo la doctrina entiende que la oferta debe ser completa y ese no es el caso de las licencias de software. Existe un aspecto esencial que no se contempla en ninguna licencia y es el precio del producto. Este es un elemento que no se puede obviar. Si bien se ha dicho que el licenciante tiene plena libertad a la hora de redactar sus condiciones se debe suponer que el no pronunciamiento con respecto al precio significa que el software es gratuito, la lógica lleva a esa suposición. En la práctica no se manifiesta de esta forma. Por una parte se encuentra la licencia de software y en otro documento se establece el

precio del mismo y se consigna el nombre del adquiriente, fecha y otros particulares como puede ser el modo de solución de los conflictos en caso de incumplimiento por las partes. De manera que lo que realmente existe es una dualidad de documentos que en su conjunto sí constituyen un contrato una vez que el usuario accede a comprar el software. De este modo la licencia en sí misma es sólo una parte del contrato que ni siquiera se debe denominar oferta por no incluir todos los elementos que requiere.

En cuanto al software libre resulta necesario distinguir desde ahora un aspecto de la terminología que se utiliza en relación a las licencias y para ambos modelos de desarrollo y comercialización de software y es el término “licencia de uso”. Resulta totalmente entendible su uso en el caso del modelo propietario, en el cual las restricciones que impone el licenciante al mismo dejan solo la posibilidad del mero uso (legal) del software. Algo muy distinto ocurre en el software libre al brindar una serie de libertades, dirigidas a modificar ese anterior orden de cosas, liberando al software al permitir ya no solo su uso (ejecución) sino también su estudio, modificación, distribución. En el caso del software propietario, como en el de cualquier otro fruto de la creación intelectual, se adquiere la propiedad tradicional sobre el continente físico que la porta y no sobre la obra en sí. Si se tiene un CD de música o un libro es dueño de estos y puede disponer, es decir, enajenar a terceros, lo que no será posible es reproducir esa obra o hacerle adaptaciones o modificaciones sin el consentimiento del autor. En el caso del software propietario el precio del producto incluye el valor del soporte material (tradicionalmente disquetes o discos compactos) y otro tanto que se abona como contraprestación al permiso de uso que le brinda el propietario sobre el software. De hecho en algunos casos las restricciones que impone el software propietario abarcan hasta esa enajenación a terceros e incluso a la propiedad suya sobre el soporte material.

1.7 Licencias Libres

La licencia libre es aquella que está redactada de forma tal que busca garantizar la mayor cantidad o grado de libertad del usuario. Pueden existir tantas como programas se realicen y se

quieran liberar teniendo esa meta de libertad. Ahora bien, debe brindar al usuario, como mínimo, las cuatro libertades que se refieren a la facultad de usar, estudiar, adaptar-mejorar y distribuir el software.

La diferencia fundamental entre las licencias libres radica en cláusulas adicionales y no en las cuatros libertades fundamentales que las enmarcan. Como ejemplo de esas cláusulas adicionales están:

- ✓ Período de tiempo por el que se pone a disposición el código fuente al licenciatario cuando no viene con el ejecutable. La Licencia Pública General establece un período mínimo de tres años, la licencia libre bajo la que se libera el navegador de Internet Mozilla establece 12 meses[6].
- ✓ Si se contempla la forma de resolver los conflictos en caso de incumplimiento de una de las partes.
- ✓ Forma en que se realizan las modificaciones y posteriores redistribuciones. Puede imponer determinadas condiciones, como es el caso de la BSD (Berkeley System Distribution).

La más conocida, tal vez por ser la primera en sentido estricto es la Licencia Pública General del proyecto GNU. Se hará referencia a ella como GPL (General Public License) o como GNU/GPL por ser sus siglas en inglés las más conocidas y utilizadas.

Producto que cada titular puede distribuir su software bajo las condiciones que él disponga la variedad de licencias es muy grande, por tanto se hará referencia a una clasificación de las licencias del software libre que se basa en la transmisibilidad, o no, de esa libertad y que es asumida por Microsoft[7].

---

[6] Cláusula 3.2 de la Mozilla Public License. Disponible en: http://www.opensource.org/licenses/mozilla1.0.php. Visitado el 24-11-2007

[7] Ver http://www.microsoft.com/spain/sharedsource/default.mspx. Visitado el 23 de noviembre de 2007.

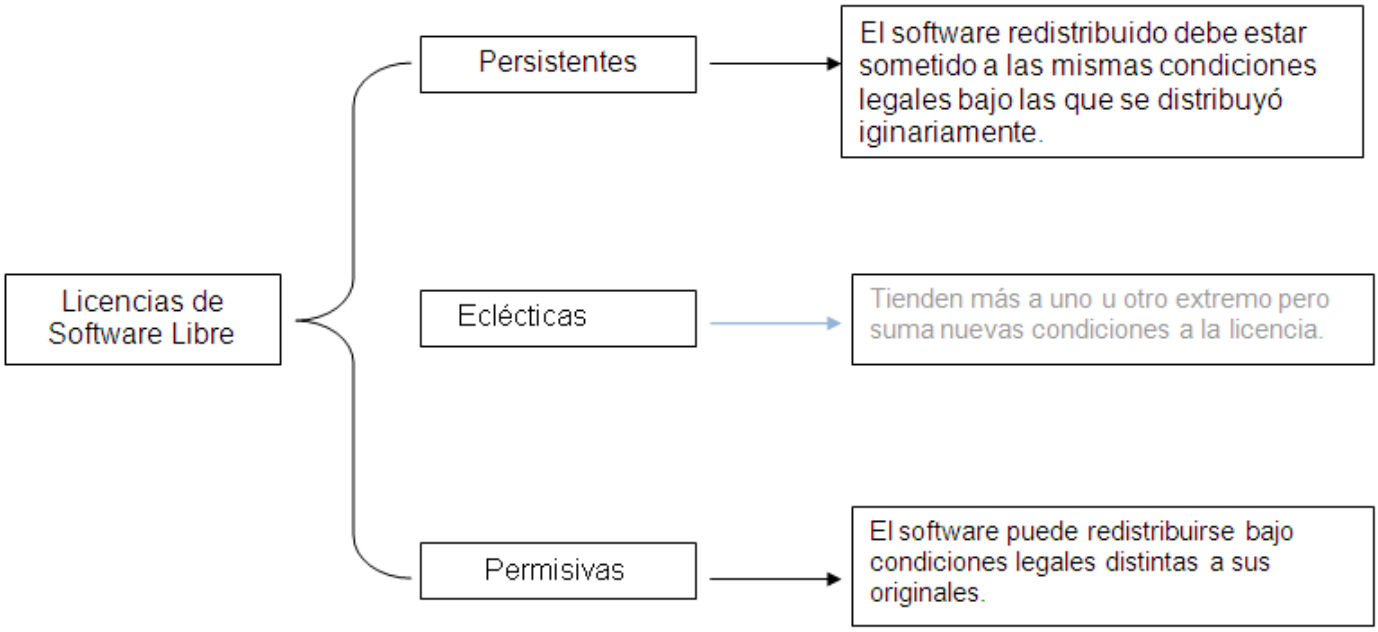

Figure 3 Grupos de Licencias de Software Libre.

Las licencias persistentes son aquellas que no solo le brindan libertad al usuario sino que se aseguran que los adquirientes del programa modificado o simplemente redistribuido lo posean en iguales condiciones que el adquiriente originario. Es una forma de evitar que el software pueda ser convertido en software propietario o cerrado. Los representantes por excelencia de este tipo de licencias son la Licencia Pública General (GPL, que se analizará más adelante) y la Licencia Pública General Reducida (LGPL).

Las licencias permisivas, por su parte, son inicialmente libres y pueden continuar siéndolo, pero ello depende de la persona y de sus intereses, se le denomina permisiva o sin restricciones, porque el adquiriente puede rediseñar o cambiar la licencia que la sustenta y redistribuirlo con las condiciones que él ahora imponga, pudiendo ser convertido con facilidad el software en propietario. El ejemplo más representativo de este tipo de licencia es la Berkeley System Distribution (BSD).

Microsoft lo considera como un modelo de software libre sin restricciones. Al igual que esa corporación existen personas, como el movimiento OSI, que consideran que esa es la verdadera

libertad: brindarla y no controlarla cayendo en una suerte de libertad en estado de anarquía. La realidad es que a aquellos que comparten el criterio del software propietario ven en este modelo la posibilidad de eliminar cualquier competencia. La forma es bien sencilla: se adopta un prometedor programa libre, se le hacen algunas modificaciones y se presenta bajo licencia propietaria. Los consumidores, adaptados al sistema tradicional del mercado del software nunca se enterarían ni entenderían la injusticia de que alguien se apodere del trabajo, mayoritariamente voluntario de a veces cientos de programadores para el beneficio de uno solo incluso cuando ese no era quizás el propósito inicial de ese producto.

Existen numerosas licencias libres que son muy parecidas entre sí (recuérdese que deben garantizar las cuatro libertades básicas) y sólo se diferencian en cuestiones de poca trascendencia. Muchas de esas licencias han sido concebidas para aplicarse a un solo producto informático, una forma rápida de identificarlo está en su nombre: Eclipse Public License, Python License, Mozilla Public License, etc.[8].

1.7.1 Licencia Pública General (GPL)

GNU GPL (Licencia Pública General) es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Existen varias licencias "hermanas" de la GPL, como la licencia de documentación libre GNU (GFDL) que cubre los artículos de la Wikipedia, la Open Audio License, para trabajos musicales, etcétera, y otras menos restrictivas, como la MGPL, o la LGPL (Lesser General Public License o Library General Public License), que permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres. La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se le denomina contrato de licencia o acuerdo de

[8] Puede visitarse sitios como http://www.opensource.org/licenses/, http://www.gnu.org/licenses/licenses.html, http://www.gnu.org/licenses/licenselist.html, en los que es posible acceder al contenido íntegro de varias licencias de Software libre.

licencia. En los países de tradición anglosajona existe una distinción doctrinal entre licencias y contratos, pero esto no ocurre en los países de tradición civil o continental. Como contrato, la GPL debe cumplir los requisitos legales de formación contractual en cada jurisdicción. Usar la GPL de GNU exige que todas las versiones mejoradas que se publiquen sean software libre. Esto significa que evitará el riesgo de tener que competir con una versión modificada privativa de su propio trabajo. No obstante, en algunas situaciones particulares puede ser mejor emplear una licencia más permisiva.

Un aspecto crucial del software libre es que los usuarios tienen la libertad de cooperar. Es absolutamente esencial que a los usuarios que deseen ayudarse entre sí se les permita compartir sus correcciones de errores y mejoras con otros usuarios. Algunos han propuesto licencias alternativas a la GPL que requerirían que las versiones modificadas fueran supervisadas por el autor original. Mientras el autor original permaneciera atento a las necesidades de mantenimiento, esto podría funcionar bien en la práctica; pero si el autor (en mayor o menor medida) tiene que dejar de hacer otras cosas o no atiende las necesidades de todos los usuarios, el procedimiento se viene abajo. Dejando a un lado los problemas prácticos, este planteamiento no permite a los usuarios ayudarse entre sí. En ocasiones, el control sobre las versiones modificadas se propone como un medio de evitar la confusión entre las diferentes versiones hechas por los usuarios. A juzgar por nuestra experiencia, esta confusión no supone mayor problema. La GPL exige al autor de una versión que ponga su nombre en ella, con el objeto de distinguirla de otras versiones y para proteger la reputación de otros responsables del mantenimiento del programa.

¿Exige la GPL que el código fuente de las versiones modificadas se ponga a disposición del público?

La GPL no le obliga a publicar el programa modificado. Usted es libre de hacer versiones modificadas y usarlas en privado, sin tener nunca que hacerlas públicas. Esto es aplicable también a organizaciones (empresas incluidas); una organización puede hacer una versión

modificada y usarla internamente sin hacerla pública fuera de la organización. Pero si hace de alguna manera pública la versión modificada, la GPL le exige que ponga el código fuente modificado, a disposición de los usuarios, bajo la GPL. Así, pues, la GPL le autoriza a publicar el programa modificado, de determinadas maneras y no de otras.

¿Me permite la GPL vender copias del programa a cambio de dinero?

Sí. La GPL autoriza a cualquier persona a hacerlo. El derecho de vender copias es parte de la definición de software libre. Excepto en una situación particular, no existe un límite al precio que puede ponerles. (Esa excepción es la oferta escrita de proporcionar el código fuente, que ha de acompañar a los binarios obligatoriamente cuando estos no se distribuyen junto a su código fuente).

¿Me autoriza la GPL a exigir que cualquiera que reciba el software haya de abonarme una cantidad o notificármelo?

No. En realidad, una exigencia de ese tipo haría que el programa no fuese libre. Si la gente tiene que pagar cuando obtiene una copia del programa, o si tiene que notificárselo a alguien en particular, entonces el programa no es libre. Véase la definición de software libre. La GPL es una licencia de software libre, y por tanto permite a la gente usar e incluso distribuir los programas sin que por hacerlo pueda exigir el abono de ninguna cantidad a nadie. La licencia ha sido reconocida por juzgados en Alemania, particularmente en el caso de una sentencia en un tribunal de Munich, lo que indica positivamente su validez en jurisdicciones de derecho civil. Dentro de varios marcos regulatorios y legislaciones de países alrededor del mundo, la inserción y utilización de la GPL puede encontrar dificultades desde el punto de vista de la aplicación y validez, debido a la rigidez o contravención de leyes y normas en defensa de los derechos de los consumidores. Sin embargo, se puede deducir que la GPL no es un contrato del consumidor ya que el licenciamiento por lo general no asume las características de un negocio.

GPL después de 14 años de ser usada sin cambios, la licencia GNU GPL vuelve a la mesa de diseño. La Free Software Foundation está actualmente enfocada a actualizarla en varios sentidos:

- ✓ Resolver formas en que a pesar de todo alguien podía quitar libertades a los usuarios.
- ✓ Como un caso especial de lo anterior: Prohibir el uso de software cubierto por la licencia en sistemas diseñados para quitar libertades.
- ✓ Resolver ambigüedades y aumentar su compatibilidad con otras licencias.
- ✓ Facilitar su adaptación a otros países.
- ✓ Incluir cláusulas que defiendan a la comunidad de software libre del uso indebido de patentes de software.

El proceso de revisión de la nueva versión de la licencia se inició el 16 de enero de 2006 en el Instituto Tecnológico de Massachusetts (MIT) con la presentación del primer borrador. La versión final de la GPL 3 fue hecha pública el 29 de junio de 2007 y es accesible a través del Portal de GNU.

Resumiendo se puede decir que la GPL:

- ✓ Regula solo la copia, distribución y modificación del software.
- ✓ Considera que el uso del software no está restringido.
- ✓ La cláusula de copyleft se activa cuando usted realiza modificaciones al software y piensa distribuirlo con ellas.

Tal y como se representa en el siguiente gráfico:

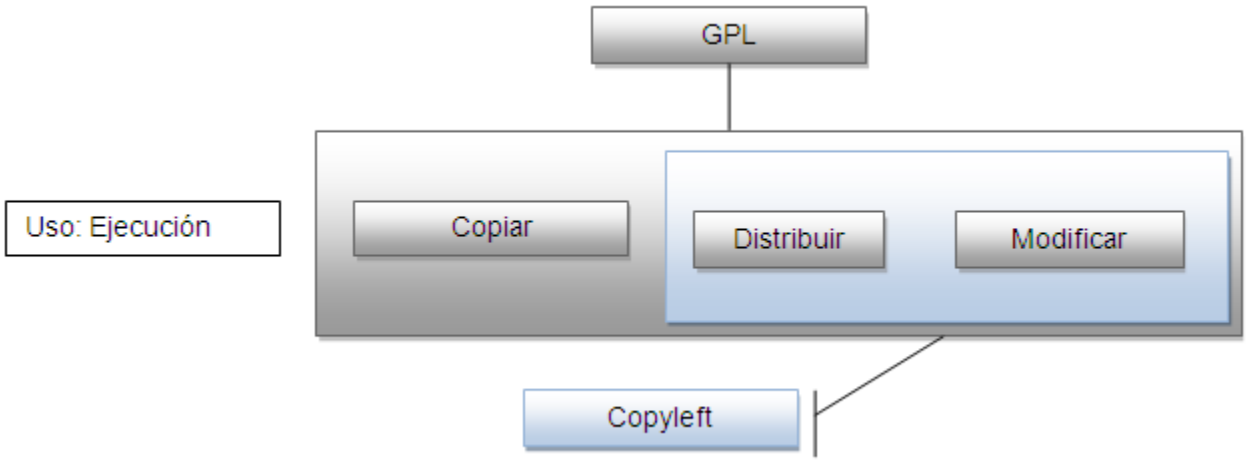

Figure 4 Licencia Pública General (GPL).

1.7.2 Licencia Pública General Modificada (MGPL)

Modified General Public License (inglés: Licencia Pública General Modificada de GNAT), a veces Modified GPL o MGPL, es una versión de la licencia GPL modificada específicamente para los lenguaje de programación genéricos Ada. Dicha modificación permite que un paquete genérico de una biblioteca con esta licencia sea instanciada por una aplicación que no sea GPL. Con el compilador GNAT se puede especificar que un fichero está licenciado bajo la MGPL utilizando la siguiente directiva de compilación:

Pragma License (Modified_GPL);

1.7.3 Licencia Pública Genera Lesser (LGPL)

Los contratos de licencia de la mayor parte del software están diseñados para jugar con su libertad de compartir y modificar dicho software. En contraste, la "GNU General Public License" pretende garantizar su libertad de compartir y modificar el software "libre", esto es para asegurar que el software es libre para todos sus usuarios. Esta licencia pública general se aplica a la mayoría del software de la "FSF Free Software Foundation" (Fundación para el Software Libre) y

a cualquier otro programa de software cuyos autores así lo establecen. Algunos otros programas de software de la Free Software Foundation están cubiertos por la "LGPL Lesser General Public License" (Licencia Pública General Menor), la cual puede aplicar a sus programas también.

Esta licencia se aplica a cualquier programa o trabajo que contenga una nota puesta por el propietario de los derechos del trabajo, estableciendo que su trabajo puede ser distribuido bajo los términos de esta "GPL General Public License". El "Programa", utilizado en lo subsecuente, se refiere a cualquier programa o trabajo original, y el "trabajo basado en el Programa" significa ya sea el Programa o cualquier trabajo derivado del mismo bajo la ley de derechos de autor: es decir, un trabajo que contenga el Programa o alguna porción de él, ya sea íntegra o con modificaciones o traducciones a otros idiomas. Otras actividades que no sean copia, distribución o modificación sí están cubiertas en esta licencia y están fuera de su alcance. El acto de ejecutar el programa no está restringido, y la salida de información del programa está cubierto sólo si su contenido constituye un trabajo basado en el Programa (es independiente de si fue resultado de ejecutar el programa). Si esto es cierto o no depende de la función del programa.

El proyecto OpenOffice.org de Sun Microsystems emplea la LGPL.

1.7.4 Distribución de Software Berkeley (BSD)

La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

Es muy similar en efectos a la licencia MIT. Las versiones más antiguas de la licencia incluían una cuarta cláusula, llamada la cláusula de publicidad. En 1999 esta cláusula fue revocada con efecto retroactivo de las distribuciones BSD de la Universidad de California, Berkeley.

Algunos proyectos incluyen esta cláusula, que aparece como la cláusula 3 en la licencia original (la última cláusula pasaría a ser la número 4):

* 3. Todos los materiales de publicidad que mencionan rasgos o uso de este software

* deben mostrar el reconocimiento siguiente:

* Este producto incluye software desarrollado por <autores>

* Y contribuyentes.

El autor, bajo esta licencia, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación.

Puede argumentarse que esta licencia asegura un “verdadero” software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre. Otras opiniones están orientadas a destacar que este tipo de licencia no contribuye al desarrollo de más software libre.

1.8 Definición de Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos (DDL: Data Definition Language), de un lenguaje de manipulación de datos (**DML**: Data Manipulation Language) y de un lenguaje de consulta (**SQL**: Structured Query Language). En los textos que tratan este tema, o temas relacionados, se mencionan los términos Sistema Gestor de Base de Datos (SGBD) y Sistema Manejador de Base de Datos (DBMS). El propósito general de los Sistemas de Gestión de Base de Datos es el de

manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información.

Índices:

El empleo adecuado de índices en una relación acelera el acceso a la información, pero consume espacio considerable, es por esto que vale la pena hacer un análisis cuidadoso de cuáles atributos requieren ser indexados.

Niveles:

Interno: cómo se almacenan y recuperan los datos (único).

Externo: cómo perciben los datos los usuarios (muchos).

Conceptual: enlace entre los anteriores.

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. Lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia**. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- **Control de la concurrencia**. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados. Tiempo de la base de datos representa el tiempo total gastado en las llamadas de la base de datos, y es un indicador del trabajo del caso total. Como se muestra en la Figura, el tiempo de la base de datos constituye una porción del tiempo de contestación de usuario global de una aplicación.

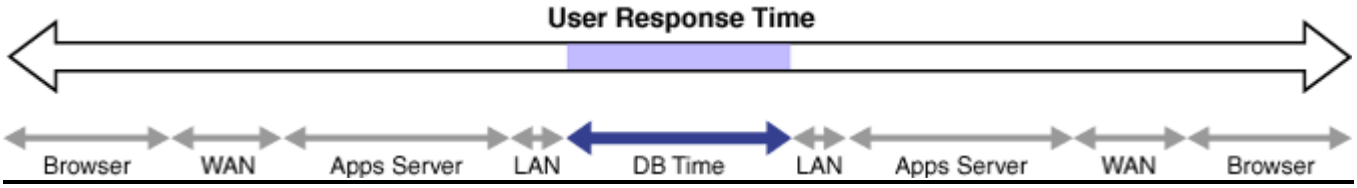

Figura 5 Tiempo de respuesta.

DB time = Suma del tiempo en gastado procesando todas las demandas del usuario = Suma de tiempo (Corriendo en la CPU + Esperando por recursos + Esperando por la CPU).

Ventajas:

1. Facilidad de manejo de grandes volúmenes de información.
2. Gran velocidad en muy poco tiempo.
3. Independencia del tratamiento de información.
4. Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
5. No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
6. Integridad referencial el terminar los registros.

Inconvenientes:

7. El costo de actualización del hardware y software son muy elevados.
8. El costo (salario o remuneración) del administrador de la base de datos es grande.

9. El mal diseño de ésta puede originar problemas futuros.
10. Un mal adiestramiento a los usuarios puede originar problemas a futuro.
11. Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
12. Generan campos vacíos en exceso.
13. El mal diseño de seguridad genera problemas en ésta.

Los SGBD se desglosan en:

1. SGBD Libres.
2. SGBD Gratuitos.
3. SGBD Propietarios.

De estos sólo se tomarán:

1. SGBD Libres.
2. SGBD Propietarios.

Dentro de estos dos grupos se analizarán brevemente:

- SGBD Propietarios.
  - ✓ Oracle.
  - ✓ MySQL.
- SGBD Libres.
  - ✓ PostGreSQL.

Hay que destacar aquí que hasta el año pasado MySQL formaba parte de los Sistemas Gestores de Base de Datos Libres pero que este año pasó a formar parte de la lista de Gestores de Base de Datos Propietarios. Esto estuvo dado debido que Sun Microsystems [9] invirtió un total de 1

[9] Empresa informática de Silicon Valley, fabricante de semiconductores y software. Fue constituida en 1982 por el alemán Andreas Von Bechtolsheim y los norteamericanos Vinod Koshla, Bill Joy, Scott McNealy y Marcel Newman.

billón de dólares, los cuales se distribuyeron en 800 millones en efectivo y otros 200 millones en la adquisición de acciones de la compañía.

1.8.1 Caracterización de la Suite de Productos de Oracle 10g.

Oracle es básicamente un herramienta cliente-servidor para la gestión de bases de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que solo se vea en empresas muy grandes y multinacionales, por norma general. Es un manejador de Bases de Datos Relacional que hace uso de los recursos de los sistemas informáticos en todas las arquitecturas de hardware, lo que permite garantizar su aprovechamiento en ambientes cargados de información, por su capacidad de almacenar y acudir a los datos de forma recurrente. El manejador de Bases de Datos Oracle, surgió a final de los años 70 y principio de los años 80. George Koch y su equipo fueron los primeros en desembarcar en el terreno de Oracle en 1982, durante un proceso de evaluación de sistema de gestión de bases de datos para una importante aplicación comercial. Oracle, conocida entonces como Relational Software, tenía poco más de 25 empleados en aquel tiempo y solo unos pocos clientes importantes. Sin embargo, cuando se completó el estudio, Oracle fue declarada vencedora. George Koch afirmó que el Sistema Gestor de Bases de Datos (SGBD) Oracle era técnicamente el mejor producto del mercado. Actualmente es el mayor y más usado Sistema Manejador de Bases de Datos Relacionales (RDBMS) en el mundo. La Corporación Oracle ofrece este RDBMS como un producto incorporado a la línea de producción. Además incluye varias generaciones de desarrollo de aplicaciones, herramientas de reportes y utilitarios. Oracle es soportado en computadoras personales (PC), microcomputadoras, computadoras con procesamiento paralelo masivo. Soporta unos 17 idiomas, funciona automáticamente en más de 80 arquitecturas de hardware y software distintos sin tener la necesidad de cambiar una sola

---

Las siglas SUN se derivan de «**S**tanford **U**niversity **N**etwork», proyecto que se había creado para interconectar en red las bibliotecas de la Universidad de Stanford. Se hizo famosa por el eslogan *«The network is the computer»* («La red es la computadora». Algunos de sus productos han sido servidores y estaciones de trabajo para procesadores SPARC, los sistemas operativos SunOS y Solaris, el sistema de archivos de red *(*NFS), la plataforma de programación Java y conjuntamente con la Corporación American Telephone and Telegraph (AT&T), la estandarización del UNIX System V Release 4.

línea de código. Esto es porque más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos. Oracle es un sistema comercial que aporta un SGBD que ofrece las particularidades básicas para trabajar en entornos multi-usuario. Como Sistema Gestor de Bases de Datos, es actualmente uno de los paquetes de software más ampliamente extendidos en todas las compañías que tienen que gestionar una cantidad importante de información.

Oracle es uno de los sistemas más conocidos, que alcanza hoy en día un buen nivel de madurez y de profesionalidad gracias especialmente a:

- ✓ Su transportabilidad, funciona hoy en día sobre decenas de plataformas.
- ✓ La potencia de sus instrumentos de desarrollo de aplicaciones.
- ✓ La riqueza de su diccionario de datos.
- ✓ Los mecanismos encargados de la seguridad y la confidencialidad.
- ✓ Una experiencia probada sobre el terreno y una buena presencia de Oracle a nivel de formación, consejo y soporte técnico.

La versión 10g de Oracle salió al mercado en febrero del 2004, primero en su versión para UNIX y posteriormente en sus versiones para Linux y Windows. La novedad más llamativa de esta versión es que pone la "g" en el nombre de la versión. Es la capacidad de estos servidores de funcionar según el paradigma de “Grid” o rejilla. La novedad principal de Oracle 10g descansa precisamente en la preparación de dicho software para poder encajar en este modelo. Además, como parte de esta manera de entender el negocio, Oracle también ha hecho cambios en el área de marketing. Así, la letra "i" que se asociaba desde hace seis años a la marca y que representaba la entrada en la era de Internet de la compañía ha sido sustituida por la "g" de Grid, de tal forma que la anterior versión Oracle 9i cambia a Oracle 10g. No obstante, ciertos usuarios valoran más las mejoras en la administración y la integración de algunos elementos que previamente no funcionaban correctamente juntos. Al instalar la base de datos, también se instala el nuevo Oracle Enterprise Manager Database Control, basado en

la Web, que será la herramienta primaria para manejo de la base de datos y establece un nuevo estándar en cuanto a facilidad de uso, ya que es un entorno visual e intuitivo, sin necesidad de usar el texto como medio de comunicación con la base de datos.

La suite de productos de Oracle contempla las siguientes herramientas:

- **Oracle Database 10g Standard Edition:** Es una base de datos de características completas para pequeñas y medianas empresas que requieren el desempeño, la disponibilidad y la seguridad de la base de datos número uno del mundo a un bajo costo. Disponible en un sólo servidor o en servidores en clúster con hasta cuatro procesadores, Oracle Database 10g Standard Edition es la opción segura para desarrollar e implementar de manera económica las aplicaciones de la base de datos. Este producto está considerado una base de datos multiusuario pero con un número limitado de usuarios. Actualmente existe para Windows, Unix y Linux.
- **Oracle Database 10g Enterprise Edition:** Ofrece confiabilidad, escalabilidad y desempeño de primer nivel para configuraciones en clúster y en un sólo servidor. Ofrece las más completas características para soportar el procesamiento de transacciones más exigente, inteligencia de negocios, y aplicaciones para la administración de contenido. Oracle Database 10g Enterprise Edition brinda desempeño y escalabilidad de récord absoluto para el procesamiento de transacciones y depósitos de datos de gran escala en Windows, Linux, y servidores UNIX.

- **Oracle Enterprise manager 10g Grid Control**: Oracle Enterprise manager 10g Grid Control es la consola central de administración y el entorno que automatiza las tareas administrativas para el conjunto de sistemas implicados en un entorno grid. Esta consola ayuda a reducir los costes de administración; con ella, los DBAs pueden agrupar múltiples nodos hardware como bases de datos, servidores de aplicación, servidores Web etc. como si fueran unidades lógicas. Ejecutando trabajos, diseñando políticas, monitorizando el rendimiento y automatizando muchas otras tareas sobre un conjunto de destinos en vez de

sobre muchos sistemas individuales, la Oracle Enterprise Manager (OEM) Grid Control permite escalar un grid fácilmente. Con OEM Grid Control, Oracle 10g automatiza la instalación, configuración y clonación de servidores de aplicación y de bases de datos sobre múltiples nodos. Este entorno puede utilizarse tanto para la adición de nuevos sistemas como para aplicar parches o añadir utilidades a sistemas ya existentes. También mantiene la sincronía entre los nodos.

- **Real Application Clusters (RAC)**: Oracle Real Application Clusters permite que una única base de datos se expanda por múltiples nodos en un grid o red, uniendo los recursos de varias máquinas. Esto que requería un proceso en versiones anteriores del servidor se puede hacer inmediatamente en Oracle 10g, y se puede empezar a balancear el flujo de trabajo hacia la nueva máquina que se incorpora al grid, a la vez que abandonarla cuando ya no es necesaria. Otros Sistemas de Bases de Datos no pueden hacer esto dinámicamente cuando la Base de Datos se encuentra ejecutándose. El nuevo software de clúster en Oracle 10g simplifica el proceso eliminando la necesidad de adquirir, instalar y configurar estas herramientas de terceros. Se pueden añadir servidores a la vez que eliminarlos en un clúster Oracle sin tiempo de inactividad, es decir, sin detener la base de datos, sin importar tampoco la plataforma donde se encuentra instalado el servidor.

1.8.2 Caracterización de PostGreSQL 8.2

PostgreSQL es un servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos Open-Source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PostgreSQL Grupo Global de Desarrollo (PGDG), sus siglas en ingles se definirían como: PostgreSQL Global Development Group. PostgreSQL ha tenido una larga evolución, comenzando con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fué uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con el mismo,

Michael decidió volver a la Universidad para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES. En proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones -las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En POSTGRES la base de datos "comprendía" las relaciones y podía obtener información de tablas relacionadas utilizando reglas.

A continuación se representan los hitos más importantes en la vida del proyecto POSTGRES.

1986- Se publicaron varios documentos que describían las bases del sistema.

1988 - Ya se contaba con una versión utilizable.

1989 - El grupo liberaba la versión 1 para una pequeña comunidad de usuarios.

1990 - Se liberaba la versión 2 la cual tenía prácticamente reescrito el sistema de reglas.

1991 - Liberación de la versión 3, esta añadía la capacidad de múltiples motores de almacenamiento.
1993 - Crecimiento importante de la comunidad de usuarios, la cual demandaba más características.
1994 - Antes de la liberación de la versión 4, el proyecto termina y el grupo se disuelve.

Después de que el proyecto POSTGRES terminara, dos graduados de la universidad, Andrew Yu and Jolly Chen, comenzaron a trabajar sobre el código de POSTGRES, esto fue posible dado

que POSTGRES estaba licenciado bajo la BSD, y lo primero que hicieron fue añadir soporte para el lenguaje SQL a POSTGRES, dado que anteriormente contaba con su propio lenguaje de consultas, creando así el sistema al cual denominaron Postgres95.

Características Adicionales

- Bloqueos consultivos («advisory locks»): permiten el control de objetos de bases de datos a nivel de aplicación usando el eficiente motor de bloqueos de PostgreSQL.
- pg_dump selectivo: permite extraer vuelcos transaccionalmente consistentes de relaciones, seleccionando la inclusión y exclusión usando expresiones regulares.
- Sentencias preparadas: tiene nuevas interfaces administrativas y mejoras de rendimiento en sentencias preparadas.
- ISN/ISBN: este módulo contiene tipos de datos para soportar descriptores de producto ISN para inventarios y sistemas de venta al por menor. Ha sido aumentado para soportar los últimos estándares internacionales.
- Criptografía: el módulo pgcrypto, soportando criptografía dentro de la base de datos, fue actualizado con los últimos algoritmos.
- Mejoras al SQL: nueva sintaxis, incluyendo UPDATE RETURNING, DROP IF EXISTS, ON COMMIT y nuevos comandos de propiedad («ownership») y permisos, para hacer más fácil el manejo de objetos de la base de datos en la línea de órdenes.
- Extracción de registros por lotes en psql: permite devolver filas a la consola en lotes en lugar de todas a la vez.
- Reconstrucción de ECPG: la interfaz embebida C de PostgreSQL ha sido extensivamente reescrita y se agregaron nuevas interfaces.
- Mejoras importantes en TSearch2: soporte de UTF-8, tesauros, soporte de reescritura de consultas e indexación GIN.
- PL/Python: ahora soporta parámetros con nombre y funciones que devuelven conjuntos.

- MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún Sistema Gestor de Base de Datos (DBMS) con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.
- La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.
- Está disponible en casi cualquier Unix (34 plataformas en la última versión estable), Linux, Solaris, BSDs, Mac OS, Beos, Windows.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).

1.8.3 Caracterización de MySQL 5.0

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C. Al contrario de proyectos como el Apache, donde el software es desarrollado por una comunidad pública, y el copyright del código está en poder del autor individual, MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson, y Michael Widenius.

SQL (Lenguaje de Consulta Estructurado) fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde ese entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL: 92, SQL: 99, SQL: 2003. MySQL es una idea originaria de la empresa Open Source MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

La procedencia del nombre de MySQL no es clara. Desde hace más de 10 años, las herramientas han mantenido el prefijo My. También, se cree que tiene relación con el nombre de la hija del cofundador Monty Widenius quien se llama My. Por otro lado, el nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso “Name the Dolphin”. Este nombre fue enviado por Ambrose Twebaze, un desarrollador de Open-Source Africano, derivado del idioma SiSwate, el idioma local de Swaziland y corresponde al nombre de una ciudad en Arusha, Tanzania, cerca de Uganda la ciudad origen de Ambrose.

Características

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multiplataforma.
- Procedimientos almacenados.
- Disparadores (Triggers).
- Cursores (Cursors).
- Vistas actualizables.
- Soporte a VARCHAR.
- Información de esquema (INFORMATION_SCHEMA).
- Modo Strict.
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle.
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial).
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación (savepoints) con InnoDB.
- Soporte para Secure Socket Layer (capa del enchufe segura).
- Escondiendo Consultas.
- Sub-SELECTs (o SELECTs anidados).

- La repetición con un amo (master) por esclavo (slave), muchos esclavos por amo, ningún apoyo automático para amos múltiples por el esclavo.
- Indexando y buscando campos de texto completos usando el motor de almacenamiento MyISAM.
- Librerías de base de datos embebidos (Embedded Database Library).
- Soporte completo para Unicode.
- Conforme a las reglas ACID (Atomicidad, Consistencia, Invisibilidad, Durabilidad), usando los motores InnoDB, BDB y Clúster.

- Nada compartido o agrupado a través de MySQL Cluster (Shared-nothing clustering through MySQL Cluster).

1.9 Definición de Grid Computing

Es una tecnología innovadora que permite utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado. En este sentido es una nueva forma de computación distribuida, en la cual los recursos pueden ser heterogéneos (diferentes arquitecturas, supercomputadores, clústeres) y se encuentran conectados mediante redes de área extensa (por ejemplo Internet). Desarrollado en ámbitos científicos a principios de los años 1990, su entrada al mercado comercial siguiendo la idea de la llamada Informática de utilidad (Utility Computing) supone una revolución que dará mucho que hablar. El término grid se refiere a una infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos que son propiedad y están administrados por diferentes instituciones. Puesto que la colaboración entre instituciones envuelve un intercambio de datos, o de tiempo de computación, el propósito del grid es facilitar la integración de recursos computacionales. Universidades, laboratorios de investigación, empresas, etc., se asocian para formar grid para lo cual utilizan algún tipo de software que implemente este concepto.

Las características de esta arquitectura son:

- Capacidad de balanceo de sistemas: no hay necesidad de calcular la capacidad de los sistemas en función de los picos de trabajo, ya que la capacidad se puede reasignar desde la granja de recursos a donde se necesite.
- Alta disponibilidad. Con la nueva funcionalidad, si un servidor falla, se reasignan los servicios en los servidores restantes.
- Reducción de costes: Con esta arquitectura los servicios son gestionados por "granjas de recursos". Ya no es necesario disponer de "grandes servidores" y podremos hacer uso de componentes de bajo coste.

1.10 En qué consiste el proceso de migración

¿Qué es una solución de migración?

Es un procedimiento mediante el cual se cambia el actual sistema o gestor de bases de datos por otros productos o por versiones que se adapten mejor a las necesidades de la organización.

¿De que consta el proceso de migración?

Los procesos que involucra la migración varían dependiendo, entre otros factores, del producto actual, de sí la decisión de la nueva base había sido tomada o si es necesario plantear diferentes alternativas, etc.

El proceso de migración consta de los siguientes pasos:

Se analiza la factibilidad de la migración. Si la migración es posible, se hace un respaldo (backup) de información y se procede a la realización de la migración. Se recupera la información y se verificará la adecuada funcionalidad e integridad de la información recuperada.

¿Cuál es el resultado?

Las migraciones traen como resultado el mejor aprovechamiento de la tecnología que se posea y la mayor velocidad de acceso a la información. En muchos casos y dependiendo del tipo de migración, también se logra mayor seguridad y versatilidad.

Casos en los que es necesaria la migración de datos

- ✓ Se ha vuelto muy lenta la carga y consulta de datos.
- ✓ Se debe mantener compatibilidad con otros productos.
- ✓ El volumen de información que se maneja necesita una mejor base de datos.
- ✓ Se desea mantener actualizados los productos.
- ✓ Es necesario cambiar hacia un producto de base de datos sin licencia.
- ✓ Se necesita bajar los gastos en tecnología.
- ✓ Se desea armar una red.
- ✓ Se han comprado nuevos equipos.

El motivo por el cual se arribará a una estrategia de migración en este trabajo de diploma está centrado específicamente en los casos 5 y 6 anteriormente planteados.

1.11 Bases de una estrategia

La estrategia como aporte de la investigación puede ubicarse entre los resultados de significación práctica, ya que la misma tiene como propósito esencial la proyección del proceso de transformación del objeto de estudio desde un estado real hasta un estado deseado.

La estrategia ha sido concebida como manera de planificar y dirigir las acciones para alcanzar determinados objetivos.

La determinación de metas y objetivos a largo, mediano y corto plazo y la adaptación de acciones y recursos necesarios para alcanzarlos son los elementos claves para llevar a cabo la estrategia.

El propósito de toda estrategia es vencer dificultades con una optimización de tiempo y recursos. La estrategia permite definir qué hacer para transformar la acción existente e implica un proceso de planificación que culmina en un plan general con misiones organizativas, metas, objetivos básicos a desarrollar en determinado plazo con recursos mínimos y los métodos que aseguren el cumplimiento de dichas metas.

De lo anterior se infiere que las estrategias son siempre conscientes, intencionadas y dirigidas a la solución de problemas de la práctica.

Diversos autores coinciden al señalar que las estrategias son instrumentos de la actividad cognoscitiva que permiten al sujeto determinada forma de actuar sobre el mundo, de transformar los objetos y situaciones.

Actualmente la estrategia ha encontrado amplia utilización en la actividad productiva, social, política, de dirección. En este ámbito la estrategia se refiere a la dirección de la transformación de un objeto desde su estado real hasta un estado deseado. Presupone por tanto partir de un diagnóstico en el que se evidencia un problema y la proyección y ejecución de sistemas de acciones intermedias, progresivas y coherentes que permitan alcanzar de forma paulatina los objetivos propuestos.

El plan general de la estrategia debe reflejar un proceso de organización coherente unificado e integrado, direccional, transformador y sistémico.

Elementos que están presentes en la estrategia:

1. Existencia de insatisfacciones respecto a los fenómenos, objetos o procesos educativos en un contexto o ámbito determinado.
2. Diagnóstico de la situación actual.
3. Planteamiento, objetivos y metas a alcanzar en determinados plazos de tiempo.

4. Definición de actividades y acciones que respondan a los objetivos trazados y entidades responsables.

5. Planificación de recursos y métodos para viabilizar la ejecución.

6. Prever la evaluación de los resultados.

Conclusiones del capítulo

El capítulo ha sido un breve recorrido por conceptos y definiciones que son parte intrínseca del trabajo, a partir de los cuales nos apoyaremos como puntos de reflexión para posible toma de decisiones. En este documento se analiza cuidadosamente el concepto de Software Libre, para dar una visión de lo importante que puede ser la inclusión del mismo en el país logrando para ello una soberanía de la tecnología con respecto a los Softwares Propietarios. Se centra la atención en las principales licencias libres que existen y las ventajas que proporcionan las mismas, en este sentido nos inclinamos por el uso de la GPL, ya que además de ser la más utilizada, la misma presenta como propósito fundamental declarar que el software cubierto por ésta licencia es software libre, protegiéndolo así de intentos de apropiación que restrinjan esas libertades a los usuarios. Se caracterizan además tres importantes gestores de base de datos que actualmente son muy utilizados en el mundo entero, tales como: Oracle, MySQL (Sistemas Gestores de Base de Datos comerciales) y PostgreSQL.

CAPÍTULO 2: PROPUESTA DE ESTRATEGIA DE MIGRACION.

Introducción

La utilización del software libre, como cualquier cambio tecnológico, no es un camino fácil y no hay manual de tránsito para uso general. No es necesario utilizar software libre, desde un principio no se puede someter a una organización a un esfuerzo que ponga en peligro su actividad. Proponer un cambio tecnológico por encima de la utilización que se le debe dar a la tecnología es una actuación temeraria. Pero sí es posible, y deseable, iniciar un camino que permita su futura adopción sin traumas. La elección de un Sistema de Gestión de Base de Datos (SGBD) es un tema de discusión necesario dentro de los departamentos de sistemas de cualquier organización, institución o empresa. Dicha decisión acarrea consecuencias para la organización, a veces de manera permanente. Los factores que inciden dentro de la decisión de adoptar una u otra plataforma son muy variados y complejos a la vez. Generalmente la inversión a realizar en una solución comercial para base de datos es muy alta, y a veces injustificada. Hasta hace poco tiempo, existían pocas y por tanto muy claras opciones a escoger. Cada opción con características que hacen que se elija con qué paquete quedarse. Sin embargo dicha situación ha cambiado, con el desarrollo de productos Open-Source totalmente gratuitos, que pueden sustituir de manera eficiente a los actuales Sistemas Gestores de Base de Datos (SGBD) propietarios.

Una guía de migración, es un documento que establece la estrategia a seguir para realizar una migración paulatina hacia Gestores de Base de Datos Open-Source. Además establece las etapas de desarrollo de dicho proceso y las tareas específicas involucradas en cada una de ellas, define los responsables de acometerlas, los recursos que deberán ser asignados y el tiempo a emplear en su desempeño, garantizando un flujo de trabajo constante y eficiente. Debe describir en lenguaje técnico claro, cómo se debería llevar a cabo dicha migración.

2.1 Desarrollo

2.1.1 Visión General

De manera general, un proceso de migración ideal debe consistir de los siguientes puntos, agrupados en el número de etapas que se desee y algunos de ellos pueden hacerse en paralelo en dependencia de las disponibilidades que se disponga a la hora de realizar el proceso.

Crear un equipo con la capacitación y el respaldo de gestión adecuados. Es importante que se disponga de apoyo de gestión, pues de lo contrario habrá resistencia al cambio. Este apoyo tendrá que ser suficiente para permitir por lo menos la construcción de pilotos representativos.
Entender el entorno final. Esto expresa la necesidad de formar al personal, contratar personal o recurrir a consultores, lo que implicará algunos costos iniciales y por ello es necesario disponer del respaldo de los responsables de la gestión.
Es muy importante conocer las características del Gestor de Base de Datos Open- Source que se utilizará. Existen algunos aspectos que hay que tener en cuenta antes de tomar cualquier decisión:

- ✓ Conocer las implicaciones de las licencias para SWL especialmente si se considera que la institución va a distribuir los cambios de software.
- ✓ Se deben tener en cuenta las diferencias entre las distintas distribuciones. Algunas distribuciones están respaldadas por empresas comerciales que prestan su apoyo y correcciones.
- ✓ Los ejecutores deben determinar qué nivel de apoyo es necesario. Se puede obtener de los desarrolladores de la aplicación o la distribución si la suministran. Si no es así, consultar a terceros que pueden prestar ese apoyo ya que se dispone del código fuente y hay muchas empresas internacionales que dan ese apoyo.

Esta es una diferencia clara respecto al software propietario donde un apoyo detallado sólo lo facilitan las empresas que tienen el privilegio de acceder al código fuente. Y esto es importante si el vendedor propietario abandona el negocio sin revelar el código fuente.

Elaborar un caso detallado de migración, que se basará en los datos recogidos y que consistirá en los siguientes puntos:

- ✓ El costo del entorno existente en un período de tiempo razonable.
- ✓ El costo de entornos alternativos y el costo de la migración a cada uno de ellos en el mismo período.
- ✓ Los puntos fuertes y débiles del entorno actual y las distintas alternativas.

Comenzar con proyectos pilotos a pequeña escala, de preferencia en un entorno auto-contenido con pocos usuarios. Esto facilitará, entre otras cosas:

- ✓ Datos más ajustados de modelos de costo total de propiedad.
- ✓ La reacción de los usuarios, que se puede emplear para facilitar la introducción.

Transición en grupos: Se pasan los datos del antiguo sistema al nuevo en grupos. Puede que los grupos se trasladen juntos para minimizar tener que compartir datos. Se pueden contener los riesgos y gestionar los recursos eligiendo grupos del tamaño adecuado.

Extender la migración a toda la Organización. Esto implicará más formación del personal técnico.

Supervisar la respuesta de los usuarios y tomar nota de los problemas que surjan. Algunas necesidades de los usuarios pueden ser tan poco claras que no se pueden detectar, ni descubrir, durante los proyectos piloto. Hay que asegurarse de que se dispone de recursos suficientes para hacer frente a esas necesidades tras la transición.

2.1.2 Etapas

Para una mejor comprensión, se utilizará una nomenclatura que divide el proceso de migración en 3 grandes grupos fundamentales: Preparación, Migración y Consolidación, considerando cada etapa como sigue:

- ✓ **Preparación**: Etapa previa al desarrollo del proceso.
- ✓ **Migración**: Etapa en la que se define la estrategia a seguir para el proceso de migración.
- ✓ **Consolidación**: Comprende como tal el proceso de migración y post-migración.

2.1.2.1 Preparación

Como preparación del proceso de migración se definirán 3 grandes fases:

1. **La justificación de la migración,** donde se presentarán:
   - ✓ Los argumentos por los cuales se inicia el proceso de migración hacia Gestores de Base de Datos Open-Source tales como: PostgreSQL.
   - ✓ Las ventajas cualitativas y cuantitativas de los Gestores de Base de Datos Open-Source.
   - ✓ Comparación de costos en términos de licencia y soporte.

2. **La planeación de la migración,** que incluirá:
   - ✓ La realización de un levantamiento informático se lleva a cabo con vistas a detallar el tipo de hardware, software y para prevenir incompatibilidades.
   - ✓ La sensibilización con la realización del proceso y la organización para acometerlo, lo que incluye la elaboración conjunta de un plan de acción que permitirá seguir de cerca y más aún, garantizar el desarrollo exitoso del proceso.
   - ✓ La formación y capacitación de la mayor cantidad de personal posible para intervenir en la realización del proceso y acometer labores de soporte, durante la migración y despliegue de la solución.
   - ✓ La creación y puesta en funcionamiento de un centro de soporte, esto a través de un centro de llamadas, mediante IRC (canales de chat), creación de un equipo para dar soporte o el desarrollo de un sitio web con un foro donde se traten FAQ (preguntas más frecuentes) para dar soporte a los futuros usuarios y que estos estén al tanto de las actualizaciones tanto de software como de noticias y para propiciar el intercambio de ideas, conocimientos y datos.

3. **Las pruebas pilotos**, que contemplarán:

- ✓ La puesta en funcionamiento, a pequeña escala, del plan de migración y todas sus dependencias, permitiendo una valoración en tiempo real del grado de exactitud del mismo, mediante la realización de pruebas pilotos.
- ✓ Chequeo del Plan de acción que se lleve a cabo.

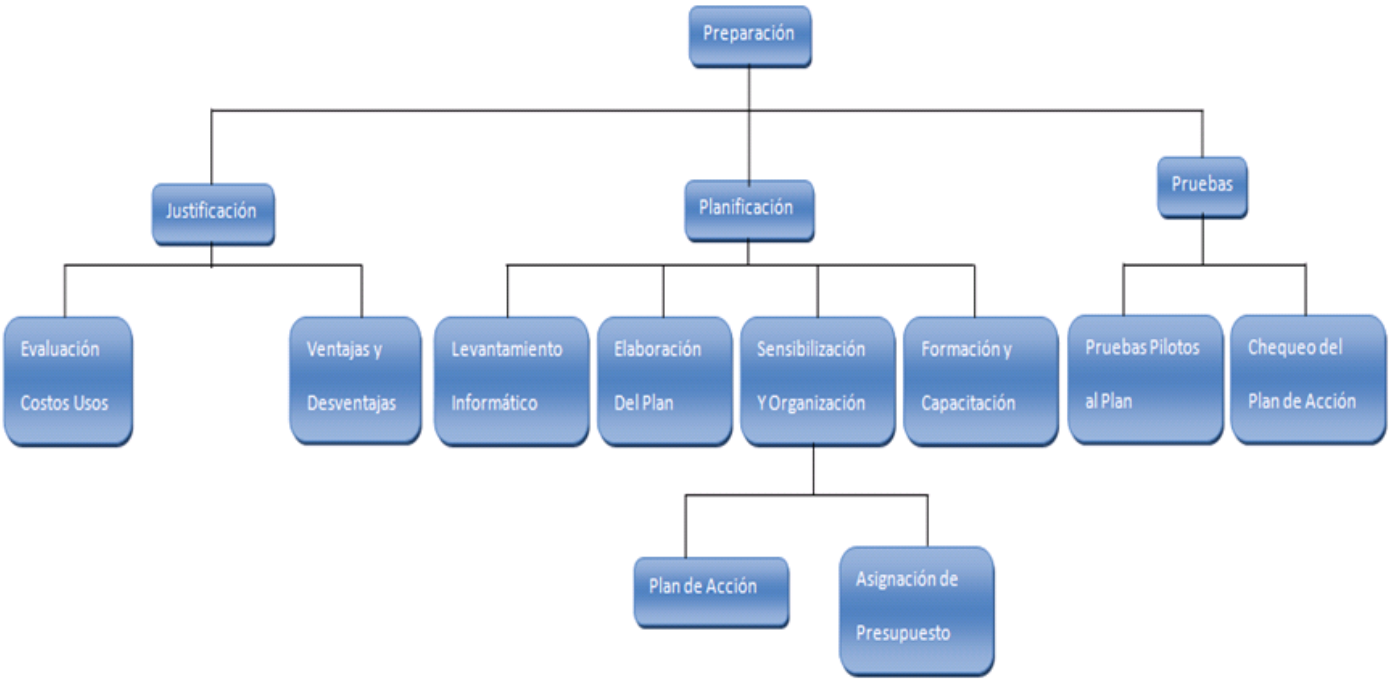

Figura 6 Etapa de Preparación.

2.1.2.2 Migración

Una migración de Base de Datos (BD) es un proceso que se realiza para mover o trasladar los datos almacenados en un origen de datos a otro, para lo cual es indispensable que antes de empezar cualquier proceso de esta naturaleza, se tenga clara y documentada la razón por la cual se está migrando, además de elaborarse la planeación detallada de las actividades contempladas. Dicha migración se requiere llevar a cabo cuando es necesario mover un

esquema dentro del mismo servidor, o de un servidor a otro, así como para actualizar la versión del software, y hacer un cambio de manejador de bases de datos por el de otro fabricante o para cambiarlo a una plataforma de cómputo distinta. Existen diversos motivos para hacer una migración, tales como: mejorar el desempeño de la base de datos, cumplir con nuevos requerimientos de usuario, de la aplicación o políticas de seguridad; así como la compatibilidad con otras aplicaciones, la actualización de versiones, la estandarización de la tecnología de información en la organización, la reducción de costos que se puede tener al cambiar por software libre, el aumento en el volumen de datos, nuevos procesos de negocio, entre otros escenarios posibles. En el caso de este trabajo dicha migración se requiere llevar a cabo para lograr la reducción de costos y la independencia tecnológica.

Estrategia a seguir:

1- Es conveniente delimitar el alcance del proceso de migración.

2- Realizar un análisis de riesgos, así como analizar las condiciones actuales y finales. En ésta fase es necesario, igualmente, determinar la viabilidad técnica y la factibilidad económica de la solución planteada; pero antes, es oportuno realizar respaldos de la base de datos antes y después del proceso de migración, así como considerar en qué momento y durante cuánto tiempo se va a detener la operación de la base de datos en producción. Si esto no es posible, se debe determinar el procedimiento para identificar los datos que fueron ingresados durante el proceso de migración, para su actualización posterior.

3- Verificar que el gestor al cual se migra presente la característica de ser multiplataforma, es decir que funcione en casi todos los principales sistemas operativos: Linux, Unix, Solaris, BSDs, Mac OS, Beos, Windows, entre otros. Ello en caso de que aun no se haya logrado una independencia en cuanto a plataforma, es decir que no se tenga como sistema operativo alguna distribución (distro) de Linux.

4- Comprobar que el gestor al cual se va a migrar presenta los siguientes requisitos:

- ✓ Seguridad: Que se tenga bien definido e implementado los controles de acceso a los datos.
- ✓ Confiabilidad: Procesamiento correcto de las solicitudes de usuario, sin violar la consistencia de la base de datos. aún cuando el sistema sobre el que opera no es confiable, es decir que esté propenso a fallas.
- ✓ Escalabilidad: Posibilidad de aumentar la capacidad de atender usuarios o volumen de datos de manera lineal con la capacidad de cómputo necesaria
- ✓ Disponibilidad: Que se pueda hacer uso del mismo el mayor tiempo posible.

5- Realizar procesos de extracción, transformación y carga, los cuales incluyen el obtener los datos desde su origen, modificarlos para cumplir con la integridad, la consistencia y las reglas definidas para finalmente insertarlos en la base de datos destino. Una actividad central del proceso es realizar un análisis del modelo de datos actual y del nuevo, para determinar cuáles son las tablas y campos críticos de ambos; posteriormente, se analizará y documentará la correspondencia campo por campo del nuevo modelo con el modelo actual, especificando los valores por defecto, nulos, la tabla o tablas que serán el origen de datos de cada relación en el nuevo modelo y las dependencias funcionales de cada una de ellas. De ésta manera, es necesario considerar las diferencias de los tipos de datos entre el modelo actual y el nuevo, asegurar que la información pueda ser almacenada en los campos bajo la nueva definición, verificar el tamaño de los objetos y de la base de datos, revisar el tipo de índices que soporta la base de datos final y su manejo de transacciones.

6- Si se incluye la migración de procedimientos almacenados, cuando hay un cambio de manejador de bases de datos (RDBMS), se debe considerar que tal vez sea necesario programarlos nuevamente, ya que el código puede no ser compatible.

7- Probar de manera exhaustiva que las consultas realizadas por las aplicaciones puedan seguir ejecutándose normalmente. Esta actividad es una parte fundamental del proceso, debido a que los datos almacenados se vuelven importantes conforme pueden ser convertidos en información valiosa para los usuarios.

8- Existen otros aspectos que se deben tener en cuenta para el éxito de la migración, tales como:

- ✓ Contar con el apoyo de la alta dirección.
- ✓ Asignar a un equipo con la experiencia y competencias requeridas.
- ✓ Identificar las aplicaciones críticas que harán uso de los datos.
- ✓ Contar con los recursos e infraestructura necesarios bajo el dimensionamiento realizado.

9- Un factor crítico para el éxito de la migración de la base de datos es la realización de pruebas las cuales, inicialmente, pueden ser a pequeña escala para validar o modificar la arquitectura final y el plan de migración, así como para comprobar que las aplicaciones que harán uso de la base de datos funcionan correctamente y optimizar los tiempos y recursos necesarios. Es recomendable hacer pruebas generales para comprobar que el proceso completo funciona correctamente, medir los tiempos para tener una planeación integral y minimizar los riesgos.

10- Migrar los datos: ésta tarea por sí misma, puede ser considerada como un proyecto complejo que para ser exitoso requiere una planeación detallada, un profundo conocimiento tanto de los datos como de las herramientas necesarias para llevar a cabo el proceso, así como en forma importante, de los sistemas y

aplicaciones que hacen uso de los datos a partir del modelo final, para asegurar su correcto funcionamiento y continuidad en la operación.

11- Una vez terminado el proceso se deben medir los resultados y entregar un reporte global del trabajo realizado, mencionando cuántas tablas u otros objetos fueron migrados, cuántos registros se migraron exitosamente, cuántos no fueron migrados y cuál fue la causa de ello.

12- Se sugiere realizar un reporte ejecutivo que resuma y presente a los directivos los resultados obtenidos. Otro documento relevante para el cliente es la memoria técnica que contenga la configuración de los parámetros de la base de datos migrada, su estructura física y espacio disponible, entre otros datos relevantes.

- **Planificación de la migración de Datos**

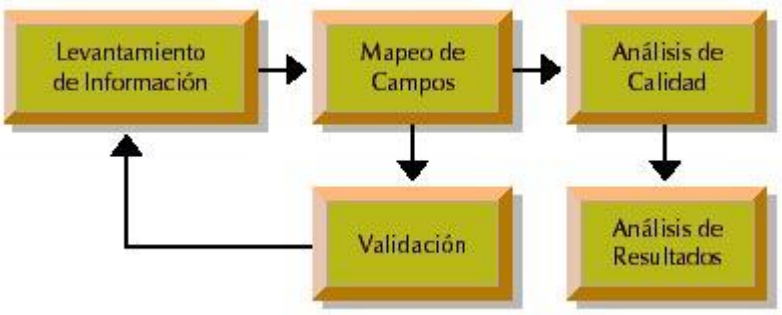

Figura 7 Migración de Datos

Actividades para planificar una migración de datos, a fin de reducir los riesgos durante el proceso de migración de Gestores de Base de Datos:

1- **Mapa de Conversión**.

- MODELO ENTIDAD/RELACIÓN DEL SISTEMA ACTUAL

- ✓ Levantamiento de información, basado en reuniones con el usuario involucrado en la migración.
- ✓ Diseño del modelo utilizando cualquier herramienta de diseño de Entidad/Relación.
- ✓ Definición de las tablas críticas a ser migradas.
- ✓ Definición de campos críticos.

- MODELO ENTIDAD/RELACIÓN DEL SISTEMA NUEVO

  - ✓ Construcción o desarrollo del modelo Entidad/Relación, a partir del modelo físico proporcionado por el proveedor.
  - ✓ Definición de tablas críticas a ser actualizadas.
  - ✓ Definición de campos críticos.

- MAPEO GENERAL DE TABLAS

  - ✓ Determinación de las equivalencias campo a campo.

- DISEÑO FÍSICO

  - ✓ Especificación de los valores por defecto, nulos y el origen de datos para cada una de las nuevas tablas del sistema nuevo.

2- Análisis de Calidad de Datos.

- ANÁLISIS DE INTEGRIDAD

  - ✓ Estudio de los datos existentes en el sistema actual, para la detección de los casos de valores inválidos presentes en la información a ser migrada.

- ANÁLISIS DE CONSISTENCIA

  ✓ Comparación de los valores presentes en cada campo, con respecto a los valores presentes en los campos que guardan relación con los mismos.

- ANÁLISIS DE NULIDAD

  ✓ Detección de valores nulos en los campos del sistema actual, no permitidos en el sistema destino.

- VALIDACIÓN DE LA INTEGRIDAD REFERENCIAL

  ✓ Validación de la existencia de registros pertenecientes a las tablas «Padre», en función de la existencia de sus correspondientes registros en las tablas «Hijos», según las relaciones específicas en el modelo Entidad/ Relación del sistema actual.

2.1.2.3 Consolidación

Caso de estudio: Migración de Oracle 10g hacia PostgreSQL 8.2

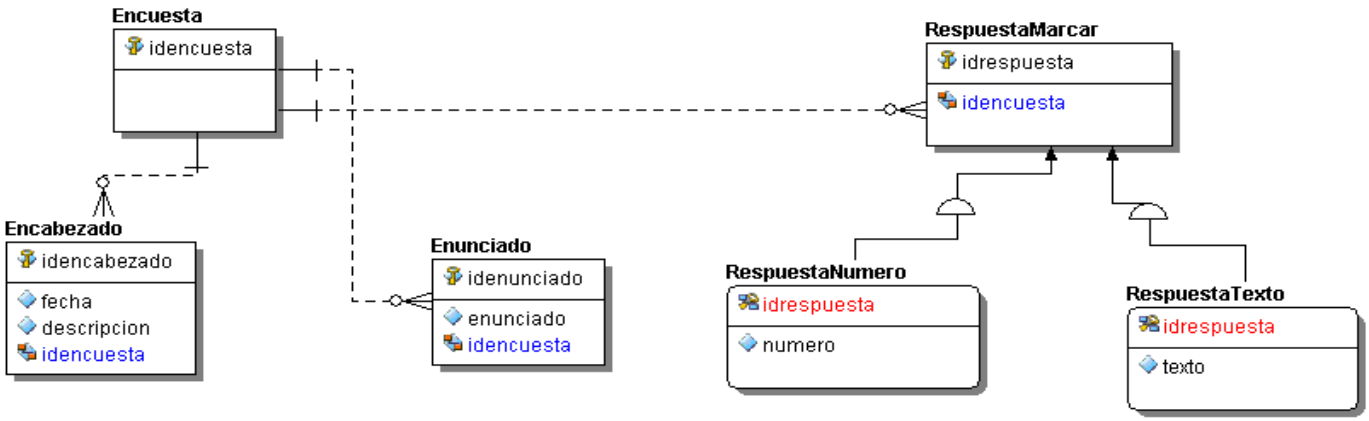

Figura 8 Diagrama de Base de Datos a migrar

- **Diferencias existentes entre Oracle y PostgreSQL**

1. PostgreSQL es una base de datos Objeto Relacional, mientras que Oracle es sólo una base de datos Relacional. Oracle no tiene la infraestructura OR y la simula con productos adicionales.

2. Oracle tiene consultas en paralelo, que PostgreSQL aún no tiene.

3. PostgreSQL tiene cinco lenguajes de programación procedurales, algunos sumamente especializados y avanzados como PL/perl y PL/R, mientras que Oracle sólo tiene uno.

4. PostgreSQL amplía el concepto de programación procedural para soportar funciones en cualquier parte (como procedimiento, como tabla, como operador, como selector, como filtro), mientras que Oracle sólo permite usarlas como procedimiento. Oracle simula parcialmente el comportamiento proveyendo paquetes, pero no tienen el alcance de PostgreSQL.

5. PostgreSQL soporta nativamente tipos de datos no escalares, con sus operadores, y ofrece la posibilidad de crear tus propios tipos de datos y operadores. Oracle requiere comprar software adicional y no puedes desarrollar tus propios tipos.

6. PostgreSQL Multi-Version Concurrency Control (MVCC, que sirve para lograr un control de concurrencia tan eficiente que en la gran mayoría de los casos no se requiere de bloqueos)

7. PostgreSQL tiene un sofisticado optimizador de consultas, que es capaz de resolver consultas complejas en tiempos comparables a los de los mejores DBMS (una última adición en este campo es la habilidad de usar varios índices para resolver consultas sobre una misma tabla);

8. Además PostgreSQL presenta herencia de tablas (aunque la orientación a objetos no está completa ésta característica se la puede utilizar para lograr el particionamiento de tablas en varios discos mediante una técnica llamada "restricciones excluyentes").

9. El sistema de reglas en PostgreSQL (el sistema re-escritor de consultas), permite identificar ciertas acciones sobre una tabla y reemplazarlas por otras o ejecutar adicionales (podemos, por

ejemplo, crear una regla para que al hacer INSERT sobre una vista se reemplace por un INSERT sobre la tabla en la que se basa la vista, dando así la impresión de vistas actualizables);

10. PostgreSQL presenta módulos contribuidos entre los que se cuentan OpenFTS (para indexación de texto completo) y PostGIS (que añade objetos geográficos lo cual permite a PostgreSQL ser usado en proyectos de bases de datos para sistemas de información geográfica, por ejemplo los que usan sistemas repartidores), también módulos de criptografía (SHA1, MD5).

11. En PostgreSQL los índices compuestos, únicos, parciales, funcionales (sobre funciones) pueden ser definidos como B-tree, R-tree, hash o GiST, y toda la infraestructura necesaria para extender estos tipos de índices.

12. En PostgreSQL la replicación es asincrónica.

13. PostgreSQL tiene transacciones anidadas (savepoints).

14. PostgreSQL cuenta con respaldos en línea (respaldos incrementales).

15. PostgreSQL presenta además escritura adelantada de registros (WAL) para evitar pérdidas de datos en caso de falla de energía, fallos del Sistema Operativo, y fallas de hardware.

16. PostgreSQL presenta juegos de caracteres internacionales, codificación de caracteres multibyte y Unicode.

17. PostgreSQL incluye características para la integridad de los datos, tales como: claves primarias, llaves foráneas con capacidad de actualizar en cascada o restringir la acción en caso que existan hijos, restricción check, restricción de unicidad y restricción not null; todas las restricciones se pueden postergar hasta el momento de terminar la transacción.

18. PostgreSQL tiene interfaces de programación para Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, y Qt sólo para nombrar algunas.

19. Oracle tiene muchísimos más parámetros de configuración que PostgreSQL, muchos de ellos son secretos, no están documentados, sólo son conocidos por el personal de soporte de Oracle que los utiliza para ayudar solamente si se tiene el contrato de mantenimiento al día.

- **Migración de la Base de Datos**

Migrar una Base de Datos Oracle a PostgreSQL

Estructura

La migración del esquema de una base de datos Oracle, a una base de datos PostgreSQL es algo complicado en función de las particularidades de Oracle que se incluya. Existen distintas herramientas para automatizar la migración de esquemas de Oracle hacia PostgreSQL, pero ninguna de ella es totalmente infalible, y pueden ser necesarios ajustes manuales.
Ejecutar un script de creación de bases de datos en PostgreSQL:

```
-postgres$ createdb nombre_bd
-postgres$ psql nombre_bd
Welcome to psql, the PostgreSQL interactive terminal.
Type: \copyright for distribution terms
\h for help with SQL commands
\? for help on internal slash commands
\g or terminate with semicolon to execute query
\q to quit
nombre_bd=#\i /ruta_hacia_el_script/script_bd.sql
```

IMPORTANTE: Las base de datos en PostgreSQL debe crearse en minúsculas. De lo contrario fallarán muchos de los comandos de psql (entre otros DROP DATABASE).

Si va a exportarse a una base de datos PostgreSQL a través de **Conectividad Abierta de Bases de Datos** (ODBC), es conveniente instalar el catálogo de extensiones ODBC, que realiza un ajuste más estricto de las funciones de PostgreSQL hacia el estándar ODBC. Para instalarlo, basta ejecutar el comando:

```
-postgres$ psql -d nombre_bd -f /usr/share/pgsql/odbc.sql
```

CREATE
CREATE
...
CREATE

En el archivo odbc.sql, se indica a PostgreSQL los ajustes necesarios. La ruta hacia el mismo puede variar en función de la instalación, versión de PostgreSQL, de la distribución Linux, etc.

Tablas

- **A continuación se muestran las tablas creadas en Oracle:**

```
/*
Creando tabla ENCUESTA
============================
*/
create table ENCUESTA
(
  IDENCUESTA NUMBER (18) not null
);
alter table ENCUESTA
  add constraint PK_ENCUESTA primary key (IDENCUESTA);
/*
Creando tabla ENCABEZADO
============================
*/
```

```
create table ENCABEZADO
(
  IDENCABEZADO NUMBER (18) not null,
  FECHA        DATE not null,
  DESCRIPCION NVARCHAR2 (100) not null,
  IDENCUESTA   NUMBER (18) not null
);
alter table ENCABEZADO
  add constraint PK_ENCABEZADO primary key (IDENCABEZADO);
alter table ENCABEZADO
  add constraint FK_ENCUESTAENCABEZADO foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
/*
Creando tabla ENUNCIADO
============================
*/
create table ENUNCIADO
(
  IDENUNCIADO NUMBER (18) not null,
  ENUNCIADO   VARCHAR2 (100) not null,
  IDENCUESTA NUMBER (18) not null
```

```
);
alter table ENUNCIADO
  add constraint PK_ENUNCIADO primary key (IDENUNCIADO);
alter table ENUNCIADO
  add constraint FK_ENCUESTAENUNCIADO foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
/*
Creando tabla PRUEBA
============================
*/
create table PRUEBA
(
  ID  NUMBER (18),
  DESCRIPCION NVARCHAR2 (100)
);
/*
Creando tabla RESPUESTAMARCAR
============================
*/
create table RESPUESTAMARCAR
(
```

```
  IDRESPUESTA NUMBER (18) not null,
  IDENCUESTA NUMBER (18) not null
);
alter table RESPUESTAMARCAR
  add constraint PK_RESPUESTAMARCAR primary key (IDRESPUESTA);
alter table RESPUESTAMARCAR
  add constraint FK_ENCUESTARESPUESTAMARCAR foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
/*
Creando tabla RESPUESTANUMERO
============================
*/
create table RESPUESTANUMERO
(
  IDRESPUESTA NUMBER (18) not null,
  NUMERO NUMBER (18)
);
alter table RESPUESTANUMERO
  add constraint PK_RESPUESTANUMERO primary key (IDRESPUESTA);
alter table RESPUESTANUMERO
  add constraint FK_RESPUESTAMRESPUESTANUMERO foreign key (IDRESPUESTA)
```

```
  references RESPUESTAMARCAR (IDRESPUESTA);
/*
Creando tabla RESPUESTATEXTO
============================
*/
create table RESPUESTATEXTO
(
  IDRESPUESTA NUMBER (18) not null,
  TEXTO  NVARCHAR2 (1000) not null
);
alter table RESPUESTATEXTO
  add constraint PK_RESPUESTATEXTO primary key (IDRESPUESTA);
alter table RESPUESTATEXTO
  add constraint FK_RESPUESTAMRESPUESTATEXTO foreign key (IDRESPUESTA)
  references RESPUESTAMARCAR (IDRESPUESTA);
```

- **Tablas migradas a PostgreSQL:**

```
/*
Creando tabla ENCUESTA
============================
*/
create table ENCUESTA
```

```
(
  IDENCUESTA NUMERIC (18, 0) not null
);
alter table ENCUESTA
  add constraint PK_ENCUESTA primary key (IDENCUESTA);
/*
Creando tabla ENCABEZADO
============================
*/
create table ENCABEZADO
 (
  IDENCABEZADO NUMERIC (18, 0) not null,
  FECHA DATE not null,
  DESCRIPCION VARCHAR (100) not null,
  IDENCUESTA   NUMERIC (18) not null
);
alter table ENCABEZADO
  add constraint PK_ENCABEZADO primary key (IDENCABEZADO);
alter table ENCABEZADO
  add constraint FK_ENCUESTAENCABEZADO foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
```

```
/*
Creando tabla ENUNCIADO
============================
*/
create table ENUNCIADO
(
  IDENUNCIADO NUMERIC (18) not null,
  ENUNCIADO VARCHAR (100) not null,
  IDENCUESTA NUMERIC (18) not null
);
alter table ENUNCIADO
  add constraint PK_ENUNCIADO primary key (IDENUNCIADO);
alter table ENUNCIADO
  add constraint FK_ENCUESTAENUNCIADO foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
/*
Creando tabla PRUEBA
============================
*/
  create table PRUEBA
(
```

```
  ID NUMERIC (18),
  DESCRIPCION VARCHAR (100)
);
/*
Creando tabla RESPUESTAMARCAR
============================
*/
create table RESPUESTAMARCAR
(
  IDRESPUESTA NUMERIC (18) not null,
  IDENCUESTA NUMERIC (18) not null
);
alter table RESPUESTAMARCAR
  add constraint PK_RESPUESTAMARCAR primary key (IDRESPUESTA);
alter table RESPUESTAMARCAR
  add constraint FK_ENCUESTARESPUESTAMARCAR foreign key (IDENCUESTA)
  references ENCUESTA (IDENCUESTA);
/*
Creando tabla RESPUESTANUMERO
============================
*/
```

```
create table RESPUESTANUMERO
(
  IDRESPUESTA NUMERIC (18) not null,
  NUMERO NUMERIC (18)
);
alter table RESPUESTANUMERO
  add constraint PK_RESPUESTANUMERO primary key (IDRESPUESTA);
alter table RESPUESTANUMERO
  add constraint FK_RESPUESTAMRESPUESTANUMERO foreign key (IDRESPUESTA)
  references RESPUESTAMARCAR (IDRESPUESTA);
/*
Creando tabla RESPUESTATEXTO
============================
*/
create table RESPUESTATEXTO
(
  IDRESPUESTA NUMERIC (18) not null,
  TEXTO VARCHAR (1000) not null
);
alter table RESPUESTATEXTO
  add constraint PK_RESPUESTATEXTO primary key (IDRESPUESTA);
```

```
alter table RESPUESTATEXTO
  add constraint FK_RESPUESTAMRESPUESTATEXTO foreign key (IDRESPUESTA)
  references RESPUESTAMARCAR (IDRESPUESTA);
```

Al migrar las tablas se tuvo que realizar algunos cambios ya que existen algunas diferencias, aunque muy pequeñas en cuanto a creación de tablas en ambos gestores, específicamente en el tratamiento de los tipos de datos.

Los datos

Tras llevar a cabo la migración de la estructura de la BD, el siguiente paso son los datos. Este proceso puede afrontarse desde muchos puntos de vista, en función de la imaginación del encargado, de la magnitud de de los datos, y de las posibilidades de los Sistemas Gestores de Base de Datos (SGBD) origen y destino. Dado que nos estamos centrando en Oracle y PostgreSQL, destacaremos las siguientes:

- ✓ Exportar los datos a un fichero de SQL: La sintaxis de dicho fichero, será más o menos estándar según las variaciones de la BD origen (ésta es la manera más habitual de exportar datos del SGBD Oracle). Para realizar la importación, pueden ser necesarios ajustes y modificaciones en el fichero para adaptarlo a la sintaxis del SGBD destino.
- ✓ Exportando los datos a un formato de texto plano. Este método es muy efectivo, y puede atacarse desde muchos frentes. Por ejemplo, se puede llevar a cabo conectándose a través de **Conectividad Abierta de Bases de Datos** (ODBC) a la base de datos Oracle con un SGBD que facilite la exportación (por ejemplo MS-Acces). Se importan los datos de las tablas al SGBD intermedio (MS-Acces), y se exportan de nuevo los mismos como texto plano con separadores (por ejemplo tabulaciones y retornos de carro). Tras ello, se pueden utilizar las herramientas de importación del SGBD destino (PostgreSQL y pg_dump) para llevar a cabo la importación. Este mismo esquema, puede enfocarse utilizando combinaciones de lenguajes de filtrado para construir el fichero destino (por

ejemplo PERL o AWK). Como última opción, existe software comercial que lleva a cabo la migración, como el producto Chyfo de Ispirer Systems.

En el caso que se tomó como prueba, el fichero de datos era texto plano con órdenes INSERT de SQL para cada una de las tablas con un tamaño de unos 26Mb. Muchas de esas órdenes INSERT, comunes del SQL, se veían afectadas por particularidades propias del Oracle. Por ejemplo, cuando la tabla sobre la que se estaba llevando a cabo la inserción tenía campos de tipo fecha, Oracle había exportado los datos preparando un INSERT como el siguiente:

INSERT INTO ENCABEZADO (IDENCABEZADO, FECHA, DESCRIPCION, IDENCUESTA)
VALUES (104, TO_Date ('09/20/1999 12:00:00 AM', 'MM, DD, YYYY HH: MI: SS AM'), '327', 1);

En PostgreSQL, la función equivalente a TO_Date () es to_date (), en este caso, lo único que habría que hacer es reemplazar todas las ocurrencias de TO_Date en el fichero, por to_date. Dado el considerable tamaño del fichero, no se podía abrir con un editor "al uso" y efectuar una Búsqueda/Sustitución. La solución, era el uso de comandos y utilidades UNIX de expresiones regulares. En este caso, se utiliza la orden sed.

```
-postgres$ cat /descargas/scripts-bd/gsa_datos.sql \
|sed's/TO_Date/to date/g' > \
/descargas/scripts-bd/gsa_datos_pg.sql
```

Al ejecutar un archivo SQL de este tipo, pueden ocurrir errores, debidos a dependencias entre claves ajenas (integridad referencial). Hay varias maneras de solucionar el problema:

- ✓ Reordenar las inserciones. Si conseguimos establecer un orden de inserción entre las distintas tablas, podemos evitar los problemas de integridad referencial.
- ✓ Crear las tablas SIN claves primarias y claves ajenas, después realizar la inserción y tras ello, establecer las claves primarias y las claves ajenas de la base de datos. El problema de éste sistema, es que si se produjo un error en la exportación de los datos o en su posterior manipulación, pueden aparecer inconsistencias en la BD y no ser detectadas.

- ✓ Ejecutar el script de inserción varias veces. A medida que se realizan las distintas pasadas, se van produciendo más inserciones en la BD, hasta que finalmente, todos los datos resultan insertados. Dependiendo de la complejidad de la base de datos puede variar el número de ejecuciones necesarias.

Secuencias y Vistas

En muchas bases de datos relacionales, se hace uso de las secuencias, principalmente, en claves primarias e índices. Hay que ser especialmente cuidadoso al migrar las secuencias entre dos bases de datos, ya que, si en la base de datos destino se crea la secuencia pero no se inicializa con el último valor que tenía en la base de datos origen, podemos encontrar muchos problema (por ejemplo, el no poder realizar inserciones porque aparezcan errores de claves primarias duplicadas). Hay que destacar que la sintaxis utilizada para la creación de secuencias tanto en Oracle como en PostgreSQL es bastante común, la única diferencia es en el tamaño de la cadena a generarse, ya que un valor máximo de una secuencia generada en Oralce puede contener hasta 27 dígitos y en PostgreSQL solo se cuenta con 17 dígitos para ello, esto se demuestra a continuación:

- **Secuencia hecha en Oracle**:

```
/*
Creando Secuencia SEQ_PRUEBA
============================
*/
create sequence SEQ_PRUEBA
minvalue 1
maxvalue 999999999999999999999999999
start with 41
increment by 1
```

cache 20;

- **Secuencia migrada a PostgreSQL**

```
/*

Creando Secuencia SEQ_PRUEBA

============================

*/

create sequence SEQ_PRUEBATONY
minvalue 1
maxvalue 99999999999999999
start with 41
increment by 1
cache 20;
```

En cuanto a las vistas, PostgreSQL proporciona un sistema de vistas compatible sintácticamente con el de otros SGBD, pero que internamente trabaja de una forma poco habitual, pues hace uso de un sistema de reglas.

Las vistas en PostgreSQL se implementan utilizando el sistema de reglas. Esto tiene algunos efectos colaterales. Uno de ellos es que la información sobre una vista en el sistema de catálogos de PostgreSQL es exactamente el mismo que para una tabla. De este modo, para los traductores de consultas (queries), no hay diferencia entre una tabla y una vista, son lo mismo: relaciones. Los beneficios de implementar las vistas con el sistema de reglas están en que el optimizador tiene toda la información sobre qué tablas tienen que ser revisadas, más las relaciones entre estas tablas, más las cualidades restrictivas a partir de la definición de las vistas, más las cualidades de la consulta (query) original, todo en un único árbol de traducción. Y ésta es también la situación cuando la query original es ya una join entre vistas. Ahora el optimizador debe decidir cuál es la mejor ruta para ejecutar la consulta (query). Cuanta más información

tenga el optimizador, mejor será la decisión. Y la forma en que se implementa el sistema de reglas en PostgreSQL asegura que toda la información sobre la query este utilizable.

- **Vistas creadas en Oracle:**

```
/*
Creando vista CANT_RESPUESTA_ENCUESTA
====================================
*/
create or replace view cant_respuesta_encuesta as
select e.idencuesta, count (r.idrespuesta) as cantidad from encuesta e inner join respuestamarcar
r on e.idencuesta = r.idencuesta
group by e.idencuesta
/
/*
Creando vista VW_ENCABEZADO
====================================
*/
create or replace view vw_encabezado as
select "IDENCABEZADO","FECHA","DESCRIPCION","IDENCUESTA" from encabezado t
order by t.idencabezado desc
/
/*
Creando vista VW_ENUNCIADO
====================================
*/
create or replace view vw_enunciado as
select "IDENUNCIADO","ENUNCIADO","IDENCUESTA" from enunciado
/
```

- **Vistas migradas a PostgreSQL:**

```
/*
Creando vista CANT_RESPUESTA_ENCUESTA
====================================
*/
create or replace view cant_respuesta_encuesta as
select e.idencuesta, count (r.idrespuesta) as cantidad from encuesta e inner join respuestamarcar
r on e.idencuesta = r.idencuesta
group by e.idencuesta;
/*
Creando vista VW_ENCABEZADO
====================================
*/
create or replace view vw_encabezado as
select t.IDENCABEZADO, t.FECHA, t.DESCRIPCION, t.IDENCUESTA from encabezado t
order by t.idencabezado desc;
/*
Creando vista VW_ENUNCIADO
====================================
*/
create or replace view vw_enunciado as
select IDENUNCIADO, ENUNCIADO, IDENCUESTA from enunciado;
```

La creación de vistas, por tanto, no presenta a priori grandes problemas de portabilidad, los únicos problemas aparecen en los CAST de las cláusulas UNION, donde no podemos aplicar la misma sintaxis que Oracle. En PostgreSQL pueden resolverse los castings haciendo uso del comando estándar SQL:

CAST (nombre_atributo AS tipo_atributo)

o bien de la notación nativa de PostgreSQL:

(nombre_atributo: tipo_atributo)

Roles

El concepto de ROL que mantienen hoy en día algunas bases de datos propietarias, no está implementado en PostgreSQL. Sin embargo, PostgreSQL plantea un sistema bastante similar, los grupos. Los grupos de PostgreSQL, son distintos a los definidos dentro del sistema operativo sobre el cual está instalado el software. Cualquier conexión a PostgreSQL debe ser realizada con un usuario específico, y cualquier usuario puede pertenecer a uno o más grupos definidos. La tabla de usuarios controla los permisos de acceso y quién está autorizado a realizar acciones en el sistema (y qué acciones puede realizar). Los grupos existen como un mecanismo para simplificar la ubicación de estos permisos. Tanto las tablas de usuarios como de grupos existen como objetos globales de base de datos, lo que significa que no están adscritas a ninguna base de datos en particular. La principal diferencia entre los grupos de PostgreSQL y los roles Oracle, es que los roles pueden anidarse entre sí y los grupos no. Es decir, un ROL puede incluir o actuar como otros ROLES acumulando permisos, pero un grupo de PostgreSQL no puede pertenecer a otro grupo. El sistema de PostgreSQL, permite conseguir los mismos resultados que el de Oracle, aunque de una manera menos flexible.

Lenguajes Procedurales PL/SQL vs PL/pgSQL

PostgreSQL, facilita la programación con distintos lenguajes procedurales. Entre todos ellos, cabe citar los siguientes, PL/pgSQL, PL/Tcl, PL/Perl, PL/Python. La instalación del soporte para cada uno de estos lenguajes se lleva a cabo por parte de un usuario administrador y de manera muy sencilla:

-postgres$ createlang --dbname=nombre_bd -.pglib=/ruta_hacia_lib_lengPL.so/ lengPL

Si queremos habilitar el soporte para el lenguaje en todas las bases de datos, basta con habilitarlo en la base de datos template1, de la que el resto heredan las propiedades. Por ejemplo, para habilitar el soporte de PL/pgSQL en todas las bases de datos de un sistema, se ejecutaría:

-postgres$ createlang --dbname=gsa_pg --pglib=/usr/lib/pgsql/ plpgsql

El lenguaje PL/pgSQL es el claro competidor del PL/SQL de Oracle. Aunque PL/pgSQL no alcanza todavía la potencia y versatilidad de PL/SQL, es un lenguaje de características muy similares, que evoluciona constantemente de forma pareja al SGBD PostgreSQL. A continuación, se ofrece una lista con las principales diferencias entre ambos lenguajes:

- ✓ PL/pgSQL no soporta paquetes (packages). Es necesario crear funciones de forma individual.
- ✓ PL/pgSQL no tiene procedimientos, solo tiene funciones. Pero, desde una función PL/pgSQL si se puede modificar la Base de Datos. Es decir, tanto las funciones como los procedimientos Oracle, pasan a ser Funciones en PostgreSQL.
- ✓ PL/pgSQL no soporta parámetros variables, ni parámetros por defecto. En PL/pgSQL, el número de argumentos de las funciones, es fijo y definido. Por contra, para solventar esta desventaja, PostgreSQL permite la sobrecarga de funciones.
- ✓ En PL/pgSQL es necesario escapar la comilla simple (') para ello, se duplican [10]('). Existen más diferencias entre ambos lenguajes (el rowid de Oracle,...). Pero si los procedimientos/funciones de Oracle no exprimen sus características más extrañas, son fácilmente portables a PL/pgSQL.
- ✓ PostgreSQL ni tiene parámetros nominados. Usted tiene que especificarlos como ALIAS dentro de su función.

[10] No es una comilla doble, sino dos sencillas

- ✓ Oracle puede tener parámetros IN, OUT, e INOUT pasados a funciones. El INOUT, por ejemplo, significa que el parámetro recibirá un valor y retornará otro. PostgreSQL sólo tiene parámetros "IN" y las funciones sólo pueden retornar un valor simple.
- ✓ El comando /show errors no existe en PostgreSQL.

- **A continuación se muestran los procedimientos creados en Oracle**:

```
/*
Creando procedimiento OBTENERRESPUESTATEXTO
=========================================
*/
create or replace procedure ObtenerRespuestaTexto
(
  prmIdRes number,
  prmIdEncuesta out number
) is
begin
  select r.idencuesta into prmIdEncuesta from respuestamarcar r where r.idrespuesta = prmIdRes;
end;
/
/*
Creando procedimiento SALVARENCABEZADO
==================================
*/
create or replace procedure SALVARENCABEZADO
(
  prmIdEncabezado encabezado.idencabezado%type,
  prmfecha encabezado.fecha%type,
```

```
  prmDescripcion encabezado.descripcion%type,
  prmIdEncuesta encabezado.idencuesta%type
) is
begin
   insert into encabezado (idencabezado, fecha, descripcion, idencuesta)
   values (prmIdEncabezado, prmfecha, prmDescripcion, prmIdEncuesta);
end SALVARENCABEZADO;
/
```

- **Procedimientos migrados hacia PostgreSQL**:

```
/*
Creando procedimiento OBTENERRESPUESTATEXTO
=======================================
*/

create or replace FUNCTION ObtenerRespuestaTexto (prmIdRes numeric) RETURNS numeric as'
declare
prmIdEncuesta numeric;
begin
select r.idencuesta into prmIdEncuesta from respuestamarcar r where r.idrespuesta = prmIdRes;
return prmIdEncuesta;
end;
'LANGUAGE 'plpgsql';
/*
 Creando procedimiento SALVARENCABEZADO
==================================
*/
```

```
CREATE OR REPLACE FUNCTION "public"."_a_insertarencabezado" (pidencabezado
numeric, pfecha date, pdescripcion varchar, pidcuenta numeric) RETURNS
"pg_catalog"."void" AS
$body$
begin
 insert into encabezado (idencabezado, fecha, descripcion, idencuesta)
   values ($1, $2, $3, $4);
   return;
end;

$body$
LANGUAGE 'plpgsql' VOLATILE CALLED ON NULL INPUT SECURITY INVOKER;
```

Triggers

En **PL / SQL** se da la posibilidad de escribir disparadores (triggers) de reacción en los eventos: BEFORE/ AFTER DELETE/ UPDATE/ INSERT. Además los factores desencadenantes (triggers) pueden ser utilizados para la actualización de datos a través de puntos de vista.
En **PL/PgSQL**: Existen reglas declarativas que son ampliación de SQL. Se puede especificar SELECT, INSERT, DELETE o UPDATE como una regla de evento. Además las reglas pueden ser utilizadas para la actualización de datos a través vistas. Factores desencadenantes de procedimiento en PL / pgsql, PL / TCL, PL / Perl, C como son CREATE CONSTRAINT TRIGGER crea un disparador para apoyar una limitación. Se puede especificar BEFORE o AFTER o INSERT, DELETE o UPDATE como un disparador (trigger).

Trigger creado en Oracle

```
/*
Creando trigger TGR_INSERT_PRUEBA
=====================================
```

```
*/
create or replace trigger TGR_INSERT_PRUEBA
  after insert on encuesta
  for each row
declare
 -- local variables here
temp varchar2 (100);
begin
temp: = to_char (sysdate);
insert into prueba (id, descripcion)
values (seq_prueba.nextval, temp);
end TGR_INSERT_PRUEBA;
/
```

- **Trigger migrado hacia PostgreSQL**

```
CREATE OR REPLACE FUNCTION "public"."_a_trigger" () RETURNS trigger AS
$body$
declare
p varchar (30);
ids numeric;
BEGIN
  p = CURRENT_DATE;
  ids = nextval ('seq_prueba');
insert into prueba (id, descripcion)
values (ids, p);
RETURN NULL;
END;
$body$
LANGUAGE 'plpgsql' VOLATILE CALLED ON NULL INPUT SECURITY INVOKER;
```

```
CREATE TRIGGER "trigger_t" AFTER INSERT
ON "public"."encuesta" FOR EACH ROW
EXECUTE PROCEDURE "public"."_a_trigger"();
```

2.2 Proposición de una herramienta para la verificación del proceso de migración.

El propósito perseguido para el desarrollo de esta herramienta, una vez llevado a cabo el proceso de migración, no es más que la verificación de la migración tanto del esquema de la base de datos como de sus tablas fue satisfactoria. Ello se lleva a cabo tras una comparación de la base de datos fuente, en este caso comercial, con la base de datos migrada hacia Open-Source para tener seguridad de la integridad estructural de los objetos de base de datos migrados.

Rasgos que debe presentar esta herramienta.

- ✓ Comparar las tablas, índices, vistas, funciones, procedimientos almacenados, y disparadores (triggers) del gestor de base de datos origen (propietario) con el gestor destino (Open-Source).
- ✓ Chequear que los objetos existan tanto en el gestor origen (propietario) como en el gestor destino (Open-Source).
- ✓ Chequear el orden de los campos de la tabla.
- ✓ Chequear que los campos de la tabla existan y que puedan producir valores nulos.
- ✓ Verificar que los procedimientos almacenados y las funciones existan en la base de datos migrada hacia Open-Source.
- ✓ Verificar el tipo de dato, escala, precisión, longitud y valores predeterminados para los campos de las tablas, funciones y procedimientos almacenados.
- ✓ Chequear el orden de los argumentos para procedimientos almacenados y funciones.
- ✓ Realizar el chequeo sencillo de filas en todas las tablas de usuarios para verificar la integridad de los datos migrados.

- ✓ Generar un reporte donde se resuma los resultados de todo el proceso de verificación al nivel de esquema y datos. Dicho reporte debe proveer además información acerca de tipo de objeto, nombre del objeto e información de la tabla, esto para identificar los objetos que no fueron migrados satisfactoriamente.
- ✓ Usar Conectividad de la Base de Datos de Java (JDBC) para conectar simultáneamente la base de datos origen con la base de datos destino.
- ✓ Soportar varios chequeos y verificaciones.

Sistema de reporte de la herramienta de verificación de la migración:

Este sistema de reporte, como se ha visto, debe proveer de un reporte en el que se resuma todo el análisis hecho entre la base de datos a ser migrada y el destino.

- ✓ El mismo debe contener información que ayude a rastrear cualquier tipo de error ocurrido.
- ✓ Debe mostrar un sumario de los objetos migrados y aquellos que no fueron migrados satisfactoriamente. Incluyendo tipo de objeto y nombre de dicho objeto.
- ✓ Debe proveer además de información detallada acerca de los fallos.
- ✓ Debe proveer el número de filas en la base de datos a migrar y la base de datos destino, y si existen diferencias entre el número de filas entre dichas bases de datos.

Conclusiones del capítulo

En el capítulo se han abordado un grupo de aspectos esenciales en el proceso de desarrollo de la estrategia de migración hacia Sistemas Gestores de Base de Datos Libres.

Se dirige la atención hacia un caso de estudio donde se lleva a cabo la migración de una base de datos de Oracle 10g hacia PostgreSQL 8.2, haciendo un especial énfasis en la migración de los datos, ya que este es un factor muy importante durante este proceso de migración. Para maximizar su rendimiento y proporcionar una infraestructura informática viable se empleará el hardware y software base adecuado capaz de cubrir los altos niveles de procesamiento resultado de un uso intensivo de los datos.

Como consecuencia de la migración se plantean una serie de pasos a tener en cuenta para lograr una utilización óptima del gestor PostgreSQL se realizan una serie de ajustes en la configuración de la instalación del mismo. Todo ello para lograr vializar el trabajo del administrador de la base de datos (DBA) con un elevado nivel de calidad.

Introducción

En el presente capítulo se exponen los resultados alcanzados luego el proceso de migración de datos en el caso de estudio anteriormente analizado. El análisis de los resultados arrojados podrá demostrar si se hace viable y positivo o no este proceso. En algunos casos se analizaran algunas dificultades encontradas alrededor de la migración de los datos.

3.1 Resultados de la estrategia de migración.

1- En la determinación del alcance de la migración dado que sólo se analizó el proceso de migración para una pequeña base de datos, el alcance del mismo estuvo delimitado sólo al nivel del caso de prueba analizado.

2- Antes y después del proceso de migración se llevó a cabo un respaldo de la base de datos (backup), ello con el objetivo de mantener salvas de la información en caso de fallos. En cuanto a la viabilidad técnica vale destacar que se detuvo la operación de la base de datos a migrar por un tiempo de 1 minuto y 30 segundos algo realmente rápido, que en base de datos de mayor volumen de podría ser un tiempo relativamente corto en dependencia del volumen de datos que se manejará, esto se logró ya que se contó con una PC con un procesador Intel(R) Pentium(R) 4 con un una Unidad Central de Procesamiento (CPU) a 3.00 GHz y una memoria RAM de 2GB. En lo referente a la factibilidad económica, esta migración se hace 100 % factible ya que nadie puede levantar una demanda por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software, mientras que el precio de Oracle es de US$4.995 por procesador y la licencia Usuario Designado Plus (Named User Plus) a US$149 por usuario con un mínimo de cinco usuarios. Además de las ofertas de soporte, se tiene una importante comunidad de profesionales y entusiastas de PostgreSQL de los que se puede obtener beneficios y contribuir gratuitamente. PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste

mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.

3- En cuanto a nivel multiplataforma, PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), Linux, Solaris, BSDs, Mac OS, Beos, Windows.

4- En cuanto a lo siguientes requisitos:

Estabilidad y Confiabilidad: Es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Siendo por ello altamente confiable. Entre algunas de estas compañías vale destacar a Skype, DeutchBank, etc.

Seguridad: La seguridad de la base de datos esta implementada en varios niveles:

- ✓ Protección de los ficheros de la base de datos. Todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier usuario que no sea el superusuario de PostgreSQL.
- ✓ Las conexiones de los clientes al servidor de la base de datos están permitidas, por defecto, únicamente mediante sockets Unix locales y no mediante sockets TCP/IP. Ha de arrancarse el demonio con la opción -i para permitir la conexión de clientes no locales.
- ✓ Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario mediante el fichero pg_hba.conf situado en PG_DATA.
- ✓ Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- ✓ A cada usuario de PostgreSQL se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.
- ✓ Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.
- ✓ La autentificación es el proceso mediante el cual el servidor de la base de datos y el postmaster se aseguran de que el usuario que está solicitando acceso a la base de datos

es en realidad quien dice ser. Todos los usuarios que quieren utilizar PostgreSQL se comprueban en la tabla pg_user para asegurarse que están autorizados a hacerlo. Actualmente, la verificación de la identidad del usuario se realiza de distintas formas:

- Desde el shell del usuario

  Un demonio (deamon) que se lanza desde el shell del usuario anota el id original del usuario antes de realizar una inserción de id de usuario (setuid) al id del usuario postgres. El id original del usuario se emplea como base para todo tipo de comprobaciones.

- Desde la red

  Si Postgres se instala como distribuido, el acceso al puerto TCP del postmaster está disponible para todo el mundo. El Administrador de la Base de Datos (DBA) configura el fichero pg_hba.conf situado en el directorio PG_DATA especificando el sistema de autentificación a utilizar en base al equipo que realiza la conexión y la base de datos a la que se conecta. Por supuesto la autentificación basada en equipos no es perfecta incluso en los sistemas Unix. Es posible, para determinados intrusos, enmascarar el equipo de origen. Estos temas de seguridad están fuera del alcance de Postgres.

Confiabilidad: PostgreSQL tiene más de 15 años de desarrollo activo y se ha ganado la reputación de ser confiable (de hecho, hay compañías que aseguran haber tenido corriendo postgres en producción durante varios años y con altas tasas de actividad sin haber experimentado problemas de ningún tipo) y mantener la integridad de los datos.

Escalabilidad: PostGreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000 peticiones por día.

Disponibilidad: Alta disponibilidad.

5- Durante el proceso de migración se obtuvo los datos de la base de datos origen, modificándolos luego y así cumplir con la integridad referencial, la consistencia y las reglas definidas, insertándolos posteriormente en la base de datos destino. Se realizó un análisis del modelo de datos actual y del nuevo, para determinar cuáles habían sido las tablas y campos críticos de ambos; posteriormente, se analizó la correspondencia campo por campo del nuevo modelo con el modelo actual, especificando los valores por defecto, nulos, la tabla o tablas que serán el origen de datos de cada relación en el nuevo modelo y las dependencias funcionales de cada una de ellas.

6- Se llevó a cabo el analizar la verificación de la integridad referencial entre las tablas de acuerdo con los requerimientos del modelo en el nuevo ambiente y determinar las limitaciones existentes, considerando las diferencias de los tipos de datos entre el modelo actual y el nuevo, para asegurar que la información se almacenara en los campos bajo la nueva definición, verificar el tamaño de los objetos y de la base de datos, revisar el tipo de índices que soporta la base de datos final y su manejo de transacciones. En este paso se tuvo que realizar un trabajo profundo debido a que Oracle 10g y PostgreSQL 8.2 comparten gran cantidad de tipos de datos pero no en su totalidad puesto que existen diferencias entre muchos tipos de datos existentes en ambos gestores. En cuanto al tipo de índices, debido a que PostgreSQL cuenta con una gran cantidad de índices existentes en Oracle, hubo muchas dificultades en cuanto a migración de índices. Hay que destacar que PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos en la base de datos. En lo referente a las transacciones de PostgreSQL, estas permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.

```
BEGIN WORK;
.....
Sentencias SQL;
.....
COMMIT WORK;
```

7- En cuanto a la migración de procedimientos almacenados, debido al cambio de manejador de bases de datos (RDBMS), fue necesario programarlos nuevamente, ya que en PostgreSQL no se maneja el término de procedimientos sino de funciones.

8- Luego se realizaron pruebas, para comprobar que las consultas realizadas por las aplicaciones se pudieran seguir ejecutándose y los resultados fueron satisfactorios, no se encontraron dificultades. Esta actividad es una parte fundamental del proceso, debido a la importancia que representan los datos almacenados.
9- PostgreSQL cuenta con soporte eficiente llevado a cabo de la siguiente forma:

Soporte de la comunidad. El principal soporte de PostgreSQL lo provee la comunidad a través de las listas de correo y los canales de las Charlas Interactivas Internet (IRC).

Soporte comercial. También existen varias empresas que proveen soporte comercial para postgres, entre las que tenemos: Fujitsu (Australia); Hub.org (Canadá); PostgreSQL, Inc. (Canadá); credativ GmbH (Alemania); Afilias Limited (Irlanda); Investigadores de Software Asociados[Software Research Associates] (Japón, con subsidiarias en USA); Delta-Soft LLC (Rusia); 2ndQuadrant (Reino Unido); 800 Pound Gorilla (USA); Command Prompt, Inc. (USA); EnterpriseDB (USA); Greenplum (USA); Pervasive Software, Inc. (USA); RedHat, Inc. (USA) y, hace unos días atrás, Sun Microsystems anunció que daría soporte a sus usuarios de Solaris.

10- Luego se pasó a la tarea de realización de pruebas las cuales, inicialmente, fueron a pequeña escala para validar o modificar la arquitectura final y el plan de migración, así como para comprobar que las aplicaciones que hacían uso de la base de datos siguieran funcionando correctamente y optimizar los tiempos de respuesta y los recursos necesarios para esto. Se realizaron pruebas generales para comprobar que el proceso completo funciona correctamente, midiéndose los tiempos para tener una planeación integral y minimizar los riesgos.

11- Migrar los de datos fue una tarea compleja y de gran envergadura, para lograr su éxito se requirió de una planeación e investigación detalladas para lograr un profundo conocimiento tanto de los datos como de las herramientas necesarias para llevar a cabo el proceso, así como en forma importante, de los sistemas y aplicaciones que hacían uso de los datos a partir del modelo final, para asegurar su correcto funcionamiento y continuidad en la operación.

Luego de analizar los datos arrojados tras el proceso de migración se puede arribar a la valoración de que la migración efectuada de Oracle 10g hacia PostgreSQL 8.2 fue aunque no en su totalidad, un proceso viable, debido a que los resultados arrojados durante el proceso fueron positivos.

3.2 Proposición de funcionalidades para mejorar PostgreSQL 8.2.

Como es sabido en todas las comunidades de desarrollo de aplicaciones de base de dato, Oracle es un gestor que presenta muchas ventajas por encima de muchos otros gestores, entre ellos fundamentalmente PostgreSQL 8.2, el cual se ha analizado en el trascurso de este capítulo. Es por ello que se hace necesario puntualizar en este trabajo la proposición de algunas mejoras que se pueden realizar a dicho gestor, PostgreSQL 8.2. La necesidad de ello está dada debido a que si como objetivo está el plantear una migración de un gestor de base de datos tan potente como Oracle hacia PostgreSQL, se hace necesario entonces encaminar el desarrollo de algunas funcionalidades implementadas en Oracle 10g que lo hacen tan potente, en PostgreSQL 8.2, que si bien tal vez no sea con el mismo nivel de complejidad si se asemeje a sus mecanismos de funcionamiento, ello con motivo de lograr un elevado grado de robustez en PostgreSQL 8.2 para al desarrollo óptimo de aplicaciones a través del mismo. Entre algunas de las herramientas que se proponen implementar en PostgreSQL 8.2 se encuentran:

1-) Real Apliccation Clusters (RAC) la única tecnología de Bases de Datos en el mercado, es un software que permite utilizar un clúster de servidores ejecutando múltiples instancias sobre una misma base de datos. Los archivos de base de datos quedan almacenados en discos física o lógicamente conectados a cada nodo, de modo tal que todas las instancias activas pueden leerlos o escribirlos. RAC maneja el acceso a los datos, de modo tal que los cambios en los datos son coordinados entre las instancias y cada instancia ve imágenes consistentes de la base. El interconnect del clúster permite que las instancias se pasen entre ellas información de coordinación e imágenes de los datos.

Real Application Clusters es un ambiente de cómputo que aprovecha el poder de procesamiento de múltiples computadoras interconectadas (clúster) para la ejecución de aplicaciones, sin necesidad de cambios en las mismas.

Este producto brindará la opción de configurar arquitecturas de sistemas de Clusters SMP (symmetric multi-processors) para correr las aplicaciones de PostgreSQL 8.2. En ésta arquitectura, en cada uno de los nodos se ejecutan concurrentemente transacciones contra la misma base de datos, RAC coordinaría el acceso a los datos compartidos, para proveer consistencia e integridad.

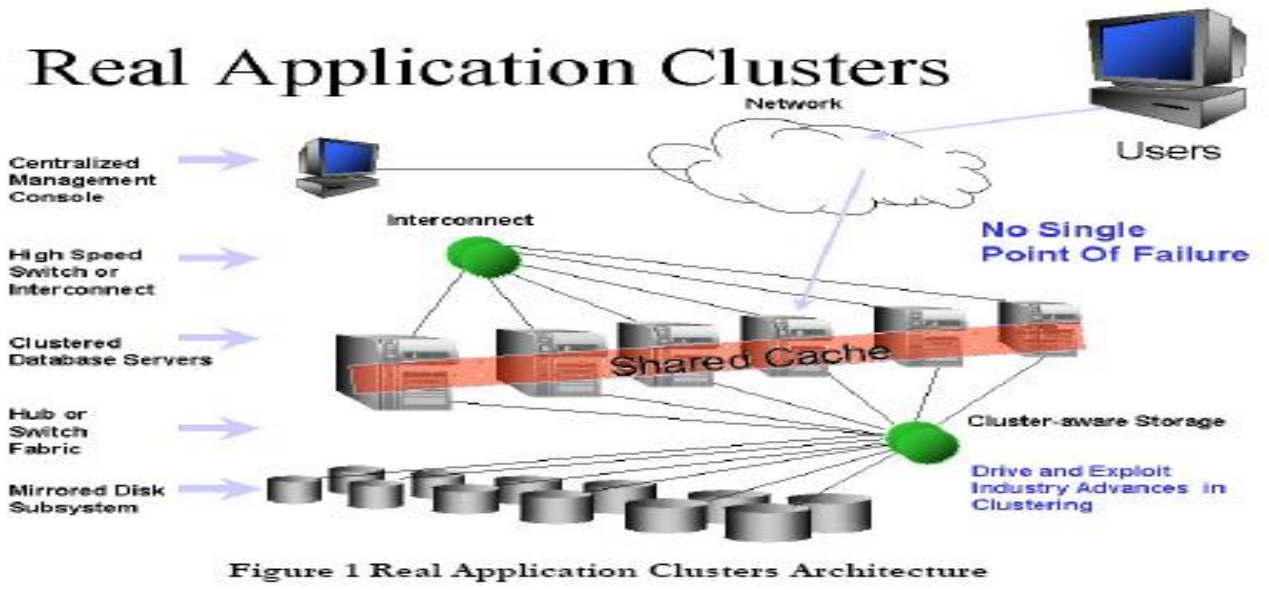

Figure 1 Real Application Clusters Architecture

Figura 9 Arquitectura de Real Apliccation Clusters (RAC).

Cuando estas arquitecturas paralelas son implementadas correctamente, se obtienen mejoras dramáticas en los tiempos de respuesta y ejecución, en las posibilidades de escalabilidad y crecimiento, y en la disponibilidad de las Bases de Datos y aplicaciones. Sin embargo, el alto grado de sofisticación de estas tecnologías requiere de un especial cuidado de parte de los administradores del sistema y de las bases de datos.

Se implementaría un servicio con el objetivo de permitir que los clientes, mediante una configuración adecuada, obtengan un uso eficiente y seguro de sus aplicaciones sobre una plataforma de RAC en el menor tiempo posible, con una relación costos-beneficios tangibles.

Con RAC, una arquitectura de caché compartida que ofrece todas las ventajas del Grid Computing, se puede obtener:

- ✓ **Actividad sin interrupciones:** Supera en rendimiento incluso a los sistemas mainframe más rápidos, proporcionándole una alta disponibilidad y una velocidad de máximo nivel. Pone a su disposición las tecnologías necesarias y una arquitectura muy eficaz de alta disponibilidad, que le permitirá mantener protegidos los datos y los sistemas de su empresa frente a factores imprevistos y prolongados.
- ✓ **Escalabilidad bajo demanda para adaptarse a los cambios futuros:** Cuando necesite más capacidad y potencia, bastará con añadir otro servidor estándar, lo cual supone una gran flexibilidad para la empresa y un ahorro de hasta el 40% en costes de hardware.
- ✓ **Menores costes de administración**: La flexibilidad, la tolerancia a los fallos y el hecho de tener un solo punto de control hacen que con RAC se pueda reducir a la mitad los costes de administración.

Beneficios de Real Application Clusters (RAC).

- ✓ **Alta Disponibilidad:** Provee una infraestructura de alta para Centros de Datos de alta demanda y disponibilidad. Es un componente esencial en la arquitectura de alta disponibilidad, la cual proporciona protección para soluciones de alta accesibilidad.
- ✓ **Recuperación:** Si se produce una falla en una de las instancias de la Base de Datos del RAC, entonces la anomalía sería notificada a las demás instancias del RAC, que automáticamente ejecutarán un proceso de recuperación en la instancia fallida.
- ✓ **Detección de Errores:** Con la utilización de Clusterware, automáticamente se puede monitorear la Base de Datos del RAC y proveer mecanismos para la detención rápida de

fallos tales como: Fast Application Notification que se encarga notificar cualquier anomalía en los componentes del RAC.

- ✓ **Funcionamiento Ininterrumpido:** Real Application Clusters provee un servicio de continua accesibilidad para aplicaciones de alta demanda y disponibilidad. Si un nodo del RAC saliera del sistema producto de una anomalía, la Base de Datos permanecería abierta y los aplicativos continuarían sus operaciones de acceso a los datos. Todo este proceso se realiza con total transparencia a los usuarios de la aplicación.
- ✓ **Escalable:** Real Application Clusters permite que una única base de datos se expanda por múltiples nodos en un grid o red, uniendo los recursos de varias máquinas. Se pueden añadir servidores a la vez que eliminarlos en un clúster sin tiempo de inactividad, es decir, sin detener la Base de Datos, sin importar tampoco la plataforma donde se encuentra instalado el servidor.

Vale destacar que aunque PostgreSQL cuenta con PgCluster herramienta que:

- ✓ Incorpora: Balanceo de Carga, Replicación de Datos, Tolerancia a fallos.
- ✓ Es síncrono, sincroniza el árbol de directorio del Pgsql.
- ✓ Cuando un servidor cae y se vuelve a poner en funcionamiento automáticamente inicia la sincronización de datos.
- ✓ Replicación de datos por consultas (querys).
- ✓ Se replican también objetos adicionales a la data como las funciones definidas por el usuario.
- ✓ Funciona dentro de la misma base de datos.

Aún así se hace necesaria el desarrollo en PostgreSQL de una funcionalidad como Real Application Clusters, la cual presenta gran robustez en cuanto al trabajo con clúster [11]de servidores

[11] Conjuntos o conglomerados de servidores construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen un único servidor.

2-) Funcionalidad para la gestión de la administración automática del espacio de almacenamiento.

Esta es una herramienta que tratara de simplificar la administración del espacio para la Base de Datos de PostgreSQL 8.2. En vez de manejar numeroso archivos para una base de datos, el administrador de la Base de Datos sólo gestiona un pequeño número de grupos de disco. Un grupo de disco es un conjunto de dispositivos físicos de disco que PostgreSQL manejará como una única unidad lógica. Un DBA puede definir un grupo de disco (diskgroup) como el predefinido para una base de datos, y PostgreSQL automáticamente gestionará ese espacio para crear y eliminar los ficheros relacionados con los objetos de la Base de Datos. También deberá permitir la integración con sistemas de almacenamiento tipo RAID o Logical Volume Managers (LVMs).

Con ésta funcionalidad se administraría la capa de almacenamiento con todas las bases de datos que se encuentren en el nodo. A cada nuevo espacio de almacenamiento al que se le de acceso se llama disco. El conjunto de espacios de almacenamiento es un grupo de discos (diskgroup), se daría ésta agrupación para darles diferente nivel de redundancia principalmente.

Ventajas de esta funcionalidad.

- ✓ Automatizaría y simplificaría la disposición óptima de los archivos de datos, archivos de control y archivos de registro.
- ✓ Los archivos de bases de datos se distribuyen automáticamente entre todos los discos disponibles y el almacenamiento de la base de datos se vuelve a equilibrar cada vez que se modifica la configuración del almacenamiento.
- ✓ Proporcionaría redundancia a través del reflejo de los archivos de la base de datos. Se diseñaría específicamente para utilizar servidores y discos de bajo costo.

Por tanto a partir del análisis anteriormente realizado se decidió implementar la gestión automática de almacenamiento para administrar los archivos de la Base de Datos. Con ésta, no se administran los ficheros de referencia de la base de datos de PostgreSQL.

3-) Herramienta para la administración de bases de datos PostgreSQL en un entorno Grid Computing

Entre los beneficios del Grid Computing en el marco de las tecnologías de la información se pueden enumerar:

- ✓ Integrar sistemas y dispositivos heterogéneos. Grid Computing proporciona un conjunto de capacidades de integración horizontal que dirige de forma efectiva los recursos de toda la empresa, e incluso extienden la solución entre múltiples organizaciones.
- ✓ Mejora del coste efectivo de los entornos operativos. A través de la visualización de la consolidación, reservas, compartición y gestión de recursos a través de las funciones heterogéneas de tecnologías de la información, el Grid Computing ayuda a simplificar los entornos operativos y su gestión reduciendo la administración de su supervisión.
- ✓ El Grid Computing es capaz de descubrir dinámicamente y ajustarse a los entornos cambiantes y fluctuantes de las tecnologías de la información. Por esta razón, Grid Computing facilita el establecimiento, restablecimiento y cambios de los parámetros que requieren los negocios respecto a la seguridad y compartición de recursos.
- ✓ Incrementa la capacidad de recursos para responder a las fluctuaciones de la demanda, permitiendo a las organizaciones de tecnologías de la información agregar recursos distribuidos y explotar una capacidad no utilizada, los Grids incrementan de forma importante la cantidad de recursos computacionales y de datos disponibles.
- ✓ Aumenta la fiabilidad de la infraestructura sacando ventaja de los recursos del Grid como una alternativa ante la recuperación de los desastres tradicionales.

PostgreSQL Grid Control: El principal valor añadido que introduciría PostgreSQL Grid Control (PGC) sería el de mantener constantes los costes de administración, aunque aumente el número de sistemas. PGC se diseñaría para simplificar las complejas tareas a las que se enfrentan los administradores de grandes entornos. Dotado de robustas funcionalidades para administrar sistemas, tanto grandes como pequeños, PGC deberá automatizar las operaciones críticas para reducir el tiempo de implementación y el riesgo de errores, que aumentan a medida que crece el número de sistemas.

PostgreSQL Grid Control deberá incluir monitorización completa, gestión del rendimiento, bases de datos distribuidas y administración de servidor de aplicaciones, diagnóstico avanzado, tuning automático y una arquitectura que permite a los administradores:

- ✓ Administración y supervisión de las aplicaciones y sistemas basados en tecnología PostgreSQL incluyendo la base de datos PostgreSQL.
- ✓ Interfaz 100% basado en web, fácil de desplegar y usar.
- ✓ Deberá funcionar con una aplicación llamada J2EE PostgreSQL Management Service, la cual tendrá un único repositorio y agentes distribuidos a lo largo de todos los sistemas administrados.
- ✓ Aprovisionamiento automático.

La figura muestra la interacción de cada uno de los componentes asociado a PGC. La configuración que se representa es resultado del proceso de instalación de PostgreSQL Grid Control usando una nueva Base de Datos.

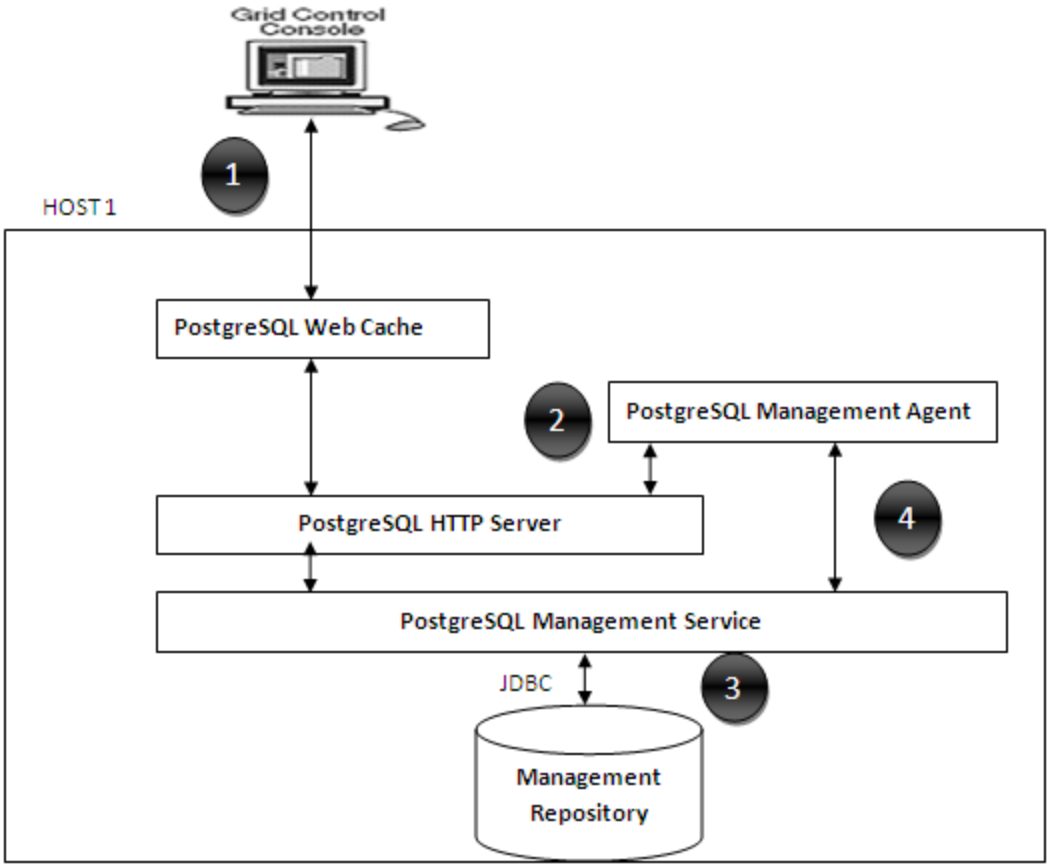

Figura 10 Interacción de los componentes asociado a OGC.

Componentes de PGC:

- ✓ **Grid Control Console:** La Consola de Grid Control se emplea para monitorear y administrar los targets (host y sus componentes) que son descubiertos por el componente PostgreSQL Management Agent instalado en cada host monitoreado.
- ✓ **PostgreSQL Management Agent:** PostgreSQL Management Agent se encargará de la captura y envío de información asociada a cada estación de trabajo o targets (hosts) monitoreado, incluido PostgreSQL Management Service y Management Repository Database a través de PostgreSQL HTTP Server.

- ✓ **PostgreSQL Management Service:** PostgreSQL Management Services (PMS) utilizará JDBC para establecer conexión con PostgreSQL Repository Database, y obtener los datos que serán mostrados al usuario a través de Grid Control Console. PMS empleara PostgreSQL Management Agent URL para monitorear la disponibilidad de PostgreSQL Management Agent.

4 –) Funcionalidad para facilitar el trabajo de un Administrador de Base de Datos (DBA)

A continuación se propondrá una funcionalidad muy importante para facilitar el trabajo del Administrador de Base de datos (DBA). Dicha herramienta se nombraría como:

Management Pack, Esta funcionalidad estaría compuesta por otras tales como:

PostgreSQL Configuration Management Pack: Permitiría la automatización de las tareas que más tiempo consumen y descubrir las causas de los fallos que normalmente son la mayor ocupación de los administradores.

Con Configuration Pack se lograría:

- ✓ Disminuir los costes de manera significativa a la hora de administrar los sistemas.
- ✓ Asegurar la consistencia de los despliegues y cumplimiento de los estándares operativos.
- ✓ Obtener un software rápido y fiable para instalación, actualización, parcheo y clonación.
- ✓ Mejorar las decisiones sobre los despliegues de software. Proporcionará información sobre los problemas de configuración.

Tuning Pack: PostgreSQL Tuning Pack proporcionaría servicios de automatización de procesos y optimización de aplicaciones, permitiendo reducir el coste a la vez que se mejora el rendimiento y la disponibilidad. PostgreSQL Tuning Pack ofrecería un conjunto de tecnologías que automatizan el proceso completo de tuning de aplicación, así se reducen los costes de mantenimiento de la base de datos a la vez que se incrementa el rendimiento y fiabilidad.

Con Tuning Pack se lograría:

- ✓ Obtener recomendaciones automáticas de tuning de sentencias SQL, índices y almacenamiento.
- ✓ Obtener consejos basados en el diagnóstico de rendimiento.
- ✓ Menor coste en la administración y mayor rendimiento de bases de datos.
- ✓ Una utilización de los recursos sensiblemente mejorada.

Diagnostics Pack: Diagnostics Pack incluiría funciones automáticas para la monitorización y diagnóstico integradas en el corazón del motor de base de datos.

Con Diagnostics Pack se lograría:

- ✓ Menor coste de administración y monitorización identificación de problemas proactiva.
- ✓ Una resolución rápida de problemas.

PostgreSQL Change Management Pack: permitiría a los administradores de base de datos realizar cambios complejos al esquema de objetos de forma segura, seguir los cambios realizados en los esquemas y las bases de datos a través del tiempo, hacer copias de esquemas u objetos y compara y sincroniza esquemas y bases de datos. Con esta se puede:

- ✓ Capturar y almacenar definición de objetos.
- ✓ Comparar las definiciones de objetos y poner de relieve las diferencias.
- ✓ Sincronizar definiciones de objetos.
- ✓ Propagar definiciones de objetos a uno o más sitios.
- ✓ Clonar objetos con un subconjunto de sus datos.
- ✓ Administrar y planificar los cambios durante la vida útil de la base de datos y sus aplicaciones.

Conclusiones del capítulo

El presente capítulo es un análisis de los resultados obtenidos luego del proceso de migración a pequeña escala realizado al caso de estudio anteriormente expuesto en el capítulo 2, a través del cual se arrojo como resultado la viabilidad de un proceso de migración del gestor de base de datos Oracle 10g hacia PostgreSQL 8.2.

Como consecuencia de la no existencia de algunas potentes funcionalidades en PostgreSQL, en el documento se ha enfatizado en la propuesta de desarrollo de una serie de funcionalidades que permitan realizar las mismas tareas que algunas implementadas en Oracle para llevar a cabo una gestión potente de los datos y hacer de PostgreSQL un gestor tan potente como Oracle para llevar a cabo el desarrollo libre de aplicaciones informáticas de gran calidad.

Se plantea además la proposición de una herramienta que lleve a cabo el proceso de verificación de migración de los datos, con lo cual se pueda dar máxima seguridad de que los datos fueron migrados satisfactoriamente o no.

CONCLUSIONES GENERALES

Con el desarrollo del trabajo, se considera que han sido cumplido todos los objetivos propuestos inicialmente. Por tanto se puede concluir expresando que:

1- Se describió el desarrollo de la estrategia utilizando una nomenclatura, dividiendo el proceso de migración en 3 grandes grupos fundamentales: Preparación, Migración y Consolidación.
2- En la etapa de Preparación se definieron 3 fases, entre ellas: la justificación de la migración, la planeación de la migración y las pruebas pilotos.
3- El proceso de migración se definió mediante una estrategia trazada para lograr el alcance satisfactorio de la misma.
4- En la fase de consolidación se llevo a cabo la puesta en práctica la migración a través un caso de estudio. Dicho caso de estudio abordó la migración del Sistema Gestor de Base de Datos Oracle 10g (herramienta privativa), hacia PostgreSQL 8.2 (herramienta Open-Source).
5- En el documento se hace la propuesta de una herramienta para la verificación del proceso de migración una vez terminado este, con el fin verificar que la migración tanto del esquema de la base de datos como de sus tablas fue satisfactoria.
6- Se hace la propuesta además de una serie de funcionalidades para facilitar el trabajo del administrador de la base de datos, así como el funcionamiento del Gestor de Base de Datos PostgreSQL 8.2 en un ambiente Grid Computing.

Finalmente, se considera que los temas abordados a lo largo del trabajo puedan contribuir a una futura migración hacia Sistemas Gestores de Base de Datos Libres de los Sistemas Informáticos que se realicen en la Universidad de las Ciencias Informáticas, que estén utilizando Gestores de Base de Datos Privativos. Además se pretende que el documento constituya una referencia para

el Diseño e Implementación de una herramienta que permita viabilizar la migración de Sistemas Gestores de Base de Datos Privativos hacia Sistemas Gestores de Base de Datos Libres mediante la estrategia planteada en el presente trabajo científico.

RECOMENDACIONES

En el desarrollo del documento se ha profundizado en un conjunto de aspectos que se consideran estratégicos en el desarrollo del trabajo; a su vez otros temas solo han sido analizados brevemente a pesar de su marcada importancia. Por tanto este trabajo no queda exento de ello; por lo que a continuación se relacionan un conjunto de ideas que se consideran necesarias para darle continuidad a este trabajo:

- ✓ **Consideraciones para la toma de decisiones tecnológicas**

Antes de ser llevado a cabo el proceso de migración es necesario tener en cuenta que para lograr la total independencia tecnológica deseada se hace necesario no sólo migrar hacia un gestor de base datos libre como se plantea a lo largo de este trabajo sino también se hace muy necesario la migración del entorno de trabajo hacia distribuciones libres, es por ello que a continuación se exponen una serie de consideraciones tecnológicas a tener en cuenta para llevar a cabo la independencia tecnológica. Entre las consideraciones a tomar en cuenta para la toma de decisiones tecnológicas que lleven a la liberación de estrategias de dominación tecnológica, se deben iniciar acciones en el sentido de diversificar los proveedores de hardware, para luego ir desarrollando el ensamblaje y la futura fabricación de equipos de computación. A la par de estas decisiones, se deben trazar planes de migración a software libre.

1.- La estrategia a seguir en un Plan de Migración a Software Libre debe ser la de la investigación, para determinar en primer término cuales son las distribuciones de Linux que estando bajo la categoría de licenciamiento gratuito (GPL), ofrecen una plataforma que cumpla a la vez con ser suficientemente robusta y al mismo tiempo sea lo más sencilla posible de parametrizar, con lo cual se ahorren altos costos de entrenamiento, así como también se reducen

la cantidad de administradores necesarios para mantener varios servidores a la vez, para lo cual se recomienda la distribución FEDORA.

2.- La selección de la distribución Linux a utilizar en los usuarios para reducir el natural impacto del rechazo al cambio, debe considerar que dicha distribución maneje un ambiente lo más parecido posible a Windows, para lo cual se recomienda la distribución XANDROS.

3.- En cuanto a los pasos a seguir para una exitosa migración, es imprescindible comenzar a obligar a tomar contacto por parte del personal de informática con algunas herramientas de software libre que corren sobre Windows. Por ejemplo, primero eliminando los iconos del Internet Explorer e instalando FIREFOX MOZILLA como navegador de Internet; segundo instalando OPEN OFFICE para que comiencen a conocerlo, dando luego una inducción a software libre y un entrenamiento en los cambios en el uso de los iconos y en la corrección de los formatos utilizados en los diversos documentos de office con respecto a las herramientas de Open Office, con esta acción se debe solicitar que los usuarios respalden sus documentos en office y se avisa que pasadas 2 semanas se eliminará el office de sus equipos, para obligar necesariamente al cambio.

4.- Referente a las bases de datos se debe instalar en un servidor Linux, el manejador de base de datos sobre POSGRESQL, de requerirse debe darse el entrenamiento al personal de administración de servidores sobre este manejador, y al personal de desarrollo de sistemas de información sobre modelación de datos y bases de datos relacionales con este manejador de bases de datos, recomiendo utilizar como herramienta metodológica para el análisis y conceptualización de proyectos UML de forma de migrar también a métodos sobre estándares universales para el desarrollo de sistemas. De esta manera se contribuye a la acumulación de conocimiento útil y disponible para todos.

5.- Se debe entrenar al personal de desarrollo de sistemas para la utilización de las herramientas de programación en software libre como PHP, y C++ en software libre (proyecto Mono, el clon de .net). Así como también, se les debe instruir en la investigación para la búsqueda de desarrollos existentes en software libre bajo licenciamiento gratuito, y las estrategias de colaboración con las comunidades de software libre.

6.- Migrar a todo el personal de informática para sistema operativo Linux (en Redes: FEDORA para servidores, en desarrollo y soporte: XANDROS para usuarios), y aplicaciones en Software Libre bajo licenciamiento gratuito (GPL).

7.- Una vez completada la migración, el entrenamiento, y la preparación del personal de informática de manera que estos puedan atender todos los requerimientos de soporte técnico que pudieran requerir los usuarios; se pasaría a la segunda fase que consiste en preparar los planes para la inducción y la migración de los usuarios a software libre, el cual puede constar de las mismas etapas definidas en los pasos 3 y 6 realizadas inicialmente para el personal de informática (SO: FEDORA para los usuarios).

- ✓ **Propuesta de desarrollo de una herramienta para llevar a cabo el proceso de Migración.**

Se propone el diseño e implementación de una herramienta que permita mediante la estrategia planteada en el presente trabajo científico, llevar a cabo el proceso de migración de Sistemas Gestores de Base de Datos Privativos hacia Sistemas Gestores de Base de Datos Libres. Vale destacar la proposición de esta herramienta está dada debido a la gran importancia que puede revestir el uso de la misma en muchos de los Sistemas Informáticos que se realicen en la Universidad de las Ciencias Informáticas, que hayan estado utilizando como Gestores de Base de Datos Privativos. Dicha herramienta será desarrollada bajo la Licencia Publica General, ya que se puede afirmar con total seguridad que la GPL es la licencia libre más importante dentro de las licencias de esta clase al estar utilizada en el 75% del software libre disponible en estos momentos, aparte de la enorme influencia que ha tenido en la elaboración de otras licencias de su tipo. En la sección de “Términos y condiciones…”, la GPL comienza en su apartado 1 definiendo algunos elementos como ‘Programa’ y ‘trabajo basado en el Programa’ pero no desde una perspectiva informática sino con un carácter general de modo tal que acepta que se pueda licenciar tanto un software como cualquier otro trabajo. El ‘Programa’ significa aquí la obra licenciada, sin importar su naturaleza. El ‘trabajo basado en el Programa’ se acoge perfectamente

a la definición de “obra derivada”, muy utilizada en el Derecho de Autor y es aquel que se obtiene como resultado de una adaptación, traducción, extensión o utilización parcial o total del mismo. La GPL establece en su contenido que “*El acto de ejecutar el Programa no está restringido*”, para una claridad mayor veamos estas palabras de Even Moglen[12], profesor de Derecho e Historia de la universidad de California y fundador y abogado de la Free Software Foundation:

“*The license does not require anyone to accept it in order to acquire, install, use, inspect, or even experimentally modify GPL'd software* (…) *The GPL only obliges you if you distribute software made from GPL'd code, and only needs to be accepted when redistribution occurs*.”[13]

Como se comprenderá eso no está permitido en el sistema de Derecho de Autor, por lo que se debe definir en la licencia, de forma clara, que la utilización está completamente permitida. Cualquier persona no puede utilizar un software si no sabe bajo qué condiciones se pone a su disposición. Cuando la licencia GPL dice que sólo regula la copia, modificación y distribución del software indica que las demás acciones se regulan por una legislación aplicable a estos particulares y en su defecto a la ley de derecho de autor vigente. En el caso de Cuba las licencias que concede el licenciante se regulan por la Resolución Conjunta 1 de 1999[14] y en lo no estipulado por esta por la Ley 14 de 1977, Ley de Derecho de Autor[15].

El derecho a la distribución de las copias, regulada en la cláusula número 1 de la GPL, implica la posibilidad del usuario de reproducir y distribuir el trabajo. Se trata de facultades que se enmarcan como derechos patrimoniales del autor y que son exclusivos tanto en el modelo

[12] Visitar su sitio para más información: http://emoglen.law.columbia.edu/
[13] Eben Moglen, *Free Software Matters: Enforcing the GPL, I*.. Disponible en: http://emoglen.law.columbia.edu/publications/lu12.html. Visitado el 1552006
[14] Resolución Conjunta 1, Reglamento para la protección de los programas de computación y bases de datos, en cuanto a su creación, concertación de contratos, así como su explotación comercial. 21 de junio de 1999. Ministerio de Cultura y Ministerio de la Industria Sidero Mecánica. República de Cuba. Disponible en: http://www.cenda.cu/doc/Cjta1.pdf.
[15] Art. 7. Ley 14. Ley del Derecho de Autor. 28 de diciembre de 1977. República de Cuba. Disponible en: http://www.cenda.cu/doc/Ley14_77.pdf. Visitado el 28 de diciembre de 2005.

tradicional y como en el del software (derecho de reproducción). Sin embargo en el modelo de software libre esta posibilidad siempre está presente para el licenciatario.

En lo que respecta al plano económico se debe hacer una salvedad. La GPL establece que se podrá cobrar por esta actividad pero sólo por el acto físico de transferir o distribuir la copia, lo cual incluiría el precio del soporte sobre el que se distribuye (normalmente disquete o disco compacto), el precio de envío por correo y cualquier otro gasto que se requiera para efectuar la transferencia. En realidad no puede cobrar por más nada porque lo único que se hace es redistribuir, no se crea nada, y por esa misma razón tiene que mantener intacto todas las referencias a la autoría del programa y a la licencia que lo acompaña (GPL).

BIBLIOGRAFÍA

1. Gnu. *Gnu.org.* [En línea] Free Software Foundation, Inc, 2001. [Citado el: 24 de 11 de 2007.] http://www.gnu.org/philosophy/free-sw.es.html .

2. Gnu. *Gnu.org.* [En línea] Free Software Foundation, Inc, 2006. [Citado el: 24 de 11 de 2007.] http://www.gnu.org/philosophy/categories.es.html.

3. PostgreSQL. *PostgreSQL.org.* [En línea] The PostgreSQL Global Development Group, 2005. [Citado el: 29 de 11 de 2007.] http://www.postgresql.org/docs/8.0/interactive/index.html.

4. PostgreSQL. *PostgreSQL.org.* [En línea] PostgreSQL Global Development Group , 2008. [Citado el: 30 de 11 de 2007.] http://www.postgresql.org/about/press/presskit82.html.es.

5. **Pozo, Salvador.** MySQL con CLASE. *Mysql.conclase.net.* [En línea] MySQL, 3 de 2004. [Citado el: 1 de 12 de 2007.] http://mysql.conclase.net/curso/index.php?cap=006.

6. MySQL. *MySQL.com.* [En línea] MySQL AB, 2008. [Citado el: 1 de 12 de 2007.] http://dev.mysql.com/doc/refman/5.0/es/replication-compatibility.html.

7. Infoworld. *iworld.com.mx.* [En línea] 2 de 1 de 2006. [Citado el: 2 de 12 de 2007.] http://www.iworld.com.mx/iw_TestCenter_read.asp?iwid=81.

8. **Gomez, Javier.** Versioncero. *versioncero.com.* [En línea] 3 de 10 de 2005. [Citado el: 3 de 12 de 2007.] http://www.versioncero.com/noticia/391/mysql-50-release-candidate-1.

9. Up to Down. *UptoDown .com.* [En línea] Media Ingea SL, 2008. [Citado el: 4 de 12 de 2007.] http://mysql-gui-tools.uptodown.com.

10. MySQL Hispano. *mysql-hispano.org.* [En línea] MySQL, 16 de 10 de 2006. [Citado el: 4 de 12 de 2007.] http://www.mysql-hispano.org/index.php?m=read&id=974.

11. MySQL. *MySQL.com.* [En línea] MySQL AB, 2008. [Citado el: 5 de 12 de 2007.] http://dev.mysql.com/doc/refman/5.0/es/index.html.

12. PC Actual. *pc-actual.com.* [En línea] RBA, 2008. [Citado el: 6 de 12 de 2007.] http://www.pc-actual.com/Actualidad/Reportajes/Inform%C3%A1tica_profesional/Empresas/20071025039/2.

13. Suite101. *Suite101.com.* [En línea] Media Inc. [Citado el: 7 de 12 de 2007.] http://www.suite101.com/article.cfm/oracle/115560/2.

14. **James Shannon, Ben Adida, and Don Baccus.** Oracle to Postgres Conversion. *openacs.org.* [En línea] [Citado el: 8 de 12 de 2007.] http://openacs.org/doc/openacs-3/html/oracle-to-pg-porting.html.

15. Fermilab Computing Division. *css.fnal.gov.* [En línea] Fermilab, 14 de 3 de 2005. [Citado el: 9 de 12 de 2007.] www-css.fnal.gov/dsg/external/freeware/mysql-vs-pgsql.html.

16. MasterMagazine. *mastermagazine.info.* [En línea] 2004. [Citado el: 10 de 12 de 2007.] www.mastermagazine.info/articulo/grid_b.php.

17. **Free Software Foundation, Inc.** El sistema Operativo GNU. [En línea] 2007. [Citado el: 26 de 11 de 2007.] http://www.gnu.org/philosophy/philosophy.es.html#LicensingFreeSoftware.

18. **Centro Nacional de Derecho de Autor CENDA.** BIblioteca Digital Informática. [En línea] 2006.[Citado el: 3 de 1 de 2008.].
http://www.ispcmw.rimed.cu/sitios/digbiblio/cont/EI/SO_Linux/SoftLibre/EstSoftLibre.htm.ISBN 959-18-0169-6.

19. **Oracle Corporation.** Oracle. [En línea] Oralce Developer Comunity, 2007. [Citado el: 3 de 1 de 2008.] http://www.oracle.com/global/es/index.html.

20. **PostgreSQL Corporation.** PostgreSQL. [En línea] PostgreSQL Global Development Group, 2007. [Citado el: 4 de 1 de 2008.] http://www.postgresql.org/.

21. ElServer.com web hosting profesional. [En línea] 2007. [Citado el: 10 de 1 de 2008.] http://www.elserver.com/hosting/grid/.

22. [En línea] http://bakara.files.wordpress.com/2007/07/uxi_3.pdf.

23. **Moral, Juan Carlos.** juancarlosmoral.es. [En línea] 22 de 4 de 2007. [Citado el: 7 de 1 de 2008.] http://www.juancarlosmoral.es/index.php/postgresql-acid-test/.

24. **Informáticas, Universidad de las Ciencias.** WikiProd. [En línea] 2007. [Citado el: 15 de 1 de 2008.] http://wiki.prod.uci.cu.

25. **Herrera, Álvaro.** *Funciones y Triggers-Introducción, Ejemplos.* 2004.

26. **Townsend, Ethan.** *Conversion of Microsoft SQL/ASP applications to PostgreSQL.* 2005.28. **Momjian, Bruce.** *PostgreSQL Introduction and Concepts.*

27. **PROAÑO, ING. DIEGO JAVIER BURBANO.** *ANALISIS COMPARATIVO DE BASES DE DATOS DE CODIGO ABIERTO VS CODIGO CERRADO (DETERMINACION DE INDICES DE COMPARACION).* Quito-Ecuador : s.n., 2006.

28. **Arguello, Ricardo.** *Las bases de datos Open Source ya juegan en Ligas Mayores.* 2003.

29. **Gascón, Manuel de la Herrán.** Red Cientifica. *redcientifica.com.* [En línea] 2004. [Citado el: 12 de 12 de 2007.] http://www.redcientifica.com/oracle.

30. **Restrepo, Ivan Alonso Montoya.** Que es estrategia? *Universidad Nacional de Colombia.* [En línea] 10 de 2 de 2004 . [Citado el: 13 de 12 de 2007.] http://www.virtual.unal.edu.co/cursos/economicas/2008551/lecciones/cap1-4-1.htm.

31. **PROAÑO, DIEGO JAVIER BURBANO.** *Bases de Datos Open Source Vs Bases de Datos Comerciales.* Quito-Ecuador : s.n., 2006.

32. **netproject.** *Directrices IDA de migracion a software de fuentes abiertas.* 2003.

33. **El equipo de desarrollo de PostgreSQL.** *Guia del Administrador de PostgreSQL.*

34. **Kuroki, Christian.** *Migración a PostgreSQL desde otras bases de datos.* 2005. 15-5456-7138.

35. **Ruiz, Santiago Gómez.** *Microsoft SQL Server,MySQL y PostgreSQL.* 2007.

36. **Cunningham, Lewis.** Oracle 10g vs PostgreSQL 8 vs MySQL 5. *ittoolbox.com.* [En línea] 8 de 22 de 2005. [Citado el: 14 de 12 de 2007.] http://blogs.ittoolbox.com/oracle/guide/archives/oracle-10g-vs-postgresql-8-vs-mysql-5-5452.

37. **The PostgreSQL Global Development Group.** *PL/pgSQL - SQL Procedural Language.* 2003.

38. **El equipo de desarrollo de PostgreSQL.** *Manual del usuario de PostgreSQL.*

39. Llegó la hora del PostgreSQL. *http-peru.com.* [En línea] 2007. [Citado el: 15 de 12 de 2007.] http://www.http-peru.com/index.php.

40. **netproject.** *Directrices IDA de migración a software de fuentes abiertas.* 2003.

41. **The PostgreSQL Global Development Group.** PL/pgSQL - SQL Procedural Language. *sobl.org.* [En línea] Proyecto S.O.B.L, 2000. [Citado el: 16 de 12 de 2007.] http://www.sobl.org/traducciones/postgresql-develdoc/index.html.

42. **Mora, Marc Gibert Ginestŕ y Oscar Perez.** *Bases de Datos en PostgreSQL.*

43. **Lara, Jesús Ignacio.** *Bases de datos en Software Libre.*

44. **A., Ernesto Quiñones.** *SERVICIOS DE ALTA DISPONIBILIDAD DE BASES DE DATOS CON POSTGRESQL.*

45. **CASTELLANOS, Ma. ARGENIS GONZALEZ.** *COMPARACION ENTRE SISTEMAS DE GESTION DE BASES DE DATOS (SGBD) BAJO LICENCIAMIENTO LIBRE Y COMERCIAL.* Bogota, Colombia : s.n., 2005.

46. SGBD libres. *iespana.es.* [En línea] 18 de 10 de 2002. [Citado el: 17 de 12 de 2007.] http://www.iespana.es/suevenos/consultas_html/190.htm.

47. **Culebro Juarez, Montserrat.Gomez Herrera, Wendy Guadalupe.** *Software libre vs software propietario - Ventajas y desventajas.* Mexico : s.n., 2006.

48. **Oracle.** *ORACLE ENTERPRISE MANAGER 10g CONFIGURATION MANAGEMENT PACK FOR ORACLE DATABASE.* 2007.

49. **A., Ernesto Quiñones.** *INTRODUCCION A POSTGRESQL.*

50. **Fundación Código Libre Dominicano.** *Introducción a Base de Datos Con PostgreSQL.*

51. **Medina, José Manuel Alarcón.** *Administración SGBD PostgreSQL.* 2006.

52. **A., Ernesto Quiñones.** *SERVICIOS DE ALTA DISPONIBILIDAD DE BASES DE DATOS CON POSTGRESQL.*

53. **Espinoza, Humberto.** *PostgreSQL Una Alternativa de DBMS Open Source.* 2005.

ANEXOS

1- Comparación entre los tipos de datos de Oracle 10g y PostgreSQL 8.2

Oracle	PostgreSQL
NUMBER (p), donde p es la precisión, es decir el número de dígitos.	SMALLINT - 2 bytes
	INTEGER - 4 bytes
	BIGINT - 8 bytes
NUMBER (p, s), donde p es el número total de dígitos y s es la cantidad de dígitos después de la coma para decimales.	NUMERIC(p,s)
	REAL – 4 bytes, 6 lugares decimales
	DOUBLE PRECISION – 8 bytes, 15 lugares decimales
No existe nada similar.	SERIAL – 0 a +2147483647 típicamente usado para crear identificadores únicos. Genera una sucesión implícita que se incrementa cuando una línea se inserta en la tabla
VARCHAR2 (p), donde p es el tamaño, su mayor valor es 4000.	CHARACTER VARYING(n), donde n es el tamaño y tiene un máximo de 1GB. VARCHAR(n) es un alias
CHAR(r), donde r es el tamaño, su máximo es de 2000.	CHARACTER(n), donde n es el tamaño, su máximo es de 1GB CHAR(n) es un alias. Se sugiere que se use el tipo TEXT si n> 10 MB.
LONG - Datos del carácter de longitud variable para datos de longitud hasta 2 GB.	TEXT – longitud variable hasta 1 GB Se sugiere que se use el tipo TEXT si n> 10 MB.
DATE – Retiene fecha y hora	TIMESTAMP Se puede usar getDate () para leer una columna de tipo TIMESTAMP, pero se perderá la parte de la hora del dato.
No existe nada similar.	DATE – Retiene sólo la Fecha.

No existe nada similar.	TIME – Retiene sólo la hora (00:00:00.00 – 23:59:59.99).
RAW (p) –datos binarios de bytes de tamaño de longitud (máximo 2000). LONG RAW – datos binarios o de longitud variable hasta 2 GB	BYTEA
No existe nada similar.	BIT(n) – Longitud de cadena arreglado de 1's y 0's BIT VARYING(n) – Longitud de cadena variable de 1's y 0's
CLOB – Objeto caracter grande(max 4GB)	TEXT (max 1GB)
BLOB – Objeto binario grande (max 4GB)	BYTEA (max 1GB) BYTEA no se documenta en PostgreSQL 7.1 pero se lleva a cabo totalmente. Además de TEXT y BYTEA, PostgreSQL soporta objetos grandes como archivos separados. Ellos son guardados en una tabla separada en un formato especial, y son enviados desde tablas regulares por un valor identificador de objeto (OID). Más información: *http://www.postgresql.org/idocs/index.php?largeobjects.html* *http://www.postgresql.org/idocs/index.php?jdbc-lo.html*
ROWID	No existe nada similar.
No existe nada similar. Típicamente, char (1) se usa para guardar un valor eso se traduce a Boolean en la lógica de la aplicación. Si se guarda 0' y 1' en un varchar (1) o en una columna char (1), entonces el driver jdbc puede correctamente interpretar estos valores como bolean falso y verdadero usando respectivamente ResultSet.getBoolean y PreparedStatement.setBoolean	BOOLEAN – Puede tener como valores: true, false o null. Si se guarda 0' y 1' en un varchar (1) o en una columna char (1), entonces el driver jdbc puede correctamente interpretar estos valores como bolean falso y verdadero usando respectivamente ResultSet.getBoolean. Sin embargo, PreparedStatement.setBoolean simplemente no hace el trabajo.
Rasgos espaciales Oracle	Tipos de datos geométricos: POINT, LINE,

	CIRCLE, etc.
No existe nada similar.	Tipos de datos de direcciones de red: INET, MACADDR, CIDR.

2- Utilización de licencias en proyectos de programas de software libre.

Como se puede apreciar en las siguientes cifras [16]es la licencia más utilizada en los proyectos que se están desarrollando.

Licencia	Proyectos
GPL	66,1 %
LGPL	10,9 %
BSD	10,9 %
Con certificado Open Source Iniciative	12,1 %
Resto	4,0 %

[16] Tomado del sitio: http://sourcefotge.net/. Visitado 16 novienmbre 2007

1