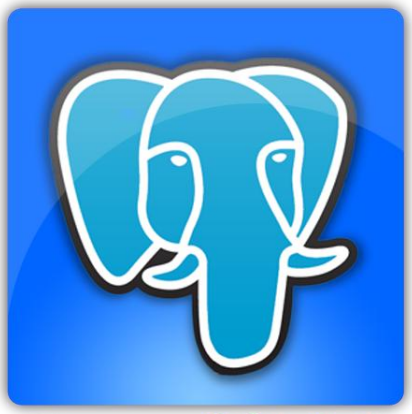
Universidad de las Ciencias Informáticas

Facultad 8



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

Guía para la personalización de PostgreSQL 8.4

Autores: Yeni Morgado Sánchez

Aldo Cristiá Alvarez

Tutores: Ing. Yudisney Vazquez Ortíz

Ing. Marcos Luis Ortíz Valmaseda

Ciudad de La Habana, junio de 2010

Año 52 de la Revolución

"Emplearse en lo estéril cuando se puede hacer lo útil, dedicarse a lo fácil cuando se tienen bríos para intentar lo difícil, es despojar de su dignidad al talento. Todo el que deja de hacer lo que es capaz de hacer, peca."

José Martí

Declaración de autoría

Por este medio declaramos ser los únicos autores de la presente tesis y cedemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Firma del autor	Firma del autor
Yeni Morgado Sánchez	Aldo Cristiá Alvarez
Firma del tutor	Firma del cotutor
Ing. Yudisney Vazquez Ortíz	Ing. Marcos Luis Ortíz Valmaseda

Opinión de los tutores del trabajo de diploma

Título: Guía para la personalización de *PostgreSQL* 8.4

Autores: Yeni Morgado Sánchez

Aldo Cristiá Alvarez

Por todo lo anteriormente expresado, consideramos que los estudiantes están aptos para ejercer como Ingenieros en Ciencias Informáticas, y proponemos que se le otorgue al Trabajo de Diploma la calificación de ___ puntos.

Firma del tutor	Firma del tutor
Ing. Yudisney Vazquez Ortíz	Ing. Marcos Luis Ortiz Valmaseda

Dedicatoria de Yeni

Dedico este trabajo primeramente a mis maravillosos padres Nevys de los Angeles Sánchez Machado y Arley Morgado Alvarez por todo el cariño y amor que me han brindado siempre, por la confianza y esfuerzo que depositaron en mí para que este gran sueño se hiciera realidad. Los quiero mucho.

A mi hermana y a mis adorados abuelos por todo el cariño que me brindan siempre y por estar presentes cada vez que los necesito.

A toda mi familia, por su aliento constante y su cariño sin fronteras.

A mi novio Aramis, por todo el amor, paciencia y comprensión que me ha brindado desde que lo conocí. Te adoro.

Dedicatoria de Aldo

A mi madre, mi Rosa más bella en el jardín de la vida, por dedicar toda su vida a la mía. Por confiar y creer en mí en todo momento. Por ser más que un ejemplo de madre y de amiga.

A mi padre, que no sabe ni siquiera que su hijo es ya todo un hombre.

A mi abuela Eneida: Por ser la estrella que ilumina mi vida cuando se torna oscura. Por haberse esforzado tanto en mi educación con el afán de que siempre fuera un hombre de bien. Por tantos regaños y tanta comprensión, por tanta ayuda y sobre todo tanto amor.

A mi abuelo Miguel: Por ser mi abuelo, mi amigo y mi padre. Por nunca haberme ni siquiera regañado, porque de ti lo único que siempre he recibido ha sido tu cariño y tu comprensión. Por tantas enseñanzas, por demostrarme que siempre las cosas se

pueden hacer mejor, porque cada consejo tuyo en mi niñez fue una herramienta para la vida.

A mis tíos Aldo y Nelson, por ser ejemplos y guías a seguir durante toda mi vida. Porque aunque no sean perfectos, siempre han querido lo mejor para mí.

A mi padrastro, porque aunque no seas mi padre de sangre sé que me quieres como si lo fueras. Por el orgullo con que dices que soy tu hijo. Por demostrarme que aunque la vida nos juegue una mala pasada siempre hay que mantener la fe y la esperanza.

A mi hermana Kenia, porque sé que me quieres mucho, solo espero que esto te sirva de ejemplo para que algún día tú te gradúes como universitaria y entonces sea mi nombre el que aparezca en la dedicatoria.

A mis primos Nelson y Aldo Manuel, por crecer juntos como hermanos y querernos como tal.

Agradecimientos

A la Revolución y a Fidel, por habernos dado la posibilidad de estudiar en esta Universidad de Excelencia y graduarnos como profesionales al servicio de esta Revolución.

A nuestros tutores por brindarnos toda su confianza desde el primer momento. Por convertirse en nuestros amigos de todos los días. Por apoyarnos y ayudarnos a que este sueño se hiciera realidad, por decirnos cuál era el camino y por recorrerlo junto a nosotros.

A todos los que nos apoyaron, contribuyeron y confiaron en nosotros y a los que no lo hicieron también.

Agradecimientos de Yeni

A mis padres, por todo su sacrificio y dedicación, por escucharme siempre y saber comprenderme y por estar presentes cada vez que los necesito.

A mi abuela Emerilda, que en estos momentos ya no se encuentra conmigo, quiero agradecerle porque sé que está muy orgullosa de que este sueño se cumpliera.

A mi hermanita Yele, por brindarme todo su cariño, apoyo y por confiar en mí.

A mis abuelitos queridos Eneida y Ramiro, por todo su amor, cariño y dedicación.

A mi novio Aramis, agradecerle con el corazón que en estos años de la carrera ha sido mi guía, mi apoyo y mi luz. Agradecerle por ser la personita que ha estado ahí cuando más lo he necesitado, por brindarme esos momentos tan bonitos que pase junto a él, gracias y nunca olvides que TE AMO ARA.

A mis tías y tíos, por brindarme todo su amor y dedicación.

A mis primas y primos, agradecerles por confiar en mí y por su cariño incondicional.

A mi inmensa familia, agradecerles por estar pendientes de mí, por su apoyo, cariño incondicional y por toda la ayuda que me han brindado siempre. Gracias por ser como son únicos.

A mis suegros Cruz y Alina, por brindarme todo su cariño y apoyo incondicional.

A mis amigas y amigos, especialmente a Nely, Aliza, Alién, Daniel, Yoel y Sandor por brindarme todo su apoyo, comprensión y por sus buenos consejos.

A mis compañeros de la UCI, por su apoyo y por estar siempre presente cuando los necesito.

A todos los profesores que influyeron en mi carrera que de una manera u otra han contribuido a mi formación profesional.

A mi compañero de tesis, por su comprensión y esfuerzo para que el trabajo saliera lo mejor posible.

Gracias a todos.

Agradecimientos de Aldo

A mi mamá Rosa, por querer siempre lo mejor para mi, por apoyarme en todas y cada una de mi decisiones. Por la confianza que me tienes aunque a veces pelees tanto. A ti te doy las gracias por traerme al mundo y por estar a mi lado siempre aunque estuvieras lejos. Felicidades por hacerte Ingeniera.

A mi otra madre y abuela Eneida, te agradezco haber estado junto a mí desde que nací, por cuidarme, protegerme y educarme. Por vivir pendiente de mí como siempre lo has hecho y por sacrificarte tanto para darme todo lo que me has dado.

A mi abuelo Miguel, mi Panga gracias por quererme y protegerme siempre, sin ti tal vez este sueño nuestro no se nos hubiese hecho realidad.

A mis tíos Aldo y Nelson, por ser los paradigmas con los que siempre me comparo. Algún día espero llegar a ser como ustedes.

A mi tía Yanela, por darme tanto amor y cariño, aunque me hallas partido la nariz una vez.

A mi padrastro, por tanta preocupación y tanta confianza en mí.

A mi hermana, por quererme tanto aunque me lo demuestre fajándose conmigo.

A mi primo Nelsito, por siempre caminar delante cuando vamos por el mismo camino, te doy las gracias por todo lo que has hecho por mí, espero algún día poder retribuírtelo.

A todos los demás primos, por querernos y cuidarnos tanto unos a los otros.

A todos los profes, desde aquella que me castigaba y me amenazaba con golpearme con una regla hasta el que me enseño que cada día debemos levantarnos con ganas de ser mejor persona en todos los aspectos de la vida.

A mi novia Elizabeth, por estar a mi lado soportando mis pesadeces tanto tiempo, te doy las gracias por quererme tanto en todo momento.

A todos mis amigos, que son muchos, pero todos están dentro de mí. Para los que están y los que no, para los que siempre lo han sido y para los que por una cuestión u otra dejaron de serlo, muchas gracias por brindarme su amistad.

A mi compañera de tesis, por tanta dedicación, gran parte del éxito del presente trabajo se lo debo a ella.

A todos, muchas gracias.

Resumen

La Universidad de las Ciencias Informáticas (UCI) tiene entre sus objetivos favorecer el desarrollo de la economía cubana y contribuir a la informatización de las instituciones del país. Su infraestructura productiva está conformada por direcciones y centros de desarrollo enfocados al cumplimiento de estos propósitos. El Centro de Tecnologías de Gestión de Datos tiene entre sus misiones la de desarrollar nuevas tecnologías de bases de datos, de procesamiento y representación de la información con un claro enfoque a la soberanía tecnológica. En la presente investigación se propone una guía para la personalización de *PostgreSQL* 8.4, sistema de gestión de bases de datos que adopta el Centro como base para sus desarrollos. Para lograr los objetivos de la investigación se realizó un estudio de los gestores de bases de datos; de *PostgreSQL*; de los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad que pueden ser incluidos en su núcleo y de los compiladores existentes para tal objetivo, seleccionando a GCC como el idóneo para realizar la compilación. La Guía está compuesta por un procedimiento de instalación y configuración de *PostgreSQL* 8.4 desde el código fuente, una descripción detallada de los módulos, un procedimiento para compilarlos, un conjunto de recomendaciones y buenas prácticas para habilitarlos y un procedimiento para su desinstalación, constituyendo un producto a la altura de las necesidades de los proyectos del Centro.

Palabras claves: módulos del *contrib*, personalización, *PostgreSQL*, sistemas de gestión de bases de datos

Índice de contenidos

Índice de tablas

Índice de figuras

Introducción

Las Tecnologías de la Información y las Comunicaciones (TICs) son actualmente actor fundamental en el desarrollo de la economía cubana. La Universidad de las Ciencias Informáticas (UCI), surgida al calor de la Batalla de Ideas, nace con el propósito de favorecer dicho desarrollo, contribuyendo a la informatización de las instituciones del país.

Con vistas a lograr este propósito, la UCI organizó una Infraestructura Productiva (IP) que se encarga de gestionar los proyectos de producción de software. La IP a lo largo de su proceso de maduración ha ido organizándose en función de lograr una producción lo más eficiente posible; surgiendo a finales del curso 2007-2008 la idea de los centros de desarrollo, conceptualmente pequeñas empresas enfocadas a la producción de software en una rama determinada. De esta forma a principios del curso 2008-2009 se conforma el Centro de Tecnologías de Gestión de Datos (DATEC) con las misiones de:

- Proveer soluciones integrales, soporte y consultorías relacionadas con tecnologías de bases de datos y análisis de información.
- Desarrollar nuevas tecnologías de bases de datos, de procesamiento y representación de la información a partir del desarrollo de proyectos de Investigación + Desarrollo (I+D), con un claro enfoque a la soberanía tecnológica.
- Contribuir con su trabajo al cumplimiento de las misiones fundamentales de la Universidad: la formación y la producción de software. Con profesionales integrales comprometidos y con un alto nivel científico y productivo.

Con tales propósitos DATEC adopta al gestor de bases de datos *PostgreSQL* como base para sus futuros desarrollos.

El 1 de julio del 2009 fue liberada la versión 8.4 de *PostgreSQL*. Los proyectos del Centro que trabajan sobre ella se enfrentan a 4 problemas fundamentales:

- Inexistencia de módulos especializados en análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos dentro del núcleo de *PostgreSQL* 8.4, que son los elementos con que trabajan.
- Inexistencia de documentación asequible para comprender las funcionalidades de los módulos.

- Inexistencia de guías para la selección de qué módulos instalar según las necesidades del proyecto.
- Inexistencia de documentación asequible para comprender la instalación y configuración de dichos módulos.

Estos elementos influyen negativamente en el desarrollo exitoso de los proyectos del Centro. La presente investigación surge para dar solución al **problema científico** que se deriva de la situación existente en DATEC: las insuficiencias del gestor de bases de datos *PostgreSQL* 8.4 relacionadas con el análisis de datos, el monitoreo, la administración, el desarrollo, la seguridad en las bases de datos y de guías para configurarlo, dificultan las posibilidades de su explotación y generalización en el desarrollo de soluciones empresariales en el Centro de Tecnologías de Gestión de Datos.

Se define como **objeto de estudio**: el sistema gestor de bases de datos *PostgreSQL* y como **campo de acción**: los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad para el gestor de bases de datos *PostgreSQL* 8.4 relacionados con las líneas de producción del Centro de Tecnologías de Gestión de Datos.

Para resolver el problema se trazó como **objetivo general**: definir una guía para la personalización de *PostgreSQL* 8.4 para soluciones de análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos.

A partir del análisis del objetivo general se derivaron los siguientes **objetivos específicos**:

- Realizar un estudio del estado del arte de los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos que existen para *PostgreSQL* y sus potencialidades, además de los productos y servicios que ofrecen las empresas líderes basados en él.
- Compilar los módulos en el *PostgreSQL* 8.4 para el sistema operativo *Debian GNU/Linux Lenny 5.0*[1].
- Definir una guía para la compilación de *PostgreSQL* 8.4 desde el código fuente, la incorporación y eliminación de los módulos al Gestor, con sus descripciones, recomendaciones y buenas prácticas para habilitarlos.

[1] Se escoge el sistema *Debian GNU/Lenny* 5.0 por su estabilidad, rapidez, poca utilización de memoria y por su sistema de empaquetamiento de software.

- Validar la propuesta.

La investigación se sustenta en la **idea a defender** de que: una guía para la personalización de *PostgreSQL* 8.4 con módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos, facilitará las posibilidades de explotación y generalización del gestor para soluciones empresariales que se desarrollan en el Centro de Tecnologías de Gestión de Datos.

Para dar cumplimiento a los objetivos específicos se trazaron una serie de **tareas a cumplir**:

- Definición del diseño teórico-metodológico de la investigación.
- Realización del estudio del estado del arte de los módulos de *PostgreSQL* y de las empresas líderes en servicios y software basados en él.
- Selección del compilador a utilizar para el empaquetamiento de los módulos.
- Definición del procedimiento para la instalación de *PostgreSQL* 8.4 desde su código fuente.
- Descripción detallada de los módulos a ser compilados en el *PostgreSQL* 8.4.
- Definición del procedimiento para la compilación de los módulos en *Debian GNU/Linux Lenny 5.0.*
- Definición del procedimiento para la desinstalación de los módulos.
- Definición de las recomendaciones y buenas prácticas para habilitar los módulos.
- Validación de la guía.

Luego de la realización de las tareas antes mencionadas se espera tener los siguientes **resultados**:

- Guía metodológica para la incorporación de módulos al *PostgreSQL* 8.4, con recomendaciones y buenas prácticas para habilitar cada uno de los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad.
- Fuentes de *PostgreSQL* 8.4 con los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad incluidos en el directorio *contrib*.

En la realización de esta investigación se utilizan los siguientes **métodos científicos**:

- Analítico-Sintético: este método se utiliza porque posibilita el estudio del estado del

arte de los componentes y funcionalidades del gestor de bases de datos *PostgreSQL*, además de los módulos que existen para dicho gestor de análisis de datos, monitoreo, administración, desarrollo y seguridad para describirlos detalladamente.

- Histórico-Lógico: este método permite realizar un estudio detallado del desarrollo de *PostgreSQL* y su empleo en el mundo, además de que mediante él se expresarán los aspectos más importantes de su trayectoria desde su surgimiento hasta la actualidad.

La investigación consta de tres capítulos:

- **Capítulo 1.- Fundamentación teórica:** se realiza un estudio del estado del arte del sistema gestor de bases de datos *PostgreSQL;* se analizan las empresas que han tenido éxito y que basan sus productos y servicios en él y se define el compilador a emplear para la realización de la propuesta.
- **Capítulo 2.- Descripción de la solución propuesta:** se describen detalladamente los módulos de análisis de datos relacionados con las líneas de producción del Centro; se definen los procedimientos para la instalación y configuración del Gestor desde su código fuente, para la compilación de los módulos y para la eliminación de los mismos, conformando la guía para la personalización del Gestor.
- **Capítulo 3.- Validación de la propuesta:** se refleja la evaluación técnica de la propuesta.

Capítulo 1
Fundamentación teórica

Introducción

En el presente capítulo se mostrará el resultado del estudio del estado del arte entorno al objeto de estudio y campo de acción de la investigación. Se explican las características de los sistemas de gestión de bases de datos, sus componentes, su clasificación, las funciones que realizan, además de las ventajas y desventajas de su utilización. Se define *PostgreSQL*, su reseña histórica, las ventajas y desventajas que posee, su arquitectura, las funcionalidades que soporta y las empresas que le brindan apoyo al gestor. Se hace un estudio además de los módulos del *contrib* del gestor que se tendrán en cuenta para la propuesta de solución. Por último se estudian los compiladores existentes y se selecciona el idóneo para la inclusión de los módulos en el núcleo del gestor.

1.1.- Sistemas de gestión de bases de datos

Muchos especialistas en el tema han dado sus propias definiciones sobre el término.

Korth lo define como una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos; su objetivo primordial es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos. (Korth, 1995)

María Marqués lo define como una aplicación que permite a los usuarios definir, crear y mantener bases de datos y proporciona acceso controlado a las mismas. (Marqués Andrés, 2001)

Los *Hansen* lo describen como un software, parecido a un sistema operativo o compilador, que brinda un conjunto de servicios a los usuarios finales, programadores y otros. Como su nombre indica, facilita la gestión de las bases de datos. (Hansen, et al., 1999)

De manera general, para la presente investigación, los sistemas de gestión de bases de datos son un tipo de software dedicado a servir de interfaz entre las bases de datos, los

usuarios y las aplicaciones que las utilizan y que permiten la creación y manipulación de los objetos y las propias bases de datos.

1.1.1.- Componentes de los sistemas de gestión de bases de datos

Los principales componentes de un sistema de gestión de bases de datos son los siguientes: (Marqués Andrés, 2001)

- Control de autorización: comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita.
- Procesador de comandos: una vez que el sistema ha comprobado los permisos del usuario, se pasa el control al procesador de comandos.
- Control de la integridad: cuando una operación cambia los datos de la base de datos, este módulo debe comprobar que la operación a realizar satisface todas las restricciones de integridad necesarias.
- Optimizador de consultas/Planificador: determina la estrategia óptima para la ejecución de las consultas.
- Gestor de transacciones: realiza el procesamiento de las transacciones.
- Gestor de recuperación: garantiza que las bases de datos permanezcan en un estado consistente en caso de que se produzca algún fallo.
- Gestor de búferes: responsable de transferir los datos entre la memoria principal y los dispositivos de almacenamiento secundario. A este módulo también se le denomina gestor de datos.

1.1.2.- Clasificación de los sistemas de gestión de bases de datos

Los sistemas de gestión de bases de datos se clasifican según: (Marqués Andrés, 2001)

- Modelo lógico en el que se basan (lenguaje usado para especificar esquemas lógicos que son descripciones de la estructura de la base de datos en términos de las estructuras de datos que pueden procesar un sistema gestor de bases de datos): (Arencibia, et al., 2007)
    - Modelo de red: los datos se representan como colecciones de registros y las relaciones entre los datos mediante conjuntos (punteros en la implementación física); los registros se organizan como un grafo, los registros son los nodos y los arcos son los conjuntos.

- Modelo jerárquico: tipo de modelo de red con algunas restricciones; los datos se representan como colecciones de registros y las relaciones entre los datos mediante conjuntos como en el modelo de red, pero cada nodo puede tener un solo padre. Una base de datos jerárquica puede representarse mediante un árbol, los registros son los nodos, también denominados segmentos y los arcos son los conjuntos.
- Modelo relacional: se basa en el concepto matemático relación, que gráficamente se puede representar como una tabla; los datos y las interrelaciones existentes entre los datos se representan mediante relaciones, cada una con un nombre único y con un conjunto de columnas; la base de datos es percibida por el usuario como un conjunto de tablas; esta percepción es sólo a nivel lógico (en los niveles externo y conceptual de la arquitectura de tres niveles), ya que a nivel físico puede estar implementada mediante distintas estructuras de almacenamiento.
- Modelo orientado a objetos[2]: define una base de datos en términos de objetos, sus propiedades y sus operaciones; los objetos con la misma estructura y comportamiento pertenecen a una clase y las clases se organizan en jerarquías o grafos acíclicos; las operaciones de cada clase se especifican en términos de procedimientos predefinidos denominados métodos.

- Número de usuarios (cantidad de usuarios que el gestor puede atender a la vez):
  - Monousuario: sólo atienden a un usuario a la vez y su principal uso se da en los ordenadores personales.
  - Multiusuario: atienden a varios usuarios al mismo tiempo.
- Distribución de los datos (cantidad de servidores en los que está hospedada la base de datos):
  - Centralizados: sus datos se almacenan en un solo servidor.
  - Distribuidos (homogéneos, heterogéneos): la base de datos real y el propio software gestor pueden estar distribuidos en varios sitios conectados por una

[2] Algunos gestores relacionales existentes en el mercado han estado extendiendo sus modelos para incorporar conceptos orientados a objetos, a estos se les conoce como sistemas objeto-relacional.

red. Los sistemas de gestión distribuidos homogéneos utilizan el mismo gestor en múltiples sitios.

- Ámbito de aplicación (objetivo para el cual está diseñado el sistema de gestión):
  - Propósito general: sirven para cualquier tipo de aplicación.
  - Propósito específico: cuando el rendimiento es fundamental, se puede diseñar y construir un gestor de propósito especial para una aplicación específica que generalmente no sirve para otras aplicaciones.

Generalmente los sistemas comerciales actuales están basados en el modelo relacional; los sistemas más antiguos estaban basados en el modelo de red o el jerárquico. Estos dos últimos requieren que el usuario tenga conocimiento de la estructura física de la base de datos a la que se accede, mientras que el relacional proporciona mayor independencia de los datos.

1.1.3.- Funciones de los sistemas de gestión de bases de datos

Los sistemas de gestión de bases de datos deben incluir por lo menos las siguientes funciones: (Hansen, et al., 1999)

- Herramienta para la definición y el control centralizado de los datos (diccionario de datos/directorio (DD/D) o catálogo): donde se almacenan las definiciones de todos los elementos de las bases de datos.
- Mecanismos de seguridad e integridad de los datos: limita el acceso a las bases de datos para el personal no autorizado y por niveles de privilegios, garantizando que existan usuarios que tengan acceso a determinados datos y a otros no.
- Acceso concurrente a los datos para varios usuarios[3]: proporciona mecanismos físicos para que varios usuarios puedan acceder de forma rápida y eficiente a los mismos datos.
- Utilidades para las consultas, manipulación y presentación de peticiones orientadas al usuario: brinda herramientas de definición y manipulación de los datos, posibilitando la interacción del usuario con la base de datos y la asimilación de la información por el usuario de manera sencilla.

[3] No todos los gestores soportan esta característica.

1.1.4.- Ventajas del uso de un sistema de gestión de bases de datos

La utilización de un sistema de gestión de bases de datos garantiza: (ibercom, 2006)

- Facilidades para la manipulación de grandes volúmenes de datos.
- Manejo de las políticas de respaldo adecuadas, garantizan que los cambios de las bases de datos serán siempre consistentes sin importar si hay errores.
- Organización de los datos con un impacto mínimo en el código de los programas.
- Bajos tiempos de desarrollo y aumento de la calidad del sistema desarrollado si son bien explotados, los sistemas de gestión de bases de datos, por los desarrolladores.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

1.1.5.- Desventajas del uso de un sistema de gestión de bases de datos

Independiente de las ventajas, utilizar un sistema de gestión de bases de datos tiene algunas desventajas: (ibercom, 2006)

- Es necesario disponer de una o más personas que administren la base de datos; lo que puede incrementar los costos de operación en una empresa.
- Si se tienen muy pocos datos que son usados por un único usuario por vez y no hay que realizar consultas complejas sobre los datos, es posible que sea mejor usar una hoja de cálculo.
- El tamaño, la complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder correr.
- El costo del hardware adicional, los requisitos de hardware para correr un gestor de bases de datos por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

1.2.- *PostgreSQL*

1.2.1.- Definición de *PostgreSQL*

PostgreSQL es un potente sistema de gestión de bases de datos objeto-relacional (O-RDBMS), multiusuario, centralizado y de propósito general, que está siendo desarrollado

desde 1977 y que está liberado bajo la licencia *Berkeley Software Distribution* (BSD). Es ampliamente considerado como el sistema gestor de bases de datos de código abierto más avanzado del mundo; fue pionero en muchos conceptos que estuvieron disponibles en algunos sistemas de bases de datos comerciales de alto calibre como por ejemplo: control de acceso simultáneo, gestión de transacciones, puntos de seguridad; fue uno de los primeros intentos en implementar un motor de bases de datos relacional. (PostgreSQL Global Development Group, 2009)

1.2.2.- Historia de *PostgreSQL*

El sistema gestor de bases de datos objeto-relacional *PostgreSQL* ha tenido una larga evolución desde su surgimiento, que comienza en la década del 70 con el proyecto *Ingres* de la Universidad de *Berkeley*. (PostgreSQL Global Development Group, 2009)

Michael Stonebraker después de haber trabajado largo tiempo en *Ingres* deja la Universidad en 1982 para comercializar el proyecto, pero decidió volver a la misma en 1985 para trabajar en un nuevo proyecto sobre la experiencia de *Ingres*, dicho proyecto fue llamado *post-ingres* o simplemente *Postgres*. Este tenía como objetivos iniciales proporcionar un mejor soporte para objetos complejos, dar la posibilidad a los usuarios de extender los tipos de datos, operadores y métodos de acceso, además de proporcionar los mecanismos necesarios para crear bases de datos activas. (PostgreSQL Global Development Group, 2009)

Los hitos más importantes en la vida del proyecto *Postgres* fueron en: (PostgreSQL Global Development Group, 2009)

- 1986: comienza la implementación de *Postgres*, además se publicaron varios documentos donde se describían las bases del sistema.
- 1987: surge la primera versión del sistema y fue mostrado en 1988 con su versión utilizable.
- 1989: fue liberada la primera versión del gestor para una pequeña comunidad de usuarios.
- 1990: en respuesta a críticas que se le hicieron al sistema, surge la versión 2 incorporándole un nuevo sistema de reglas.
- 1991: surge la versión 3, en la cual se añadió soporte para múltiples gestores de almacenamiento, se incluyó una serie de mejoras para mayor eficiencia en el

ejecutor de consultas además del sistema de reglas reescrito.

- 1992: *Postgres* se convirtió en el principal gestor de bases de datos para *Sequoia* 2000 (proyecto de computación científica).
- 1993: se duplicó el tamaño de la comunidad de usuarios, la cual demandaba más características.
- 1994: se unieron a la comunidad dos estudiantes de la Universidad para añadir un intérprete de lenguaje SQL (*Structured Query Language*) a *Postgres*, dado que anteriormente contaba con su propio lenguaje de consultas, creando así el sistema al cual nombraron *Postgres*95; se le realizaron muchos cambios internos que mejoraron el rendimiento y la facilidad de mantenimiento del gestor.
- 1996: se decidió cambiar el nombre de *Postgres*95 de tal modo que reflejara la característica del lenguaje SQL y lo terminaron llamando *PostgreSQL*.
- 1997: surge la primera versión formal de *PostgreSQL* 6.0. Desde entonces, una gran cantidad de usuarios, desarrolladores y administradores de motores de bases de datos se unieron al proyecto y comenzaron a incorporar características al potente gestor.
- 2000: surge la versión de *PostgreSQL* 7.0, que fue una de las que demostró el crecimiento continuo del gestor de base de datos. Entre las mejoras más significativas que se le incorporaron están las llaves foráneas y el optimizador de revisión.
- 2005: fue liberada la primera versión de *PostgreSQL* 8.0, que se ejecuta de forma nativa en *Microsoft Windows* como servidor y además es compatible con emisiones basadas en NT de *Windows*, como *Windows* 2000 SP4, *Windows* XP y *Windows* 2003. (PostgreSQL Global Development Group , 2010)
- 2009: se libera la versión 8.4 la cual muestra un enfoque dirigido a la adición de características como por ejemplo la autenticación, el control, la reutilización del espacio y agrega soporte a parte del estándar SQL 2008. (PostgreSQL Global Development Group , 2010)

El proyecto *PostgreSQL* continúa haciendo lanzamientos menores de reparación de errores, todos disponibles bajo la licencia BSD y basados en contribuciones de

proveedores comerciales, programadores de código abierto mayormente y empresas que ofrecen ayuda al Gestor.

1.2.3.- Arquitectura de *PostgreSQL*

PostgreSQL está diseñado con una arquitectura cliente/servidor y usa múltiples procesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

En la figura siguiente se mostrarán, de manera general y simplificada, los componentes más importantes de *PostgreSQL*, que hacen de él una aplicación robusta.

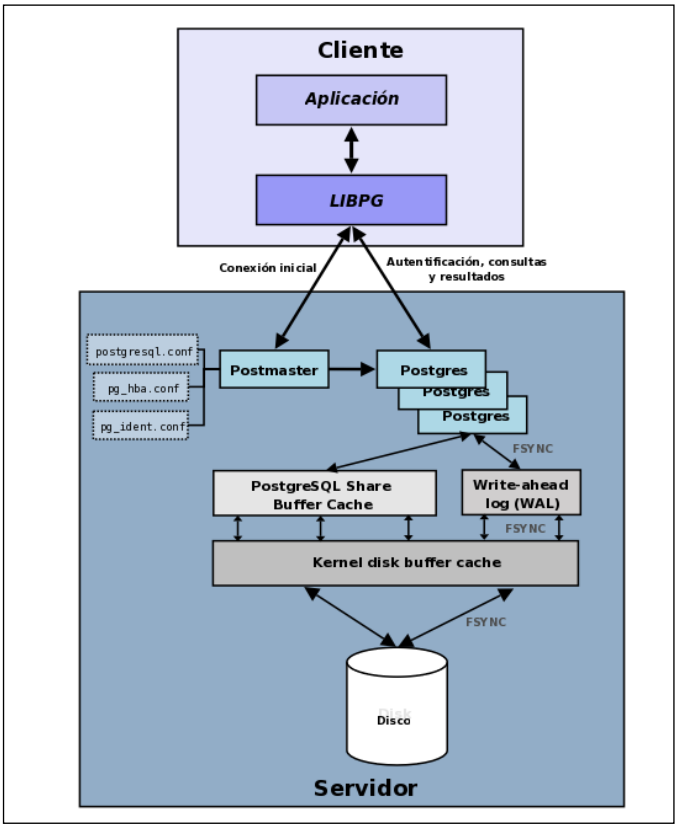


Figura 1.- Arquitectura de *PostgreSQL* (Martínez, 2009)

- **Aplicación cliente:** aplicación que utiliza a *PostgreSQL* como gestor de bases de datos; la conexión puede ocurrir vía TCP/IP o *sockets* locales.
- **Demonio *postmaster*:** proceso principal de *PostgreSQL*, que se encarga de escuchar por un puerto/*socket* por conexiones entrantes de clientes y de crear los procesos hijos.

- **Ficheros de configuración:** los ficheros principales de configuración utilizados por *PostgreSQL* son *postgresql.conf* que se utiliza para la configuración global, *pg_hba.conf* utilizado para la configuración de autenticación de clientes y *pg_ident.conf* útil para el mapa de autenticación usando identificador.
- **Procesos hijos *postgres*:** son los encargados de autenticar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.
- ***PostgreSQL share buffer cache*:** memoria compartida usada por *PostgreSQL* para almacenar datos en la caché.
- ***Write-Ahead Log* (WAL):** componente del sistema encargado de asegurar la integridad de los datos, al registrar los cambios en un registro antes de ser escritos al disco.
- ***Kernel disk buffer cache*:** caché de disco del sistema operativo.
- **Disco:** disco físico donde se almacenan los datos y toda la información necesaria para que *PostgreSQL* funcione.

1.2.4.- Funcionalidades de *PostgreSQL*

PostgreSQL es el gestor de bases de datos de código abierto más avanzado actualmente, ofrece sofisticadas características como: (Medrano, 2009)

- Organiza los datos mediante un modelo objeto-relacional.
- Capaz de manejar procedimientos, rutinas complejas y reglas.
- Soporta *tablespaces*[4], transacciones anidadas, copias de seguridad en línea y soporte para parte de los estándares SQL 92, 99, 2003 y 2008.
- Cuenta con una API sumamente flexible propia para el trabajo con varios lenguajes de programación y procedurales como C, C++, .NET, Bash, Delphi, PL/Java, PL/Perl, PL/Tcl, PL/pgSQL, PL/Ruby, PL/PHP, PL/Python, PL/Scheme y PL/R.
- Ofrece transacciones que permiten el paso entre dos estados consistentes manteniendo la integridad de los datos.

[4] Elemento que se utiliza para el almacenamiento de objetos temporales y para la organización física de los objetos.

- Es altamente extensible, soporta operadores, funciones, métodos de acceso y tipos de datos declarados por el usuario; soporta además sobrecarga de operadores, sobrecarga de procedimientos, vistas materializables, particionamiento de tablas y datos.
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de la información dentro de las bases de datos.
- Ofrece control de acceso simultáneo a través de la gestión de múltiples versiones de un mismo registro (MVCC). (Casanova, et al., 2009)
- Las restricciones y disparadores tienen la función de mantener la integridad y consistencia en las bases de datos.
- Usa una arquitectura cliente/servidor basada en un proceso por usuario; existe un proceso maestro que se ramifica para proporcionar conexiones adicionales por cada cliente que se intenta conectar a *PostgreSQL*.
- Tiene incorporado el mecanismo WAL (*Write-Ahead Logging*), que incrementa la confiabilidad de las bases de datos al registrar los cambios antes de ser escritos al disco; lo que asegura que, en caso de ocurrir un fallo crítico en las bases de datos, exista un registro de transacciones del cual se pueda restaurar. (Herrera, et al., 2009)

1.2.5.- Ventajas de *PostgreSQL*

El gestor de base de datos *PostgreSQL* es robusto y por ende hoy en día es el más usado con respecto a gestores libres existentes como: *SQLite*, *MySQL* en su versión libre, *FireBird*, DB2 *Express*-C y *Apache Derby*, ya que presenta una mayor escalabilidad y rendimiento bajo grandes cargas de trabajo. A continuación se reflejan las diferentes ventajas que este gestor ofrece: (Espinoza, 2005)

- Instalación ilimitada: no hay costo asociado a la licencia del software.
- Soporte: existe una gran comunidad de profesionales y empresas que ofrecen soporte a *PostgreSQL*, de la cual el Grupo Global de Desarrollo de *PostgreSQL* es la principal.
- Estabilidad y confiabilidad legendaria: miles de compañías reportan que *PostgreSQL* nunca ha presentado caídas en varios años de operación de alta actividad.

- Extensible: el código fuente está disponible para todos sin costo.
- Multiplataforma: soporta alrededor de 34 plataformas incluyendo *Linux* y *Unix* en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, *Mac OS X*, *Solaris*, Tru64) y *Windows*.
- Posee herramientas gráficas de diseño y administración de bases de datos: existen varias herramientas gráficas de alta calidad para administrar las bases de datos (*pgAdmin*, *pgAccess*) y para hacer diseño de bases de datos (*Data Architect*).
- Soporta funciones con privilegios por usuario: que pueden definirse para ejecutarse con los derechos del usuario administrador o con los derechos de un usuario previamente definido, además retornan filas donde las salidas pueden tratarse como un conjunto de valores retornados por una consulta.
- Soporta bloques de código: que se ejecutan en el servidor y que pueden ser escritos en diferentes lenguajes de programación con la potencia que presenta cada uno.
- El máximo tamaño de bases de datos es ilimitado: depende del sistema de almacenamiento. (Martínez, 2009)
- El máximo número de filas por tablas es ilimitado. (Ibarra, et al., 2008)

1.2.6.- Desventajas de *PostgreSQL*

Independientemente de las potencialidades de *PostgreSQL*, tiene algunas cuestiones que pueden incidir negativamente en su funcionamiento satisfactorio, las más significativas son: (Ibarra, et al., 2008)

- Consume muchos recursos y sobrecarga con facilidad el sistema, si no es configurado de manera óptima.
- Las filas de las tablas tienen como límite de tamaño 8Kb, se puede ampliar a 32Kb[5] recompilándolo, pero con un costo añadido en el rendimiento.
- La velocidad de respuesta es un poco deficiente al gestionar bases de datos relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar

[5] *PostgreSQL* debe correr sobre un sistema operativo de 64 bits.

bases de datos realmente grandes; lo que depende de la configuración del *postgresql.conf*.

- El máximo tamaño de tabla es de 32Tb.
- El máximo tamaño de fila es de 1.6Tb.
- El máximo tamaño de campo es de 1Gb.
- El máximo número de columnas por tablas es de 250-1600, dependiendo del tipo de columnas.
- El máximo número de índices por tablas es ilimitado[6].

1.3.- *PostgreSQL* 8.4

El Grupo Global de Desarrollo de *PostgreSQL* el 1 de julio del 2009 liberó su versión 8.4. Esta versión cuenta con una gran cantidad de funcionalidades y mejoras que se le incorporaron para la administración, consultas y programación en aras de continuar mejorando el rendimiento del Gestor.

La mayoría de los cambios nuevos o mejorados con los que cuenta *PostgreSQL* 8.4 son herramientas, órdenes de administración y monitoreo. Entre las mejoras más significativas que se le incorporaron están: (Catrin, 2009)

- Restauración de bases de datos en procesos paralelos, que acelera la recuperación de un respaldo hasta 8 veces mayor[7].
- Nuevos métodos del ejecutor para *antijoins* y *semijoins* (EXISTS y NOT EXISTS).
- Privilegios por columna, permiten un control más específico de los datos confidenciales.
- Soporte mejorado para Kerberos 5 (Krb5, protocolo de autenticación de red diseñado para proporciona autenticación fuerte para las aplicaciones cliente/servidor), GSSAPI (acrónimo de *Generic Security Services Application Programming Interface,* API genérica para la autenticación cliente/servidor) y SSPI

[6] El uso de índices debe ser eficiente, debido al hecho de que una gran cantidad de los mismos puede confundir al optimizador del sistema y hacer más lentas las consultas.

[7] La recomendación es tener un número de *jobs* en dependencia de la cantidad de procesadores de la computadora.

(*Security Support Provider Interface,* permite a una aplicación usar varios modelos de seguridad disponibles en una computadora o red sin cambiar la interfaz de seguridad del sistema).

- Reducción de la memoria en el manejo de los disparadores.
- Configuración de lenguajes por bases de datos, lo cual hace a *PostgreSQL* más útil en entornos con múltiples idiomas.
- Migraciones desde 8.3 a 8.4 con muy bajo tiempo de inactividad, gracias al uso de *pg-migrator beta*.
- Nuevas herramientas para el monitoreo de consultas, que le permiten a los administradores mayor información sobre la actividad del sistema.
- Reducción de carga por el *vacuum,* ampliamente reducida gracias al mapa de visibilidad.
- Mejoras en la implementación en los índices *hash*.
- Nueva implementación del mapa de espacio libre (*Free Space Map*) para el uso de espacio físico en disco y no en memoria.
- Nuevas implementaciones de los módulos del *contrib auto_explain, pg_stat_statements, citext, btree_gin* y *pgbench.* (Herrera, 2009)
- Consultas recursivas (*WITH RECURSIVE*), que permiten hacer llamadas recursivas a las subconsultas.
- Inclusión de las CTE (*Common Table Expression*) permiten la declaración de consultas de uso común para usarlas en un determinado orden de operaciones.
- Mejoras en el manejo de los certificados SSL (*Secure Socket Layer*), esto hace que los esquemas de autenticación basados en SSL sean mucho más flexibles y seguros, además de la adición de un nuevo método de autenticación *cert*.
- Inclusión de funciones ventanas (*Windows Functions*), estas funciones permiten realizar operaciones de agregados a un subconjunto de filas, las cuales forman parte del estándar SQL/2008.
- La cláusula LIMIT puede estar basada en una subconsulta o consulta.
- Nuevo control de estructuras con la cláusula CASE.

- Soporte de argumentos variables en funciones.
- Soporte de argumentos por defecto en funciones.

La versión 8.4 hace el análisis de datos mucho más sencillo a través de funcionalidades avanzadas de ANSI SQL 2003, las funciones ventanas, expresiones comunes de tabla y consultas recursivas; estas estructuras de consulta aumentan sustancialmente la expresividad del lenguaje SQL de *PostgreSQL*, permitiendo a los usuarios realizar preguntas interesantes en una sola consulta.

1.3.1.- Problemas arreglados en *PostgreSQL* 8.4.1

Una vez analizadas las mejoras más significativas que se le hicieron al gestor en su versión 8.4, se propone conocer algunos de los cambios realizados recientemente, que dieron paso a la versión menor 8.4.1: (PostgreSQL Global Development Group, 2010)

- Ajustes en la inicialización del encabezado de la página WAL al final de la recuperación del archivo; esto podría llevar al fracaso el proceso WAL en una recuperación de los archivos en subsecuencia.
- Arreglos en los problemas que podrían hacer que expiraran las filas después de un error; este error se trata de un *bit* de estado de la página que no se establece correctamente después de un fallo del servidor.
- Deshabilitar *RESET ROLE* y *RESET SESSION AUTHORIZATION* que inciden en la seguridad de las funciones definidas.
- Arreglos al habilitar y deshabilitar un módulo.
- Creación de funciones ventanas, *PARTITION BY* y *ORDER BY* para los elementos que pueden ser interpretados como expresiones simples.
- Ajustes de errores en la planificación de *semijoins*.
- Ajustes en el manejo de todas las referencias de fila a subconsultas que están dentro de una combinación externa.
- Arreglos en el manejo local de PL/Perl, que antes podía causar cambios en la configuración del servidor local cuando se llama una función de este tipo y dando lugar a la corrupción de los datos.
- Ajustes en las pérdidas de memoria para *array_agg* () en las consultas de *GROUP BY*.

- Se incluye la parte fraccionaria en el resultado *EXTRACT* (*second*) y *EXTRACT* (*milliseconds*) para el tiempo, además el tiempo con las entradas de zona horaria.
- Arreglos en el desbordamiento para el intervalo de x milisegundos cuando x es más de 2 millones de datos de tiempo enteros que están siendo utilizados.
- Mejoras en el rendimiento cuando se están procesando los valores de pruebas, esto es particularmente útil para *PostGIS*[8].
- Soluciones en errores tipográficos que deshabilitaban el *commit_delay*.
- Ajustes en las salidas principales de los mensajes para *postmaster.log* si el servidor es iniciado en modo silencioso.
- Eliminación de las preguntas frecuentes (*Frequently Asked Questions*, FAQs) traducidas.
- Arreglos en *pg_ctl*, para no entrar en un bucle infinito si *postgresql.conf* se encuentra vacío.
- Soluciones en errores de *pg_dump* para actualización en modo binario.

1.4.- Productos empresariales basados en *PostgreSQL*

PostgreSQL es uno de los proyectos de código abierto más grandes y maduros que existe en la actualidad. La comunidad que le da soporte no sólo ha producido un gestor de bases de datos, sino que también es conocido por tener la mejor documentación de código abierto disponible hoy en día, cuenta además con diferentes empresas que apoyan y contribuyen de distintas maneras con el Proyecto.

1.4.1.- *EnterpriseDB*

EnterpriseDB es el proveedor líder de productos de clase empresarial y de los servicios basados en *PostgreSQL*. Esta compañía además de contribuir con el Proyecto, construye y comercializa productos como el *Postgres Plus*, que le permite a las organizaciones sustituir o complementar sus bases de datos comerciales existentes, de un costo muy bajo de propiedad sin sacrificar el rendimiento o las características críticas, además de productos ideales para la operación de aplicaciones de uso intensivo que requieren un

[8] Es un módulo que añade soporte de objetos geográficos a las bases de datos de *PostgreSQL* para su utilización en Sistemas de Información Geográfica.

rendimiento superior, masiva escalabilidad y la compatibilidad con los productos de gestores de bases de datos propietarios. (EnterpriseDB Corporation, 2009)

La compañía *EnterpriseDB* contribuye de muchas formas con la comunidad de *PostgreSQL,* además ofrece diversos servicios y opciones de soporte técnico para ayudar en el desarrollo y despliegue de aplicaciones basadas en *PostgreSQL* los cuales se mencionarán a continuación: (EnterpriseDB Corporation, 2009)

- *Query Optimizer Hints*: es el componente de un sistema de gestión de bases de datos para determinar la forma más eficiente de ejecutar una consulta. En general, al optimizador de consultas no se puede acceder directamente por los usuarios, una vez que las consultas se envían al servidor de bases de datos y son analizadas por el intérprete, pasan por el optimizador de consultas donde se realiza la optimización.
- *DBA Management Server*: permite a los administradores controlar el rendimiento real de las bases de datos y para identificar problemas de configuración para un número ilimitado de bases de datos *Postgres Plus* o *Postgres Plus Advanced Server*.
- *Postgres Plus Advanced Server*: construido sobre las características de *Enterprise Class Packaging* de la fuente de *PostgreSQL* a nivel de base y que además se adicionan a dicha base varios módulos adicionales de código abierto que pueden ser clasificados en cuatro categorías: extensiones, seguridad, lenguajes procedurales y los módulos de contribución. Todas las categorías anteriores se integran en un programa de instalación sencillo e intuitivo, certificado y repetible a través de un extenso proceso de garantía de calidad.
- *GridSQL*: incluye inteligencia para maximizar las consultas en paralelo en varios servidores, ofreciendo un tiempo de respuesta mayor que puede lograrse mediante bases de datos en un solo nodo. La arquitectura de la escala horizontal capaz de acceder a grandes conjuntos de datos es transparente a la aplicación de llamada, se paraleliza cada consulta para utilizar eficientemente todos los recursos disponibles a través de múltiples servidores, manteniendo una visión única para la aplicación de la llamada. (EnterpriseDB Corporation, 2010)

1.4.2.- *CommandPrompt*

La compañía *CommandPrompt* es la más antigua y más grande proveedora de *PostgreSQL* dedicada a brindar soporte en América del Norte. Desde 1997 ha venido desarrollando, apoyando, desplegando y promoviendo el uso del gestor de bases de datos más avanzado del mundo en código abierto. Esta empresa brinda diferentes servicios para *PostgreSQL* que incluyen: (Command Prompt, 2010)

- Consultoría.
- Desarrollo personalizado de *PostgreSQL*.
- Características de implementación para *PostgreSQL*.
- Servicios de administración y soporte proactivo.
- Capacitación.
- Soporte las 24 horas, los 7 días de la semana.

1.4.3.- *2ndQuadrant*

2ndQuadrant ofrece servicios, capacitación y apoyo, ayudando a obtener lo mejor en las bases de datos *PostgreSQL*. También ofrece conocimientos técnicos de clase mundial para empresas grandes y en crecimiento de todo el mundo, con experiencia centrada en *PostgreSQL* para el procesamiento de transacciones en línea (OLTP, *OnLine Transaction Processing*), Web y almacenes de datos. (2ndQuadrant, 2010)

2ndQuadrant ofrece una amplia gama de cursos de capacitación, resaltando la cantidad de paquetes de servicios más solicitados con regularidad, para los cuales se han desarrollado métodos y herramientas internas para asegurar que se entreguen de forma rentable, utilizando sus conocimientos y experiencias, los fundamentales son:

- Optimización del rendimiento de *PostgreSQL*.
- Replicación y alta disponibilidad de *PostgreSQL*.
- Escalabilidad en la base de datos.
- Bases de datos/migración de aplicaciones.
- Actualizaciones en línea.

Dentro de los servicios que brinda esta empresa se encuentra además el proyecto de desarrollo de software y consultoría que se utiliza para diseñar y desarrollar software para

el núcleo *PostgreSQL*, regularmente se comprometen en comisiones para desarrollar complementos, herramientas y ampliar las funciones o mejorar el rendimiento para *PostgreSQL*.

1.4.4.- *Skype*

Skype es una compañía global líder de comunicaciones por Internet. Produce un software que permite conversaciones por todo el mundo, que millones de personas y empresas utilizan para hacer llamadas de video y voz, enviar mensajes instantáneos y compartir archivos con otros usuarios. (Skype, 2008)

Es el responsable del 8% de las llamadas internacionales que se realizan por minutos y, con usuarios que realizan 3.1 mil millones de minutos en llamadas a teléfonos fijos y móviles en el tercer trimestre del 2009.

Skype es una solución fácil de implementar que administra todas las necesidades de comunicaciones de las personas, los empleados trabajan con más productividad, además los clientes pueden comunicarse fácilmente con la empresa que deseen.

Durante el desarrollo de la infraestructura del *backend* de *Skype*, se han utilizado bases de datos *PostgreSQL* de varias maneras. Para optimizar las necesidades de negocios de la empresa han creado varios servicios los cuales se mencionarán a continuación: (Skype Limited, 2008)

- *PL/Proxy*: es un lenguaje de programación integrado para *PostgreSQL*, siendo una solución para el particionamiento horizontal de datos entre servidores.
- *SkyTools*: es un paquete de herramientas utilizadas para garantizar la alta disponibilidad (*PgBouncer*), redundancia de datos (*Londistes*), optimización de las peticiones a los servidores *PostgreSQL* (PgQ) y manejo del archivado de registros WAL (*WalMng*).

1.4.5.- *PgExperts*

PostgreSQL Experts es otra de las compañías en Norteamérica dedicada a los servicios profesionales para *PostgreSQL*. El equipo de *PostgreSQL* y expertos en tecnología de código abierto brindan soporte a todas las bases de datos y sistemas necesarios para desarrollar nuevas aplicaciones, acelerar las ya existentes, refactorizar su infraestructura y resolver los problemas de las bases de datos. (PostgreSQL Experts, 2009)

Adicionalmente los expertos de esta empresa se extienden más allá de las bases de datos, tienen habilidades para construir o rediseñar las aplicaciones de bases de datos completas, hacerlas más escalables, mantenibles y seguras. Los servicios específicos que ofrece esta compañía para *PostgreSQL* son:

- Implementación de nuevas características en *PostgreSQL*.
- *Tuning*, optimización y resolución de errores.
- Servicios de alta disponibilidad e instalación de mecanismos de replicación.
- Diseño personalizado de la arquitectura de las bases de datos.
- Implementación de almacenes de datos.
- Instalaciones, actualizaciones y migraciones.
- Servicios de administración remota de bases de datos.

Los servicios relacionados son:

- Aplicaciones Web desarrolladas en *Perl, Python, Ruby y Java*.
- Salvas para la automatización y administración de la gestión de datos.
- Exportación de aplicaciones de alto o bajo costo de realización para *PostgreSQL*.
- Adquisición de datos a distancia, almacenamiento y análisis para aplicaciones científicas e industriales.

1.4.6.- *Continuent*

Principal proveedor de la disponibilidad de los datos y el rendimiento de las soluciones de escalabilidad de la mayoría de las bases de datos, ofrece una alta disponibilidad en ellas e incrementa el rendimiento utilizando el hardware de los productos básicos y las bases de datos. (Continuent, 2010)

Ofrece además soporte, capacitación y servicios de consultoría a clientes en todo el mundo. Esta empresa tiene una larga historia de apoyo para *PostgreSQL,* tanto por el suministro de productos a los usuarios, así como el apoyo a la comunidad de *PostgreSQL* como un todo. Estas son algunas de las ofertas de productos claves para *PostgreSQL* de esta empresa:

- Simplifica la complejidad estableciendo la replicación.
- Proceso de copias de seguridad automatizadas.

- Recuperación de desastres a través de la red de área extensa (*Wide Area Network*, WAN).
- Replicación de *Oracle* a *PostgreSQL*.
- Alta disponibilidad con conmutación por error de inmediato en la pérdida de la base de datos.
- Extensión de aplicaciones de lectura intensa con un mínimo o sin cambios en las aplicaciones.

1.5.- Módulos del *contrib* para *PostgreSQL*

Los módulos de contribución son paquetes que se pueden encontrar o colocar en el directorio *contrib* de la distribución de *PostgreSQL*. Esto incluye herramientas para migración, utilidades de análisis y características de tipo *plugins* las cuales no son parte del núcleo de *PostgreSQL*, sobre todo porque son realizadas para un propósito específico o porque son muy experimentales para ser parte del árbol del código fuente principal. Esto no se opone a su utilidad, aunque para utilizarlos hay que configurarlos ya que cuando se instala desde el código fuente ellos no lo hacen de manera automática. (EnterpriseDB Corporation, 2009)

Producto del alto impacto y aceptación que ha tenido el *Postgres Plus* en el sector empresarial a nivel mundial, se decide hacer un estudio en profundidad de los módulos que han sido añadidos al *contrib* del Gestor en él y que serán, los módulos a considerar para el desarrollo de la guía propuesta por esta investigación.

1.5.1.- Descripción de los módulos de análisis de datos

Los módulos de análisis de datos existentes a ser considerados en la propuesta de solución son: (PostgreSQL Global Development Group, 2010)

- *cube*: implementa un tipo de datos para la representación de cubos multidimensionales.
- *fuzzystrmatch*: proporciona varias funciones para determinar las similitudes y la distancia entre las cadenas.
- PL/R: lenguaje procedural para *PostgreSQL* que permite escribir funciones *y* contar con las funciones agregadas del lenguaje de análisis estadísticos y gráficos R. (bostongis.com, 2010)

1.5.2.- Descripción de los módulos de monitoreo

Los módulos de monitoreo existentes a ser considerados en la propuesta de solución son: (PostgreSQL Global Development Group, 2010)

- *pg_buffercache*: proporciona un medio para examinar lo que está sucediendo en el búfer de caché en tiempo real.
- *pg_freespacemap*: proporciona un medio para examinar el mapa del espacio libre (FSM). Además proporciona una función llamada *pg_freespace* o dos funciones sobrecargadas y las funciones de mostrar el valor registrado en el mapa de espacio libre para una página dada, o para todas las páginas en la relación.
- *pg_stat_statements*: proporciona un medio para el seguimiento de las estadísticas de la ejecución de todas las sentencias SQL en el servidor.

1.5.3.- Descripción de los módulos de administración

Los módulos de administración existentes a ser considerados en la propuesta de solución son:(PostgreSQL Global Development Group, 2010)

- *adminpack*: proporciona una serie de funciones de apoyo utilizadas por la administración y herramientas de gestión para funcionalidades adicionales, como la gestión remota de los archivos de registro del servidor.
- *oid2name*: es una utilidad que ayuda a los administradores a examinar la estructura de archivos utilizados por *PostgreSQL*.
- *pgbench*: es un programa para ejecutar las pruebas de referencia sobre el *PostgreSQL*.
- *lo*: proporciona apoyo para la gestión de objetos grandes.
- *pg_standby*: apoya la creación de un servidor suplente de bases de datos. Está diseñado para ser un programa de producción listo, así como una plantilla personalizable en caso de necesitar modificaciones específicas.
- *pgrowlocks*: proporciona una función para mostrar información del bloqueo de registros de una tabla especificada.
- *pgstattuple*: proporciona diversas funciones para obtener estadísticas a nivel de tupla.
- *pageinspect*: proporciona funciones que le permiten inspeccionar el contenido de

las páginas de bases de datos en un nivel bajo, que es útil para propósitos de depuración.

Todas estas funciones pueden ser utilizadas únicamente por super-usuarios.

1.5.4.- Descripción de los módulos de desarrollo

Los módulos de desarrollo existentes a ser considerados en la propuesta de solución son: (PostgreSQL Global Development Group, 2010)

- *auto_explain*: proporciona un medio para registrar automáticamente los planes de ejecución de sentencias lentas. Esto es especialmente útil para la optimización de las consultas en las aplicaciones de gran tamaño.
- *hstore*: implementa un tipo de datos para almacenar conjuntos de pares (clave, valor), dentro de un único campo de datos *PostgreSQL*; útil en diversos escenarios tales como las filas con muchos atributos que rara vez son examinados o en datos semi-estructurados.
- *ltree*: implementa un tipo de datos *ltree* para la representación de las etiquetas de los datos almacenados en un árbol de estructura jerárquica.
- *pg_trgm*: proporciona funciones y operadores para determinar las similitudes de texto basado en la coincidencia trigrama[9], así como clases de operadores de índices que soportan la búsqueda rápida para cadenas similares.
- *seg*: implementa un tipo de datos para la representación de segmentos de línea, o intervalos de punto flotante; puede representar la incertidumbre en las variables de intervalo.
- *tablefunc*: incluye diversas funciones que devuelven varias filas.
- *uuid-ossp*: proporciona funciones para generar identificadores únicos universales (UUID) usando uno de varios algoritmos estándares.
- *citext*: proporciona un tipo de cadena de caracteres en mayúsculas y minúsculas.
- *dict_xsyn* (diccionario de sinónimos extendido): es un ejemplo de un diccionario plantilla añadido para la búsqueda de texto completo, este tipo de diccionario sustituye palabras con los grupos de sus sinónimos, haciendo posible la búsqueda

[9] Conjunto de 3 letras.

de una palabra con alguno de sus sinónimos.

- *spi*: proporciona varios ejemplos viables para usar SPI y disparadores.
- *test_parser*: es un ejemplo de un analizador de encargo para la búsqueda de texto completo.
- *btree_gin*: proporciona una muestra de una clase de operador Gin[10] que implementa el comportamiento equivalente de *B-Tree* (que son estructuras de árbol que se encuentran comúnmente en las implementaciones de bases de datos y sistemas de archivos) para diferentes tipos de datos.
- *btree_gist:* ofrece una muestra de clase de operador Gist[11] que implementa *B-tree* equivalente al comportamiento para diferentes tipos de datos.
- *earthdistance*: ofrece dos enfoques diferentes para calcular las distancias de gran círculo en la superficie de la Tierra.
- *isn*: proporciona un tipo de datos *isn* para el producto internacional de numeración de diferentes estándares.

1.5.5.- Descripción de los módulos de seguridad

Los módulos de seguridad existentes a ser considerados en la propuesta de solución son: (PostgreSQL Global Development Group, 2010)

- *chkpass*: implementa un *chkpass*, que es un tipo de datos que está diseñado para almacenar contraseñas encriptadas, cada contraseña se convierte automáticamente en forma cifrada a la entrada y siempre se almacena de esa manera.
- *pgcrypto*: implementa funciones criptográficas para *PostgreSQL*.
- *veil*: es una inclusión de seguridad de datos para *PostgreSQL*.

[10] Índice GIN (Índice Invertido Generalizado), es una estructura de índices que almacena un conjunto de pares (clave, lista de desplazamiento).

[11] Índice GiST (Árbol de Búsqueda Generalizado), permite el desarrollo de tipos de datos personalizados con los métodos de acceso apropiado.

1.6.- Personalización de *PostgreSQL*

Según lo visto hasta el momento:

- *PostgreSQL* es un sistema de gestión de bases de datos objeto-relacional, multiusuario, centralizado y de propósito general.
- *PostgreSQL* es el sistema de gestión de bases de datos libre más potente, ofrece características avanzadas que solamente poseen sistemas propietarios de alto calibre como *Oracle* y SQL *Server*.
- El hecho de que sea de propósito general es bueno porque sirve para todas las aplicaciones, pero hay aplicaciones específicas que necesitan funcionalidades especiales que el Gestor no cubre y que deben ser añadidas vía *plugins* o módulos.
- Muchas empresas desarrollan productos tomando como base al Gestor, dándole mayor potencialidad y flexibilidad.
- Los productos desarrollados hasta el momento no satisfacen las necesidades del Centro de Tecnologías de Gestión de Datos en cuanto a análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos dentro del núcleo de *PostgreSQL* 8.4.

1.6.1.- Ventajas

La personalización del gestor con estos paquetes tendrían las siguientes ventajas:

- El Centro de Tecnologías de Gestión de Datos contará con un gestor personalizado, que tendrá funcionalidades adicionales para el análisis de datos, monitoreo, administración, desarrollo y seguridad en las bases de datos dentro del núcleo de *PostgreSQL* 8.4.
- El Centro de Tecnologías de Gestión de Datos contará con documentación asequible y fácil de comprender con las funcionalidades de los módulos.
- El Centro de Tecnologías de Gestión de Datos contará con guías para la selección de qué módulos instalar según las necesidades de los proyectos.
- El Centro de Tecnologías de Gestión de Datos contará con documentación asequible y fácil de comprender para la habilitación, configuración y eliminación de dichos módulos.

1.7.- Compiladores

Para realizar la compilación de los módulos de análisis de datos en el *PostgreSQL* 8.4 es necesario seleccionar la herramienta de compilación a utilizar. En los subepígrafes siguientes se describirán las soluciones más empleadas en el mundo y se determinará cuál utilizar.

1.7.1.- *Clang*

Clang es una interfaz para el compilador de C, C++ y otros lenguajes de programación *Objective-C*. Utiliza el *Low-Level Virtual Machine* (LLVM) como su *backend*. Fue cuidadosamente diseñado y construido para sentar las bases de una nueva generación de C/C++/*Objective*-C. Fue creado en la Universidad de *Illinois* por *Chris Lattner* y sus contribuidores bajo la licencia de NCSA *Open Source License*. (LinuxLinks.com, 2009)

Los operadores de *Clang* y las características del lenguaje son intencionalmente diseñados para ser compatibles con el compilador GNU *Compiler Collection* (GCC) como sea razonablemente posible, facilitando la migración de GCC a *Clang*. Algunas de las características que incluye son: (LinuxLinks.com, 2009)

- Realiza una rápida compilación y utiliza poca memoria.
- Diagnóstico expresivo, haciendo el diagnóstico de error y mensajes de advertencia generados por el compilador, lo más útil posible.
- Compatibilidad con GCC.
- Un analizador unificado de C, *Objective* C, C++, y el *Objetive* de C++.
- Conformidad con C/C++/*Objective*-C y sus variantes.

1.7.2.- *G95*

El compilador *G95* es libre, portátil, de código abierto en *Fortran* 95. Fue creado bajo la licencia GNU *General Public License* (GPL), teniendo como desarrolladores a *Steven Bosscher, Paul Brook, Arnaud Desitter, Katherine Holcomb, Niels Kristian Bech Jensen, Steven G. Johnson* entre otros. (LinuxLinks.com, 2009)

Implementa el estándar de *Fortran* 95, parte del estándar de *Fortran* 2003 y algunas viejas y nuevas extensiones, incluye características de las propuestas para el estándar *Fortran* 2008 como *Co-array Fortran*. Algunas de las características que incluye son: (LinuxLinks.com, 2009)

- Funcionamiento de los programas compilados, pueden ser modificados por una larga lista de variables de entorno, documentado en el programa compilado.
- Opción por comas en abrir, leer y escribir para denotar el punto decimal.
- Los corchetes [ ] pueden ser utilizados para constructores de matriz.
- Declaración de importación utilizada en un organismo de interfaz para permitir el acceso a las entidades de la unidad de acogida de alcance.
- MIN () y MAX () para los caracteres, así como tipos numéricos.

1.7.3.- *Low-Level Virtual Machine*

El *Low-Level Virtual Machine* (LLVM) es un compilador de infraestructura, una colección de las bibliotecas y herramientas que facilitan la construcción de compiladores, optimizadores, programas *Just-In* generadores de código de tiempo. Fue creado bajo la licencia de *Open Source License* (NCSA) en la Universidad de *Illinois*. (LinuxLinks.com, 2009)

Los puntos fuertes de la infraestructura LLVM son su diseño extremadamente simple (lo que hace que sea fácil de entender y usar), la fuente de la independencia de lenguaje, potente optimizador de nivel medio, soporte automatizado de depuración del compilador, la extensibilidad y su estabilidad y fiabilidad. LLVM se utiliza actualmente para albergar una gran variedad de proyectos de investigación académica y proyectos comerciales. LLVM puede generar código para *x86*, *sparcv9*, *PowerPC*, o se puede generar código C. Alguna de las características que incluye: (LinuxLinks.com, 2009)

- Optimizador agresivo, escalar, interprocedural y algunas optimizaciones de ciclo simple.
- Completa gama de optimizaciones escalares.
- *Just-In-Time* (JIT), sistema de generación de código, que actualmente soporta *x86*, *x86-64*, *PowerPC* y *PowerPC-64*.
- Marco de prueba con un número de códigos de referencia y aplicaciones.
- API y las herramientas de depuración para simplificar el desarrollo rápido de los componentes LLVM.
- Soporta un modelo de compilación de larga vida, incluyendo enlaces de tiempo, tiempo de instalación, tiempo de ejecución y optimización fuera de servicio.

1.7.4.- *GNU Compiler Collection*

GNU *Compiler Collection* (GCC) es la colección de compiladores de GNU. Creado bajo la licencia de GNU GPL y GNU LPGL. Es desarrollado por el proyecto GNU. La colección de compiladores de GNU es una completa herramienta compilador ANSI C, con soporte para K & R C, así como C++, Objective-C, Java, Objective-C++, Ada y Fortran. (LinuxLinks.com, 2009)

GCC proporciona varios niveles de error de código fuente chequeado tradicionalmente por otras herramientas, genera información de depuración y puede realizar diferentes optimizaciones del código objeto resultante. Es un compilador portable, que funciona en la mayoría de las plataformas disponibles hoy en día y puede producir una salida para muchos tipos de procesadores. (LinuxLinks.com, 2009)

Además de los procesadores utilizados en ordenadores personales, también soporta micro controladores, procesador digital de señal y CPU de 64 bits. GCC es el compilador más utilizado en el desarrollo de código abierto. GCC se usa para representar el compilador C de GNU, pero dado que este es compatible con varios lenguajes aparte de C, ahora es una colección de compiladores de GNU. Algunas de las características que incluye son: (LinuxLinks.com, 2009)

- Compilador nativo.
- Un diseño modular que permite el apoyo a nuevos lenguajes y arquitecturas que se le añadirán.
- Se puede compilar cualquier programa y producir archivos ejecutables para un sistema diferente del utilizado por el propio GCC.
- Interfaces para múltiples lenguajes, para analizar diferentes lenguajes, incluyendo C (GCC), C++ (g++), Java (gcj), Ada (GNAT), Objective-C (gobjc), Objective-C++ (gobjc++), Fortran (gfortran), Modula-2, Modula-3, Pascal (GPC), PL/I, D (GDC), Mercurio y VHDL (ghdl).

1.7.5.- Selección del compilador a utilizar

Producto de la gran cantidad de compiladores existentes, se hace necesario escoger cuál utilizar para la propuesta de solución. En la siguiente tabla se hace una comparación de los compiladores en base a las características mencionadas anteriormente y en base a un conjunto de indicadores deseables a tener el compilador seleccionado.

Tabla 1.- Selección del compilador a utilizar para la confección del paquete

Indicadores	Compiladores			
	Clang	***G95***	***LLVM***	***GCC***
Interfaces para múltiples lenguajes	x			x
Producción de archivos ejecutables				x
Compilador nativo de Linux				x
Facilidad de uso				x
Optimización del código objeto resultante			x	x
Multiplataforma				x
Realiza compilación rápida	x			
Usa poca memoria	x			
Licencia	NCSA	GPL	NCSA	GPL & LGPL

Como se observa en la tabla anterior, el compilador GCC es el más completo y por tanto es la opción que se tomará para la compilación de los módulos de análisis de datos para el *PostgreSQL* 8.4 de la presente investigación.

Conclusiones

En el capítulo se mostraron los resultados del estudio realizado. Se demostró a partir de sus características y funcionalidades por qué *PostgreSQL* es el sistema de gestión de bases de datos de código abierto más avanzado del mundo. Mediante el estudio de las empresas líderes que brindan servicios y productos basados en el Gestor, se seleccionaron 32 módulos a ser incluidos en su directorio *contrib*, 3 de análisis de datos, 3 de monitoreo, 8 de administración, 15 de desarrollo y 3 de seguridad. Para la compilación del paquete se escogió GNU *Compiler Collection* por cumplir con la mayoría de los

indicadores definidos en base a las necesidades para la compilación. Como resultado de la propuesta el Centro contará con una guía para la personalización de *PostgreSQL* con funcionalidades agregadas para el análisis de datos, el monitoreo, la administración, el desarrollo y la seguridad en las bases de datos.

Capítulo 2

Descripción de la solución propuesta

Introducción

En el capítulo se describirá la Guía para la personalización de *PostgreSQL* 8.4, para lo que se definirá el procedimiento para la instalación y configuración del Gestor desde el código fuente, se hará una descripción detallada de los módulos de análisis de datos que podrán ser encontrados en el directorio *contrib* de las fuentes del Gestor, se definirá el procedimiento para la compilación de los módulos, se plasmarán las recomendaciones y buenas prácticas para habilitarlos y se definirá el procedimiento para su eliminación.

2.1.- Estructura de la Guía para la personalización de PostgreSQL 8.4

La Guía tendrá la siguiente estructura:

- Prefacio: donde se explica a quién está dirigida la Guía, las convenciones que se emplearán en ella, las secciones que la conformarán y los requerimientos que se deben cumplir para poder ejecutarla de manera satisfactoria.
- Descripción del sistema de gestión de bases de datos *PostgreSQL*: donde se explican las características del Gestor, sus principales funcionalidades y las nuevas características incorporadas a su versión 8.4.
- Procedimiento para la instalación y configuración del paquete *PostgreSQL* 8.4.1 desde el código fuente: donde se explican detalladamente los pasos para instalar y configurar al Gestor desde su código fuente.
- Descripción detallada de los módulos del *contrib*: donde se detallan los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad contemplados en la Propuesta.
- Procedimiento para la compilación de los módulos del *contrib* de *PostgreSQL* 8.4: donde se explican detalladamente los pasos genéricos para compilar los módulos en el Gestor.

- Procedimiento para la compilación del módulo PL/R: donde se explican detalladamente los pasos para compilar específicamente PL/R en el Gestor.
- Recomendaciones y buenas prácticas para habilitar cada uno de los módulos del *contrib*: donde se explican la gestión de dependencias, las buenas prácticas para la organización de los paquetes y el desarrollo de scripts personalizados para la compilación del Gestor.
- Procedimiento para la eliminación de los módulos una vez instalados: donde se explican detalladamente los pasos para eliminar los módulos de la instalación del Gestor.

2.2.- Procedimiento para la instalación y configuración del paquete *PostgreSQL* 8.4.1 desde el código fuente

Los pasos para instalar *PostgreSQL* 8.4.1 desde el código fuente son los siguientes:

Paso 1.- Instalar make, tar y gzip para satisfacer los requerimientos de los paquetes (generalmente éstos son instalados por defecto cuando se instala el sistema operativo, pero para estar seguros se ejecuta este paso); para ello se utilizará el gestor de paquetes apt con el comando siguiente:

```
apt-get install make tar gzip
```

Paso 2.- Instalar la herramienta build-essential para satisfacer el requerimiento del compilador; puede hacerse con el comando siguiente:

```
apt-get install build-essential
```

Paso 3.- Instalar las dependencias que se necesitan durante el proceso de instalación del Gestor.

```
apt-get install libncurses5-dev libreadline5-dev zlib1g-dev
```

Paso 4.- Copiar el código fuente del Gestor en el directorio usr/local/src.

```
cp -r [ubicación_paquete] usr/local/src
```

Paso 5.- Ubicar el directorio desde donde se ejecutarán los comandos del terminal en src, que será donde se instalará el PostgreSQL; para ello se debe utilizar el comando siguiente:

```
cd /usr/local/src/
```

Paso 6.- Descompactar el paquete PostgreSQL 8.4.1, puede hacerse con el comando siguiente:

tar xvzf postgresql-8.4.1.tar.bz2 -C /usr/local/src/

Paso 7.- Ubicar el directorio desde donde se ejecutarán los comandos del terminal en el paquete PostgreSQL descompactado en el paso anterior PostgreSQL-8.4.1; para ello puede emplearse el comando:

cd /usr/local/src/postgresql-8.4.1/

Paso 8.- Proceder a la instalación del gestor, para ello escribir los comandos siguientes:

. /configure --prefix=/usr/local/pgsql --without-pam --without-ldap --without-krb5 --enable-thread-safety

make

make install

make clean

Paso 9.- Crear usuario del sistema, a través del siguiente comando:

adduser --system --quiet --home /var/lib/pgsql --shell /bin/bash --group --gecos "PostgreSQL administrator" postgres

Paso 10.- Cambiar el usuario actual al usuario del sistema creado en el paso anterior, para ello puede escribir el siguiente comando:

su - postgres

Paso 11.- Crear el clúster de bases de datos, a través del siguiente comando:

/usr/local/pgsql/bin/initdb -D /var/lib/pgsql/data

Paso 12.- Crear el directorio para los logs, necesario porque en caso de no hacerlo se guardarían los logs del Gestor dentro de la carpeta donde está instalado, ocupando mucho espacio. Para ello se deben ejecutar los siguientes comandos:

exit

mkdir /var/log/pgsql/

chown postgres.postgres /var/log/pgsql

Paso 13.- Crear script de inicio con el siguiente contenido:

```
cat > /etc/init.d/postgresql << _EOF_
#!/bin/sh
        case "\$1" in
                start)
                        su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D
                        /var/lib/pgsql/data -l /var/log/pgsql/logfile.log start"
                        ;;
                stop)
                        su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D
                        /var/lib/pgsql/data -l /var/log/pgsql/logfile.log stop"
                        ;;
                restart)
                        su - postgres -c "/usr/local/pgsql/bin/pg_ctl -D
                        /var/lib/pgsql/data -l /var/log/pgsql/logfile.log restart"
                        ;;
                *)
                echo "Usage: \$0 {start|stop|restart}"
                ;;
        esac
        exit 0
_EOF_
```

Paso 14.- Iniciar el sistema, para ellos se pueden emplear los comandos siguientes:

```
chmod 755 /etc/init.d/PostgreSQL
update-rc.d PostgreSQL defaults 99 01
```

Paso 15.- Iniciar PostgreSQL, para ello se pueden emplear los comandos siguientes:

```
/etc/init.d/PostgreSQL start
```

Paso 16.- Editar el fichero /etc/profile, con los siguientes comandos:

nano /etc/profile

Añadir la línea al final del fichero antes de la línea export PATH:

PATH=/usr/local/pgsql/bin:$PATH

2.3.- Descripción detallada de los módulos de análisis de datos

En esta sección sólo serán descritos los módulos de análisis de datos contemplados en la propuesta, los demás están descritos en la Guía para la personalización de *PostgreSQL* 8.4 resultado de la investigación.

Para la descripción de los módulos se utilizará una plantilla definida con los elementos que, a partir del estudio realizado, son difíciles de encontrar y son necesarios para la selección de qué módulo utilizar según las necesidades del proyecto. Ver Anexo 1.

2.3.1.- Descripción de los módulos de análisis de datos

Tabla 2.- Descripción del módulo *cube* de análisis de datos (EnterpriseDB Corporation, 2010), (PostgreSQL CVSweb, 2010), (PostgreSQL Global Development Group, 2010)

Elemento	Descripción
Descripción	Implementa un tipo de datos *cube* para la representación de cubos multidimensionales. A continuación se muestran las representaciones externas válidas para el tipo de datos *cube*: - x y (x): punto de dimensiones o longitud cero de una dimensión de intervalo. - x1, x2,..., xn y (x1, x2,..., xn): punto en el espacio dimensional n, representado internamente como un cubo de volumen cero. - (x), (y) y [(x), (y)]: dimensión de intervalos iniciada en x y terminada en y, o viceversa. - (x1,..., xn), (y1,..., yn) y [(x1,..., xn), (y1,..., yn)]: cubo con n dimensiones representado por un par de esquinas diagonalmente opuestas.

	Los valores son almacenados internamente como puntos de números flotantes de 64 bits (los números con más de 16 dígitos significativos, serán truncados).
Fecha de creación	11-12-2000
Lugar de origen	Grupo Global de Desarrollo de *PostgreSQL*
Desarrolladores	*Gene Selkov*, ha hecho contribuciones sobresalientes a los campos de análisis matemático/computacional del metabolismo celular, la cronobiología, la bioinformática y la ingeniería metabólica, además contribuyó con *PostgreSQL* añadiéndole en el *contrib* el módulo *cube*.
Lenguaje de desarrollo	C
Fecha de última versión	v 1.37, 11-06-2009
Soporte	Grupo Global de Desarrollo de *PostgreSQL*
Grado de generalización	Alto
Comandos para su instalación[12]	Para instalar el módulo desde la distribución de la fuente del Gestor se debe realizar lo siguiente: En la fuente principal de *PostgreSQL* existe una carpeta llamada *contrib*, dentro de esta carpeta se encuentran otras con el nombre de cada módulo, para instalarlo debe posicionarse en el terminal en la ubicación del módulo y escribir los comandos siguientes: make make install Muchos de los módulos cuentan con pruebas de

[12] Los comandos y ubicación son los mismos para la mayoría de los módulos, de ahora en adelante se prescindirá de estas filas en el resto de las descripciones, a no ser que sean diferentes.

	regresión, las cuales puede ser ejecutada con: make installcheck
Ubicación	Directorio *contrib* del paquete de instalación de *PostgreSQL* 8.4.1.

Tabla 3.- Descripción del módulo *fuzzystrmatch* de análisis de datos (EnterpriseDB Corporation, 2010), (PostgreSQL CVSweb, 2010), (PostgreSQL Global Development Group, 2010)

Elemento	Descripción
Descripción	Proporciona varias funciones para determinar las similitudes y la distancia entre las cadenas, brinda apoyo para *levenshtein*/*soundex*/*metaphone.* - *Levenshtein:* calcula la distancia *levenshtein* entre dos cadenas. *levenshtein(text source, text target, int ins_cost, int del_cost, int sub_cost) returns int* *levenshtein(text source, text target) returns int* - *Soundex*: método para obtener los nombres que coincidan y convertirlos en un mismo código. El módulo *fuzzystrmatch* proporciona dos funciones para trabajar con los códigos de *Soundex*: *soundex(text) returns text* *difference(text, text) returns int* - *Metaphone:* esta función, como *soundex*, se basa en la idea de construir un código de representación de una cadena de entrada; dos cadenas de la serie se considerarán similares si tienen los mismos códigos.
Fecha de creación	27-02-2006

Lugar de origen	Grupo Global de Desarrollo de *PostgreSQL*
Desarrolladores	*Bruce Momjian*, se unió al proyecto en el año 1996, proporcionó junto a otras personas el primer servidor de desarrollo no universitario para el esfuerzo de desarrollo de código abierto y comenzó a trabajar para estabilizar el código de *Postgres*95. *Joe Conway*, ha participado con *PostgreSQL* como contribuyente desde el 2001. Su trabajo en el Grupo es la implementación de funciones de tablas, mejoras en tipos de datos *array* y *byte*, librerías del *contrib*, además de otros ajustes y características realizadas para el Gestor.
Lenguaje de desarrollo	C
Fecha de última versión	v 1.32, 02-01-2010
Soporte	Grupo Global de Desarrollo de *PostgreSQL*
Grado de generalización	Alto

Tabla 4.- Descripción de PL/R de análisis de datos (bostongis.com, 2010)

Elemento	Descripción
Descripción	Lenguaje procedural para *PostgreSQL* que permite escribir funciones *y* contar con las funciones agregadas del lenguaje de análisis estadísticos y gráficos R.
Fecha de creación	12-06-2007
Lugar de origen	Grupo Global de Desarrollo de *PostgreSQL*
Desarrolladores	*Joe Conway*[13].

[13] Ver en la Tabla 3 su breve reseña.

Lenguaje de desarrollo	C
Fecha de última versión	v plr-8.3.0.9 ,01-02-2010
Soporte	Grupo Global de Desarrollo de *PostgreSQL*
Grado de generalización	Alto
Ubicación	http://www.joeconway.com/plr/

2.4.- Procedimiento para la compilación de los módulos del contrib de PostgreSQL 8.4

Antes de compilar cualquier módulo es necesario realizar los pasos 1 y 2, que serán hechos sólo una vez aun cuando se quieran compilar tantos módulos como se deseen e incluso cuando dicha compilación no sea realizada en el mismo momento.

A continuación se muestran los pasos para la instalación:

Paso 1.- Compilar el directorio port para que sean incluidas librerías necesarias para la ejecución de los módulos, para ello se pueden emplear los comandos siguientes:

```
cd /usr/local/src/postgresql-8.4.1/src/port

make

make install
```

Paso 2.- Compilar los directorios interfaces para que sean incluidas librerías necesarias para la ejecución de los módulos, para ello se pueden emplear los comandos siguientes:

```
cd /usr/local/src/postgresql-8.4.1/src/interfaces

make

make install
```

Los pasos para la compilación de los módulos específicos son los siguientes:

Paso 1.- Ubicar el directorio desde donde se ejecutarán los comandos del terminal en el módulo [nombre_módulo] que se desee compilar, para ello se debe utilizar el comando siguiente:

```
cd /usr/local/src/postgresql-8.4.1/contrib/[nombre_módulo]
```

Paso 2.- Compilar el módulo seleccionado, para ello se deben emplear los comandos siguientes:

```
make

make install
```

Paso 3.- Autenticarse como el usuario postgres para acceder a la plantilla 1 (template1) del Gestor, para ello utilice el comando siguiente:

```
su - postgres
```

Paso 4.- Aplicar el módulo al template1 para que todas las bases de datos que se crean a partir de él hereden sus funcionalidades, para ello escribir el comando siguiente:

```
/usr/local/pgsql/bin/psql -f /usr/local/src/postgresql-
8.4.1/contrib/nombre_módulo/nombre_módulo.sql -d template1
```

2.4.1.- Procedimiento para la compilación del módulo PL/R

R es un paquete estadístico de última generación al mismo tiempo que un lenguaje de programación, lo cual lo hace muy versátil. Los pasos para instalarlos son los siguientes:

Paso 1.- Instalar las dependencias de R para la compilación del lenguaje, para ello se pueden emplear el comando siguiente:

```
apt-get install r-base r-base-core r-recommended r-base-dev pkg-config
```

Paso 2.- Definir la variable de entorno R_HOME en el profile del usuario postgres, para ello se pueden emplear los comandos siguientes:

```
su - postgres

R_HOME=/usr/lib/R

export R_HOME

exit

# Poner contraseña del usuario para convertirse en un usuario root después del siguiente comando

sudo su

R_HOME=/usr/lib/R

export R_HOME
```

cp -r /usr/share/R/include /usr/lib/R

Paso 3.- Instalar plr, para ello se pueden emplear los comandos siguientes:

cd /usr/local/src/postgresql-8.4.1/contrib/plr

make

make install

Paso 4.- Reiniciar PostgreSQL, para ello se pueden emplear los comandos siguientes:

/etc/init.d/PostgreSQL restart

Paso 5.- Autenticarse como el usuario postgres para acceder a la plantilla 1 (template1) del Gestor, para ello utilice el comando siguiente:

su - postgres

Paso 6.- Aplicar el módulo al template1 para que todas las bases de datos que se creen a partir de él hereden sus funcionalidades, para ello escribir el comando siguiente:

/usr/local/pgsql/bin/psql –f /usr/local/src/postgresql-8.4.1/contrib/plr/ plr.sql -d template1

2.5.- Procedimiento para la eliminación de un módulo

Para llevar a cabo la desinstalación de cualquiera de los módulos se debe ejecutar el fichero unistall_[nombre del módulo].sql generado por la compilación del mismo dentro de la base de datos template1.

Los pasos para desinstalar algún módulo desde el código fuente son los siguientes:

Paso 1.- Cambiar el usuario actual al usuario del sistema, para ello puede escribir el siguiente comando:

su - postgres

Paso 2.- Ejecutar el desinstalador del módulo dentro del template1, para ello escribir el comando siguiente:

/usr/local/pgsql/bin/psql -f /usr/local/pgsql/share/contrib/uninstall_[nombre del módulo].sql -d template1

Paso 3.- Desconectarse como postgres, para ello escribir el comando siguiente:

exit

Paso 4.- Borrar el fichero [nombre del fichero].so que es generado por la compilación, para ello escribir el comando siguiente:

rm /usr/local/pgsql/lib [nombre del módulo].so

Conclusiones

Durante el desarrollo del presente capítulo se definió el procedimiento para la instalación y configuración del Gestor desde el código fuente, se realizó una descripción detallada de los módulos de análisis de datos que podrán ser encontrados en el directorio *contrib* de las fuentes del Gestor, se definió el procedimiento para la compilación de los módulos y el procedimiento para su eliminación.

Capítulo 3

Validación de la propuesta

Introducción

En este capítulo se muestran los resultados de la validación de la Guía en el Centro de Tecnologías de Gestión de Datos (DATEC), como parte de los servicios que ofrece el Centro. Para la realización de la evaluación técnica de la propuesta descrita en el capítulo anterior, se probará el correcto funcionamiento de los módulos de análisis de datos compilados en el *contrib* de *PostgreSQL* 8.4.

3.1.- Validación para el módulo PL/R del *contrib*

PL/R es una extensión del lenguaje *PostgreSQL* que permite escribir funciones agregadas en el lenguaje de análisis estadísticos y gráficos R, dentro del cual se escogió la función *plr_environ()* que muestra el entorno en que el *postmaster* se está ejecutando actualmente. Esto puede ser útil en la depuración de cuestiones relacionadas con R, en específico con las variables de entorno. Se demostrará con un ejemplo específico su correcto funcionamiento.

3.1.1.- Pasos para demostrar el ejemplo seleccionado

Paso 1.- Autenticarse como usuario *postgres*, pudiendo acceder al sistema de ficheros generados por el Gestor y siendo el encargado de ejecutar el motor de bases de datos, para ello utilizar el comando siguiente:

```
su - postgres
```

Paso 2.- Ubicarse dentro de la base de datos en la que se va a realizar la prueba, para ello utilizar el comando siguiente:

```
psql template1
```

Paso 3.- Una vez dentro de la base de datos, realizar la consulta con la función especificada, para ello puede utilizar el siguiente comando, ver figura 2:

```
SELECT * FROM plr_environ();
```

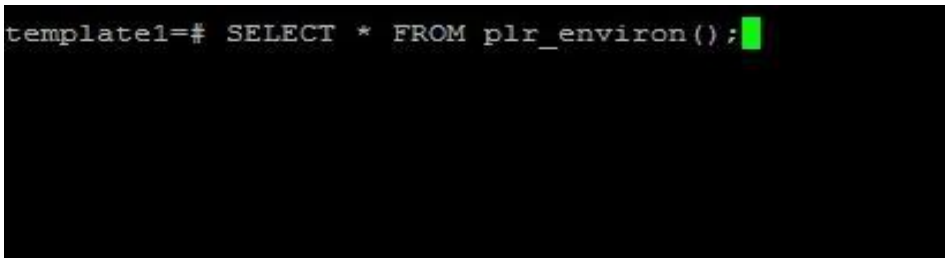


Figura 2.- Ejecución de la función *plr_environ ()* en la base de datos *template1*

Paso 4.- Presionar la tecla ENTER y se mostrará lo que está ejecutando el *postmaster* en ese momento. Con lo que queda demostrado que el módulo PL/R está correctamente compilado. Ver figura 3.

```
template1=# SELECT * FROM plr_environ();
     name      |                                value
---------------+-------------------------------------------------------------
 TERM          | linux
 SHELL         | /bin/bash
 USER          | postgres
 PATH          | /usr/local/pgsql/bin:/usr/local/bin:/usr/bin:/bin:/usr/games
 MAIL          | /var/mail/postgres
 _             | /usr/local/pgsql/bin/postgres
 PWD           | /var/lib/pgsql
 LANG          | en_US.UTF-8
 PGSYSCONFDIR  | /usr/local/pgsql/etc
 HOME          | /var/lib/pgsql
 SHLVL         | 2
 LOGNAME       | postgres
 PGDATA        | /var/lib/pgsql/data
 LC_COLLATE    | en_US.UTF-8
 LC_CTYPE      | en_US.UTF-8
 LC_MESSAGES   | en_US.UTF-8
 LC_MONETARY   | C
 LC_NUMERIC    | C
 LC_TIME       | C
(19 rows)

template1=#
```

Figura 3.- Resultado de la ejecución de la función *plr_environ ()* en la base de datos *template1*

3.2.- Validación para el módulo *cube* del *contrib*

El módulo *cube* implementa un tipo de datos para la representación de cubos multidimensionales. Se realizará una prueba con la función *cube_union(cube1, cube2)* que devuelve la unión de los dos cubos pasados como parámetros.

3.2.1.- Pasos para demostrar el ejemplo seleccionado

A continuación se muestran los pasos que dan lugar a la ejecución de la prueba:

Paso 1.- Se debe autenticar como usuario *postgres*, para poder acceder al sistema de ficheros generados por el Gestor y ser el encargado de ejecutar el motor de bases de datos, para ello utilizar el comando siguiente:

su - postgres

Paso 2.- Debe ubicarse dentro de la base de datos en la que se va a realizar la prueba, para ello utilizar el comando siguiente:

psql template1

Paso 3.- Una vez que esté dentro de la base de datos, debe realizar la consulta con la función especificada, para ello puede utilizar el siguiente comando o Ver figura 4:

SELECT cube_union ('(0,5,2),(2,3,1)', '0');

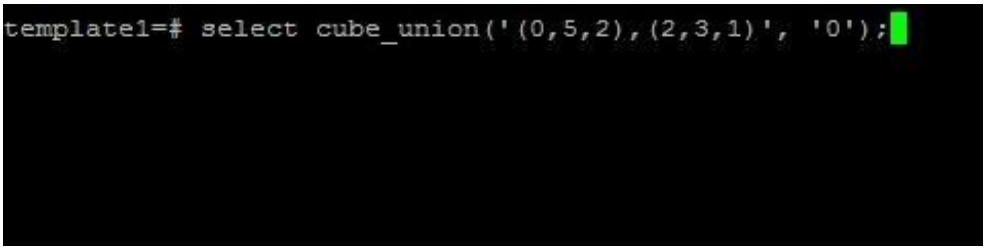


Figura 4.- Ejecución de la función *cube_union ('(0,5,2),(2,3,1)', '0')* en la base de datos *template1*

Paso 4.- Presionar la tecla ENTER y se mostrará el resultado de la unión de los dos cubos. Con lo que queda demostrado que el módulo *cube* está correctamente compilado. Ver figura 5.

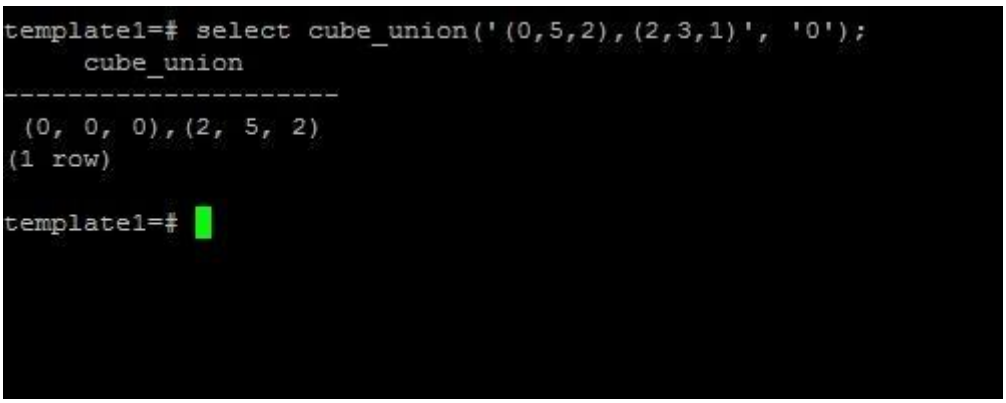


Figura 5.- Resultado de la ejecución de la función *cube_union ('(0,5,2),(2,3,1)', '0')* en la base de datos *template1*

3.3.- Validación para el módulo *fuzzystrmatch* del *contrib*

El módulo *fuzzystrmatch* proporciona varias funciones para determinar las similitudes y la distancia entre las cadenas. Una de estas funciones es *levenshtein* la que permite conocer la distancia de edición o distancia entre palabras, brindando el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, una inserción, eliminación o sustitución de un caracter.

Por ejemplo, la distancia entre "casa" y "calle" aplicando dicha función es de tres, porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro.

1. casa → cala (sustitución de 's' por 'l')
2. cala → calla (inserción de 'l' entre 'l' y 'a')
3. calla → calle (sustitución de 'a' por 'e')

3.3.1.- Pasos para demostrar el ejemplo seleccionado

Paso 1.- Autenticarse como usuario *postgres*, pudiendo acceder al sistema de ficheros generados por el gestor y siendo el encargado de ejecutar el motor de bases de datos, para ello utilizar el comando siguiente:

```
su - postgres
```

Paso 2.- Ubicarse dentro de la base de datos en la que se va a realizar la prueba, para ello utilizar el comando siguiente:

```
psql template1
```

Paso 3.- Una vez dentro de la base de datos, realizar la consulta con la función especificada, para ello puede utilizar el siguiente comando, ver figura 6:

```
SELECT levenshtein ('CASA', 'CALLE');
```

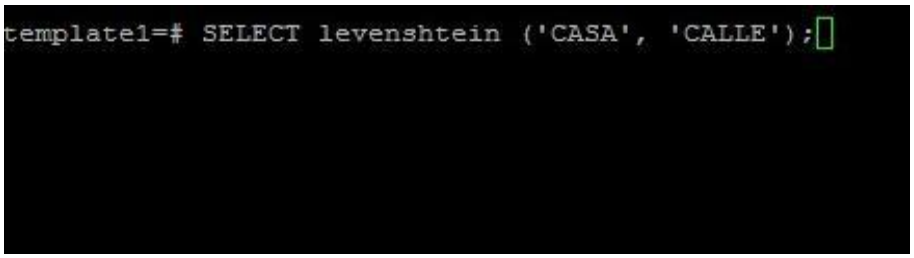


Figura 6.- Ejecución de la función *levenshtein ('CASA', 'CALLE')* en la base de datos *template1*

Paso 4.- Presionar la tecla ENTER, mostrando la cantidad de cambios que se deben hacer para que la primera palabra se convierta en la segunda. Con lo que queda demostrado que el módulo *fuzzystrmatch* está correctamente compilado. Ver figura 7.

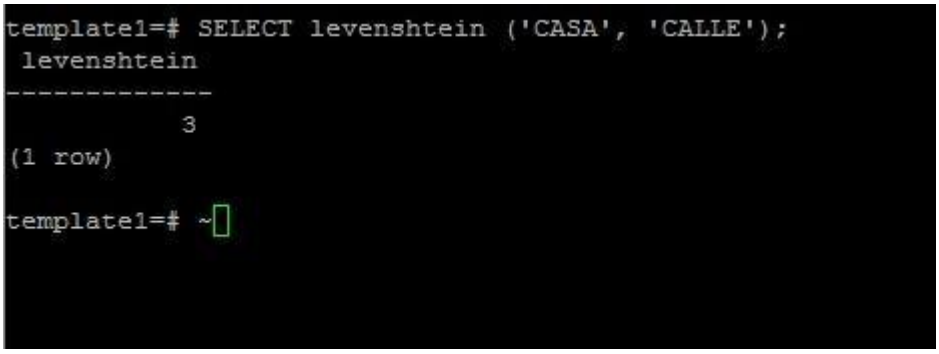


Figura 7.- Resultado de la ejecución de la función *levenshtein ('CASA', 'CALLE')* en la base de datos *template1*

Conclusiones

A partir de los resultados obtenidos en la puesta en práctica de los pasos propuestos en la Guía se puede llegar a la conclusión de que (1) se demostró la solución propuesta en su totalidad debido a que todas las condiciones del entorno se crearon satisfactoriamente y (2) los resultados obtenidos fueron probados correctamente.

Conclusiones generales

Basado en todo el proceso investigativo del trabajo de diploma para el desarrollo de la Guía para la personalización de *PostgreSQL* 8.4 se arribaron a las siguientes conclusiones generales:

- Se realizó un estudio del estado del arte de los sistemas de gestión de bases de datos, de *PostgreSQL* como el gestor de código abierto más potente, de los módulos de análisis de datos, monitoreo, administración, desarrollo y seguridad que han sido desarrollados para él y sus potencialidades, de los productos y servicios que ofrecen las empresas líderes basados en él, además del estudio de los posibles compiladores a emplear para la habilitación de dichos módulos.
- Se definió un procedimiento para la instalación del Gestor desde su código fuente.
- Se compilaron 32 módulos contemplados en las categorías definidas, conformando el procedimiento para la instalación y configuración de los mismos.
- Se definió un procedimiento para la eliminación de los módulos una vez compilados.
- Se definieron las recomendaciones y buenas prácticas para habilitar los módulos de forma general.
- Se conformó la Guía para la personalización del gestor con todos los procedimientos y recomendaciones definidas
- Fueron evaluados los pasos de la Guía, observando los resultados satisfactorios que arrojaron las pruebas realizadas, lo que da una garantía de la validez de la investigación.

Por todo lo anterior se concluye que los objetivos propuestos para la presente investigación han sido cumplidos satisfactoriamente.

Recomendaciones

Independientemente de que se hayan alcanzado los objetivos trazados al inicio de la investigación se recomienda:

- Analizar, describir e incluir como parte de la Guía para la personalización del gestor el resto de los módulos que no son de análisis de datos, monitoreo, administración, desarrollo y seguridad, abarcando las nuevas funcionalidades desarrolladas por esta vía.
- Incluir en las descripciones de los módulos un elemento que especifique cómo emplearlos.
- Actualizar la Guía con las nuevas funcionalidades que se desarrollen para versiones posteriores a la 8.4.
- Crear otras guías con la misma estructura para otros sistemas operativos usados en los servidores como *RedHat*, *Fedora* y *Ubuntu*.
- Empaquetar las fuentes del gestor con los módulos compilados en un ejecutable, para facilitar su empleo por los usuarios finales.

Referencias bibliográficas

Korth, Henry F. 1995. *Fundamentos de Base de Datos. s.l. : Segunda edición, 1995.*

Marqués Andrés, María Mercedes. 2001. *Apuntes de ficheros y bases de datos. 2001.*

Hansen, Gary W. y Hansen, James V. 1999. *Diseño y administración de bases de datos. s.l. : Editorial Prentice Hall, 1999.*

Arencibia, Daniel Fernández y Santos, Yunieski Fábregas. 2007. *Administración, configuración y optimización de un Sistema de Bases de Datos Descentralizado en Oracle Database 10g release 2. Ciudad de la Habana : s.n., 2007.*

ibercom. 2006. Sistema de gestión de bases de datos. [En línea] 2006. [Citado el: 24 de febrero de 2010.] Disponible en: https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=85

PostgreSQL Global Development Group. 2009. *PostgreSQL 8.4.1 Documentation. California : s.n., 2009. pág. 2119.*

PostgreSQL Global Development Group . 2010. PostgreSQL. [En línea] 2010. [Citado el: 2 de marzo de 2010.] Disponible en: http://www.postgresql.org

Martinez, Rafael. 2009. PostgreSQL-es.org. [En línea] 2009. [Citado el: 15 de diciembre de 2009.] Disponible en: http://www.postgresql-es.org/sobre_postgresql

Medrano, Jesus Rafael Sanchez. 2009. *Introducción a Base de Datos con PostgreSQL.* 2009. pág. 69.

Casanova, Jaime y Herrera, Álvaro. 2009. *MVCC. [presentación] Ciudad de la Habana : s.n., 2009.*

Herrera, Álvaro y Casanova, Jaime. 2009. *WAL. [presentación] Ciudad de la Habana : s.n., 2009.*

Espinoza, Humberto. 2005. PostgreSQL: Una Alternativa de DBMS Open Source. [En línea] 2005. [Citado el: 21 de noviembre de 2009.] Disponible en: http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf

Ibarra, Antonio Aliaga y Flores, Marcos Agustin Miani. 2008. iessanvicente.com. [En línea] 21 de enero de 2008. [Citado el: 16 de febrero de 2010.] Disponoble en: http://www.iessanvicente.com/colaboraciones/postgreSQL.pdf

Catrin, Franco. 2009. FayerWayer.com. [En línea] 1 de julio de 2009. [Citado el: 23 de enero de 2010.] Disponible en: http://www.fayerwayer.com/2009/07/postgresql-84-mantiene-distancia-con-sus-rivales

Herrera, Álvaro. 2009. *PostgreSQL 8.4. [presentación] Ciudad de la Habana : s.n., 2009.*

EnterpriseDB Corporation. 2009. *Beyond PostgreSQL-A Look Inside Postgres Plus Advanced Server. 2009. pág. 39.*

EnterpriseDB Corporation. 2010. EnterpriseDB: The Enterprise Postgres Company. [En línea] 2010. [Citado el: 12 de febrero de 2010.] Disponible en: http://www.enterprisedb.com/products/postgres_plus_as/overview.do

Commamd Prompt. 2010. Commamd Prompt: The PostgreSQL Company. [En línea] 2010. [Citado el: 17 de febrero de 2010.] Disponible en: http://www.commandprompt.com

2ndQuadrant. 2010. 2ndQuadrant Profesional PostgreSQL. [En línea] 2010. [Citado el: 16 de febrero de 2010.] Disponible en: http://www.2ndquadrant.com

Skype Limited. 2008. Skype Developer Zone. [En línea] 2008. [Citado el: 19 de febrero de 2010.] Disponible en: https://developer.skype.com/SkypeGarage/DbProjects/SkyTools#head-1adf6fc652fd3a7bbc7851058ef423d6250c0629

PostgreSQL Experts. 2009. PGX PostgreSQL Experts. [En línea] 2009. [Citado el: 17 de febrero de 2010.] Disponible en: http://www.pgexperts.com

Continuent Inc. 2010. Continuent. [En línea] 2010. [Citado el: 17 de febrero de 2010.] Disponible en: http://www.continuent.com

LinuxLinks.com. 2009. LinuxLinks.com. [En línea] 13 de agosto de 2009. [Citado el: 25 de noviembre de 2009.] Disponible en: http://www.linuxlinks.com/article/20090730140634585/clang.html

LinuxLinks.com. 2009. LinuxLinks.com. [En línea] 13 de agosto de 2009. [Citado el: 25 de noviembre de 2009.] Disponible en: http://www.linuxlinks.com/article/20090730140503307/g95.html

LinuxLinks.com. 2009. LinuxLinks.com. [En línea] 13 de agosto de 2009. [Citado el: 25 de noviembre de 2009.] Disponible en: http://www.linuxlinks.com/article/20090801134745804/LLVM.html

LinuxLinks.com. 2009. LinuxLinks.com. [En línea] 13 de agosto de 2009. [Citado el: 25 de noviembre de 2009.] Disponible en: http://www.linuxlinks.com/article/20071021090044634/GCC.html

EnterpriseDB Corporation. 2010. EnterpriseDB: The EnterpriseDB Postgres Company. [En línea] 2009. [Citado el: 9 de marzo de 2010.] Disponible en: http://www.enterprisedb.com/docs/en/8.4/pg/contrib.html

PostgreSQL CVSweb. 2010. PostgreSQL CVSweb. [En línea] 2010. [Citado el: 1 de marzo de 2010.] Disponible en: http://anoncvs.postgresql.org/cvsweb.cgi/pgsql/contrib

PostgreSQL Global Development Group . 2010. PostgreSQL. [En línea] 2010. [Citado el: 2 de marzo de 2010.] Disponible en: http://www.postgresql.org/docs/current/static/contrib.html

bostongis.com. 2010. Boston Geographic Information Systems. [En línea] 2010. [Citado el: 12 de abril de 2010.] Disponible en: http://www.bostongis.com/PrinterFriendly.aspx?content_name=postgresql_plr_tut01

doxygen. 2010. Veil. [En línea] 2010. [Citado el: 12 de abril de 2010.] Disponible en: http://veil.projects.postgresql.org/curdocs/index.html

pgfoundry.org. 2010. pgFoundry. [En línea] 2010. [Citado el: 12 de abril de 2010.] Disponible en: http://pgfoundry.org/projects/veil

Bibliografía

eaprende.com. 2008. Gestor de Base de Datos: MySQL, POstgreSQL, SQLite. Gestor de Base de Datos: MySQL, POstgreSQL, SQLite. [En línea] 2008. [Citado el: 10 de enero de 2010.] Disponible en: http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html

PostgreSQL Global Development Group. 2010. PostgreSQL. [En línea] 2010. [Citado el: 25 de febrero de 2010.] Disponible en: http://www.postgresql.org/docs/8.4/static/release-8-4-1.html

PostgreSQL Global Development Group. 2009. PostgreSQL. [En línea] 2009. [Citado el: 2 de marzo de 2010.] Disponible en: http://www.postgresql.org/about/press/presskit82.html.es

PostgreSQL Global Development Group. 2010. PostgreSQL. [En línea] 2010. [Citado el: 12 de abril de 2010.] Disponible en: http://www.postgresql.org/docs/8.4/static/features.html

PostgreSQL Global Development Group. 2010. PostgreSQL. [En línea] 2010. [Citado el: 23 de enero de 2010.] Disponible en: http://www.postgresql.org/docs/8.4/static/release-8-4.html

PostgreSQL Global Development Group. 2010. PostgreSQL. [En línea] 2010. [Citado el: 12 de abril de 2010.] Disponible en: http://www.postgresql.org/docs/8.4/static/features-sql-standard.html

Anexos

Anexo 1.- Ficha de descripción de los módulos del *contrib*

Elemento	Descripción
Descripción	*<Características y funcionalidades del módulo>*
Clasificación	*<A cuál de las siguientes clasificaciones pertenece: Análisis de datos, Monitoreo, Administración, Desarrollo o Seguridad. En caso de que no se encuentre en una de las clasificaciones anteriores se debe especificar a qué nueva clasificación se acogería>*
Fecha de creación	*<Fecha exacta de la creación del módulo>*
Lugar de origen	*<Lugar donde fue creado: universidad, centro de desarrollo, empresa>*
Desarrolladores	*<Nombre y breve perfil de los desarrolladores: dónde trabajan, grado científico>*
Lenguaje de desarrollo	*<En qué lenguaje está desarrollado>*
Fecha de última versión	*<dd-mm-aaaa de la versión que se empaquetará>*
Soporte	*<Institución, empresa o sujeto que brinda soporte a la solución>*
Grado de generalización	*<Nivel de uso del módulo>*
Comandos para su instalación	*<Explicación de los comandos con que se instala y su configuración>*
Ubicación	*<Dirección donde descargar el paquete>*

Glosario de términos

API: Interfaz de Programación de Aplicaciones (*Application Programming Interface)*. Conjunto de especificaciones de comunicación entre componentes software; conjunto de llamadas al sistema que ofrecen acceso a los servicios del sistema desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

BSD: *Berkeley Software Distribution*, sistema operativo derivado del *Unix* y nacido a partir de las aportaciones realizadas a ese sistema por la Universidad de California en *Berkeley*. Esta licencia pertenece al grupo de licencias de software libre y tiene menos restricciones en comparación con otras, estando muy cercana al dominio público.

Cliente/Servidor: arquitectura de comunicación en la que un servidor central actúa como coordinador de la comunicación. Todos los mensajes deben pasar por él antes de llegar al destino.

Compilador: es un programa que se encarga de traducir los programas escritos por el programador en lenguaje de alto nivel a un lenguaje de bajo nivel, comprensible por la máquina y que permite que pueda ser ejecutado por la computadora.

Data Architect: es una herramienta de diseño de modelado e integración de datos para una empresa; diseñada para ayudar a los arquitectos de datos a diseñar bases de datos relacionales y federadas, comprender los elementos de datos valiosos, sus relaciones y para optimizar los proyectos de base de datos.

Disparador o (*trigger*): es la acción definida en una tabla de una base de datos y ejecutada automáticamente por una función programada por un usuario. Es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción, actualización o eliminación en una tabla.

Índices hash: los índices son una forma común de mejorar el rendimiento de las bases de datos y le permite al servidor de la base de datos encontrar y recuperar filas mucho más rápido de lo que podría hacer sin un índice.

Integridad referencial: es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y

que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

Lenguajes procedurales: estos lenguajes no están construidos dentro de *PostgreSQL*. Son lenguajes de programación compilados en forma de objeto compartido y cargado cuando sean necesarios.

Lenguaje de programación: es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Mapa de espacio libre o (Free Space Map): está organizado como un árbol de páginas. Las páginas de nivel inferior almacenan el espacio libre disponible en cada montón o índice, utilizando un byte para representar cada página. Los niveles superiores de información agregada de los niveles inferiores.

Mapa de visibilidad: simplemente almacena un *bit* por montón de la página. Un *bit* significa que todas las tuplas de la página se sabe que son visibles para todas las transacciones.

Multiplataforma: se refiere a los programas, sistemas operativos, lenguajes de programación u otra clase de software que puedan funcionar en diversas plataformas.

MVCC: control concurrente de múltiples versiones, es el método que usa *PostgreSQL* para controlar la consistencia de datos, cuando varios procesos acceden a la misma tabla de la base de datos.

Open source: es código abierto, en inglés es el término con el que se conoce al software distribuido libremente.

O-RDBMS: sistemas de gestión de bases de datos objeto-relacionales, es un gestor de bases de datos que desde el modelo relacional evoluciona hacia uno más extenso y complejo, incorporando para obtener este fin, conceptos del modelo orientado a objetos.

PgAccess: proporciona una interfaz gráfica para *Postgres* donde se pueden gestionar las tablas, editarlas, definir consultas, secuencias y funciones.

PgAdmin: es una aplicación gráfica para gestionar el gestor de bases de datos *PostgreSQL*, siendo la más completa y popular con licencia *Open source*.

Pg-migrator: permite la migración entre los lanzamientos principales de *PostgreSQL* sin una sobrecarga de datos.

Python: es un lenguaje de programación interpretado, permite dividir el programa en módulos reutilizables desde otros programas *Python*.

Servidor Web Apache: Apache es un servidor del protocolo HTTP, comúnmente llamado servidor Web pues es la mayor utilidad para dicho protocolo. Puede alojar varios sitios y pueden coexistir varios servidores Apache en un sólo equipo.

SPI: Interfaz de programación de servidor (*Server Programming Interface*), es una interfaz de programación que provee a los desarrolladores de funciones en el lenguaje C la habilidad de ejecutar comandos SQL dentro de las mismas.

SQL: Lenguaje de consulta estructurado (*Structured Query Language)*, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones.

Tiempo de inactividad o (downtime): tiempo en el que un ordenador no está disponible al usuario, debido normalmente a razones de mantenimiento del equipo.

Tora: es un conjunto de herramientas multiplataforma de software libre creado para ayudar a los administradores y desarrolladores de aplicaciones de bases de datos *Oracle*. También suministra soporte para *MySQL* y *PostgreSQL*.

TPC-B: está diseñado para hacer escenarios de pruebas en la parte principal de un sistema de base de datos, es decir, se centra en la parte de un sistema que realiza el procesamiento de transacciones reales.

Unix: es un sistema operativo portable, multitarea y multiusuario; desarrollado en 1969 por un grupo de empleados de los laboratorios *Bell* de *AT&T*.

VACUUM: recupera espacio de almacenamiento ocupado por tuplas que han sido borradas.

WAL: Log de escritura adelantada (*Write-Ahead Log*), es un componente del sistema *PostgreSQL* que se encarga de asegurar la integridad de los datos.

WAN: Red de Área Amplia (*Wide Area Network*), es un tipo de red de computadoras que se extienden sobre un área geográfica extensa.