Universidad de las Ciencias Informáticas

Facultad 2

Migración del diseño y administración de la base de datos del Sistema de Gestión de Emergencias y Seguridad Ciudadana (171) hacia un gestor de base de datos libre

Trabajo de Diploma

Presentado para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Daymel Bonne Solís

Tutores: Ing. Daniel Ernesto Vargas

Ing. Henry Góngora Columbié

"Año del 50 Aniversario del Triunfo de la Revolución"

Ciudad de la Habana, Cuba.

Declaración de auditoría

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

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

Daymel Bonne Solís	**Ing.Daniel Ernesto Vargas**	**Ing. Henry Góngora Columbié**
Autor	**Tutor**	**Tutor**

Dedicatoria

Agradecimientos

RESUMEN

Como parte de la estrategia de migración a software libre de los componentes del Sistema de Gestión de Emergencias de Seguridad Ciudadana 171, surge esta investigación, que propone una estrategia acerca de cómo realizar el proceso de migración del diseño y la administración de la base de datos de ese sistema hacia un gestor de bases de datos libre, encaminada una lograr la soberanía tecnológica y económica.

Esta investigación dará paso a lograr una base de datos gestionada por un Sistema Gestor de Bases de Datos libre que soporte las necesidades de almacenamiento de información del Sistema de Gestión de Emergencias de Seguridad Ciudadana 171.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

Introducción

En la lucha mundial por lograr la soberanía tecnológica y garantizar la democratización y apropiación social de las tecnologías de información, un conjunto de países como Cuba, Brasil y Venezuela se han propuesto un plan de migración hacia software libre.

Uno de los países donde con más fuerza se ha llevado a cabo este plan es en la República Bolivariana de Venezuela, donde su presidente Hugo Chávez Frías, emitió el Decreto N° 3.390, publicado en la Gaceta Oficial Nº 38.095 de fecha 28/12/2004 sobre el uso obligatorio del software libre en el país para todas las dependencias públicas de carácter oficial. De esta forma, el ejecutivo nacional establece que es prioridad del Estado incentivar y fomentar la producción de bienes y servicios para satisfacer las necesidades de la población, mediante el uso de estas herramientas desarrolladas con estándares abiertos para robustecer la industria nacional, aumentando y aprovechando sus capacidades y fortaleciendo su soberanía (1).

Además el artículo 110 de la Constitución de la República Bolivariana de Venezuela, reconoce como de interés público la ciencia, la tecnología, el conocimiento, la innovación y los servicios de información, con el objeto de lograr el desarrollo económico, social y político del país. Esta disposición constitucional se expresa con fuerza en los artículos 1° de la Ley de Telecomunicaciones y 12° de la Ley Orgánica de la Administración Pública. Con el Decreto N° 825, emitido el 10 de Mayo de 2000, se establece el acceso y el uso de Internet como política prioritaria para el desarrollo cultural, económico, social y político del Estado.

Venezuela cuenta con alrededor de 500 empresas dedicadas a la integración de sistemas, el desarrollo y la comercialización de software propios a terceros utilizando software propietarios, las cuales generan más de 35 mil empleos entre directos e indirectos; cerca del 50% de estas empresas dedicadas al software propietario, exportan sus productos principalmente hacia Latinoamericana, vendiéndolos a países a altos precios; por otro lado encontramos a la industria de software libre en Venezuela, la cual se encuentra en pleno surgimiento gracias al auge mundial que se le ha dado como alternativa a las plataformas soportadas por sistemas de operación como Unix y Windows. (2)

Introducción

Otro factor que ha ayudado al crecimiento del software libre en Venezuela es el apoyo recibido desde el Gobierno Nacional con un marco regulatorio que promueve la utilización de software libre principalmente en la Administración Pública Nacional. Si bien es cierto que la Industria del Software Libre en Venezuela está creciendo, aun no se encuentra al nivel de la Industria de Software propietario, por lo tanto se requiere seguir potenciando la industria nacional así como el fortalecimiento del Plan de Migración al Software Libre de la Administración Pública Nacional (APN) para alcanzar a mediano plazo una Industria de Software Libre Nacional de alta calidad.

Una de las estrategias de migración propuestas es la concerniente a los Sistemas Gestores de Base de Datos (SGBD) utilizados en soluciones de software a determinados problemas de la sociedad venezolana, destacando que aunque en Venezuela existe una sucursal del SGBD más usado a nivel empresarial (Oracle) el país enfrenta un gran problema de usabilidad del mismo, este problema es sin lugar a dudas las grandes sumas de dinero que se debe pagar por la licencia de uso, trayendo como consecuencia que muchas dependencias del gobierno esparcidas por todo el país en ocasiones no cuentan con el dinero suficiente para financiar el pago de dicha licencia, elemento este que hace compleja la creación de soluciones informáticas para dichas dependencias que utilicen Oracle como sistema gestor de bases de datos, por lo que se quiere llevar a cabo la migración a SGBD libres.

Actualmente en la República Bolivariana de Venezuela uno de los sistemas informáticos que tiene sus datos gestionados por el SGBD Oracle es el Sistema de Gestión de Emergencias de Seguridad Ciudadana 171 (SIGESC 171), en el cual la base de datos es utilizada para:

- Almacenamiento de datos.
- Establecer un punto de intercambio de información entre los diferentes módulos.
- Servir como fuente de generación de reportes estadísticos.
- Mantener la disponibilidad e integridad de la información.
- Servir como fuente de datos para otros sistemas informáticos del Ministerio de Justicia.
- Garantizar la confidencialidad y seguridad de la información manipulada.

Introducción

En el servidor se almacenan todos los datos de configuración de las aplicaciones y de los usuarios, necesaria para que el sistema funcione correctamente. Además se hospedan los datos históricos que se generan como resultado de las solicitudes realizadas por la población.

De esta forma las aplicaciones buscan información en el servidor de base de datos la mayor parte del tiempo para su funcionamiento, datos que asincrónicamente han sido insertados, actualizados o eliminados por estas mismas aplicaciones, estableciendo así un medio de comunicación entre ellas.

Dándole cumplimiento a lo planteado en el Decreto N° 3.390, se decide por parte de la dirección del SIGESC 171 migrar el SGBD utilizado en la solución (Oracle) a un SGBD libre.

Problema Científico

Como resultado de lo analizado anteriormente surge el siguiente **problema científico** ¿Cómo realizar la migración del diseño y administración de la base de datos del SIGESC 171 hacia un Sistema Gestor de Bases de Datos libre?

El **objeto de estudio** son los Sistemas Gestores de Base de Datos. Siendo el **campo de acción** los Sistemas Gestores de Bases de Datos Libres.

Para dar solución al problema planteado con anterioridad se propone como **objetivo general** de la investigación:

Desarrollar una estrategia para la migración del diseño y administración de la base de datos del SIGESC 171 hacia un Sistema Gestor de Bases de Datos libre.

Para desarrollar satisfactoriamente la investigación se definieron un conjunto de tareas de investigación que permiten cumplir el objetivo general propuesto, estas son:

1. Caracterizar los principales SGBD libres que existen y seleccionar hacia cual se realizará el proceso de migración.
2. Establecer las fases necesarias para llevar a cabo la migración.

3. Describir cada elemento o paso para lograr la migración, manteniendo y en lo posible mejorando lo que existe en la base de datos del SIGESC 171.

4. Realizar la validación de la solución propuesta.

El presente documento esta compuesto por las siguientes partes: resumen, introducción, desarrollo, conclusiones, recomendaciones, referencias bibliográficas, anexos y glosario de términos. El desarrollo está estructurado en 3 capítulos:

- Capítulo I: “Fundamentación Teórica”, incluye los aspectos teóricos que soportan esta investigación. Se describen los principales SGBD libres. Selección del gestor de base de datos.

- Capítulo II: “La Migración”, se describe el proceso a seguir para lograr la migración del diseño, administración y configuración de la base de datos del SIGESC 171 hacia el SGBD seleccionado.

- Capítulo III: “Validación de la solución propuesta”, incluye el método y los resultados de la validación teórica de la solución.

1 Fundamentación Teórica

Este capítulo abarca:

- Licencias de Software
- Introducción a las Bases de Datos
- Comparación entre Sistemas de Bases de Datos Libres

CAPÍTULO I

Fundamentación Teórica

1.1 Introducción

En este capítulo se presentan un grupo de conceptos teóricos a los que se hará referencia en el resto del trabajo. Se abordan aspectos relacionados con el uso de las nuevas tecnologías de la informática y las comunicaciones. Se hace referencia a conceptos importantes dentro del mundo de las bases de datos y temas relacionados con algunos de los SGBD Libres.

1.2 Proceso de migración a Software Libre

La migración a Software Libre se refiere a un conjunto de actuaciones cuya finalidad es la sustitución de infraestructuras tecnológicas apoyadas en software propietario por otras con funciones equivalentes basadas en Software Libre (3).

Esta migración proporciona beneficios tanto económicos como prácticos. No es una mera cuestión técnica, sino que es una cuestión de uso, y debe abordarse de esta forma para tener éxito en la misma, las características principales son:

- Alta fiabilidad.
- Facilidad de ampliar y personalizar.
- Reducción de virus.
- Amplio soporte tanto comercial como comunitario.
- Ahorro económico en licencias de software.

1.3 Licencias de Software

Una licencia es aquella autorización formal con carácter contractual que un autor de un software da a un interesado para ejercer "actos de explotación legales" (4). Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del software libre, existen distintas variantes del concepto o grupos de licencias:

1.3.1 Licencias GPL

Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL (5).

Es decir, la licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Y añade que si se reutiliza en un mismo programa código "A" licenciado bajo licencia GNU GPL y código "B" licenciado bajo otro tipo de licencia libre, el código final "C", independientemente de la cantidad y calidad de cada uno de los códigos "A" y "B", debe estar bajo la licencia GNU GPL.

En la práctica esto hace que las licencias de software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

En el sitio web oficial de GNU hay una lista de licencias que cumplen las condiciones impuestas por la GNU GPL y otras que no.

Aproximadamente el 60% del software licenciado como software libre emplea una licencia GPL. (6)

1.3.2 Licencias estilo BSD

Llamadas así porque se utilizan en gran cantidad de software distribuido junto a los sistemas operativos BSD. El autor, bajo tales licencias, mantiene la protección de copyright únicamente para la renuncia de

garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles. Puede argumentarse que esta licencia asegura “verdadero” software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre (4).

1.4 Base de Datos

En los sistemas informáticos hoy es muy importante la presencia de bases de datos que almacenen la información.

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta (7).

Estos sistemas se clasifican según la forma en que se diseña la estructura de almacenamiento, pueden ser jerárquicas, de red, relacional, orientada a objetos, entre otras clasificaciones.

1.4.1 Base de Datos Jerárquica

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas (8).

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

1.4.2 Base de Datos de Red

Éste es un modelo ligeramente distinto del jerárquico su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico) (9).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aún así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

1.4.3 Base de Datos Relacional

Una base de datos relacional es una base de datos basada en un modelo relacional. Estrictamente hablando el término se refiere a una colección específica de datos, pero a menudo es usado como sinónimo del software utilizado para gestionar esa colección de datos. Ese software se conoce como sistema gestor de base de datos relacional o RDBMS (Relational Database Management System) (10).

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia. Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. (10)

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL(Structured Query Language o Lenguaje Estructurado de Consultas), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

1.4.4 Base de Datos Orientada a Objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos y que los implementa en sus operaciones, estos conceptos son los siguientes (11):

- Encapsulación - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

Se está trabajando en SQL3, que es el estándar de SQL92 ampliado, que soportará los nuevos conceptos orientados a objetos y mantendría compatibilidad con SQL92.

1.4.5 Bases de Datos Documentales

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes (12).

1.4.6 Bases de Datos Deductivas

Un sistema de base de datos deductivos, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática (12).

1.4.7 Bases de Datos Distribuidas

La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades y sucursales de tiendas (12).

1.5 Sistemas gestores de Bases de Datos Libres

1.5.1 PostgreSQL

El padre de PostgreSQL es el eminente profesor de la Universidad de California (Berkeley para los proyectos públicos) Michael Stonebraker, que lo crea a partir de otro popular proyecto llamado Ingres, el cual estaba basado en la aplicación de las teorías de las bases de datos relacionales. En el año 1986, cambia el nombre del proyecto a Postgres, con el objetivo de aplicar los conceptos de bases de datos Objetos Relacionales. (13)

En 1995, vuelve a cambiar el nombre por Postgres95, continuando el trabajo de Stonebraker dos de sus más prominentes alumnos, que luego derivaría en PostgreSQL, por el hecho de que estos alumnos le incluyeron el SQL (Structured Query Language) al mismo. Ya en versiones recientes PostgreSQL cuenta con su propio lenguaje procedural, llamado PL/pgSQL, que comparte algunas características con el PL/SQL de Oracle. En 1996, el proyecto cambia su concepto al mundo del código abierto e inicia su

versión 6.0. En el año 2000 se comienza la implementación del soporte para Ipv6. Corre el año 2004, y ya PostgreSQL es reconocido como uno de los mejores motores de bases de datos del mundo, y es la 5ta DBMS más popular en Estados Unidos ese mismo año. En el año 2005, PostgreSQL pasa la prueba de Coverty Inspected, en la cual encontraron sólo 20 errores en 775,000 líneas de código, lo cual constituyó un orgullo y un compromiso para el proyecto. (13)

Ya en el 2006, PostgreSQL llega a su versión 8.0, la cual crece y se mejora por día. Actualmente la última versión estable es la 8.3.7, que mejora muchas características de las versiones anteriores, combinando rapidez con estabilidad, pues a PostgreSQL se le catalogaba de "lento" con respecto a otros productos como MySQL 5.0 y Oracle 10g (13). Algunas de las características principales de este excelente producto son:

SGBD Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Control de Concurrencia Multi-Versión (MVCC)

MVCC, es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios (14). Cuando en un sistema se utiliza algún SGBD con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las que una lectura tiene que esperar para acceder a la información de la base de datos.

La espera es provocada por usuarios que están escribiendo en dicha base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles. (14)

Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como WAL incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean guardados en la base de datos (15).

Esto garantiza que en el hipotético caso de que la base de datos deje de funcionar o se bloquee, existirá un registro de transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en caso de que la dase de datos quede fuera de servicios o se inhabilite ya que cualquiera de los cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez que el sistema haya quedado restaurado, un usuario puede continuar trabajando en el punto en que lo dejó. (15)

Altamente Extensible

PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario. (16)

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el SGBD PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike. (13)

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo llamado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad de usar Perl, Python, o TCL como lenguaje procedural embebido. (13)

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. (13)

Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (JOIN) SQL92.

Multiplataforma

Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc. (34 plataformas soportadas). (13)

Excelente Documentación

Documentación muy bien organizada, pública y libre, con comentario de los propios usuarios.

Gran comunidad de usuarios y desarrolladores

Comunidades muy activas, varias comunidades en castellano.

Soporte para disparadores, procedimientos almacenados, índices, secuencias, etc.

Soporte de todas las características de una base de datos profesional (disparadores, procedimientos almacenados (en PostgreSQL conocida como funciones), secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)

Soporte para SSL

Soporte de protocolo de comunicación encriptado por SSL.

Características adicionales para servicios de alta disponibilidad

Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

Características para la mejora y optimización de la base de datos.

Utilidades para la limpieza de datos (**Vacuum**) así como para el análisis y optimización de consultas (**Explain**).

Tipo de almacenamiento para datos grandes

Almacenaje especial para tipos de datos grandes (TOAST)

Los límites para el almacenamiento en PostgreSQL son:

- Máximo de base de Datos: ILIMITADO.
- Máximo de tamaño de Tabla: 32TB.
- Máximo de tamaño de Registro: 1.6TB.
- Máximo de tamaño de Campo: 1GB.
- Máximo de registros por Tabla: ILIMITADO.
- Máximo de campos por Tabla: 250 a 1600 (depende de los tipos usados).
- Máximo de índices por Tabla: ILIMITADO.
- Número de lenguajes en los que se puede programar funciones: aproximadamente 10 (PL/pgsql, PL/java, pl/perl, pl/python, tcl, PL/php, C, C++, Ruby, etc.).
- Métodos de almacenamiento de índices: 4 (B-tree, Rtree, Hash y GisT)

Licencia de PostgreSQL

PostgreSQL está bajo la licencia BSD.

1.5.2 MySQL

MySQL es un Sistema de Gestión de Base de Datos relacional, multihilo y multiusuario con más de seis millones de instalaciones, actualmente es el gestor de bases de datos de código abierto más popular del mundo. La versión estable más reciente es la 5.1. (17)

MySQL es una idea originaria de la empresa MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL[1] para conectar las tablas usando rutinas de bajo nivel, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

MySQL es muy utilizado en aplicaciones web como, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones**InnoDB**

InnoDB es una tecnología de almacenamiento de datos de fuente abierta para MySQL, incluido como formato de tabla estándar en todas las distribuciones de MySQL AB a partir de las versiones 4.0. Su característica principal es que soporta transacciones de tipo ACID y bloqueo de registros e integridad referencial. InnoDB ofrece una fiabilidad y consistencia muy superior a MyISAM, la anterior tecnología de tablas de MySQL, si bien el mejor rendimiento de uno u otro formato dependerán de la aplicación específica (18).

MyISAM

MyISAM es la tecnología de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales MySQL. Este tipo de tablas están basadas en el formato ISAM[2] siglas de *Indexed Sequential Access Method* (Método de Acceso Secuencial Indexado) pero con nuevas extensiones (19). En las últimas versiones de Mysql, el motor InnoDB está empezando a reemplazar a este tipo de tablas por su capacidad de ejecutar transacciones de tipo ACID y bloqueo de registros e integridad referencial.

[1] mSQL o Mini SQL es un cliente/servidor de bases de datos ligero de Hughes Technologies.

[2] Se trata de un método para almacenar información a la que se pueda acceder rápidamente. ISAM fue desarrollado originalmente por IBM.

La elección es un tema delicado ya que hay que conseguir la mejor relación de calidad acorde con nuestra aplicación, obviamente si necesitamos transacciones, claves foráneas y bloqueos tendremos que escoger InnoDB por el contrario escogeremos MyISAM en aquellos casos en los que predominen las consultas SELECT a la base de datos (un gran número de páginas webs).

Algunas de las características principales de MySQL son:

Interioridades y portabilidad

- Escrito en C y en C++.
- Funciona en diferentes plataformas.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL (20).
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado en aplicaciones

autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Seguridad

Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor. (17)

Escalabilidad y límites

Soporte a grandes bases de datos. Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT. (21)

Localización

El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas. Soporte completo para distintos conjuntos de caracteres, incluyendo latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas. El soporte para Unicode está disponible. (22)

1.5.3 FireBird

Firebird se deriva del código fuente de InterBase 6.0, de Borland. Es de código abierto y no tiene licencias duales. Tanto si se usa en aplicaciones comerciales o de código abierto, ¡es totalmente LIBRE! (23)

La tecnología de Firebird ha estado en uso por 20 años, lo que lo hace un producto muy estable y maduro.

Firebird es un poderoso y completo RDBMS. Puede manejar bases de datos desde solo unos cuantos KB hasta muchos Gigabytes con muy buen desempeño y prácticamente libre de mantenimiento (23). Sus principales características son:

- Completo soporte para Procedimientos Almacenados y Disparadores.

- Transacciones 100% ACID.
- Integridad Referencial.
- Arquitectura multi-generacional.
- Bajo consumo de recursos.
- Completo lenguaje interno para procedimientos almacenados y disparadores (PSQL).
- Soporte para Funciones Externas (UDFs).
- Poca o ninguna necesidad de DBAs especializados.
- Prácticamente no requiere configuración ya que la configuración por defecto que trae es muy buena tanto para entornos empresariales como para uso local.
- Gran comunidad y muchos sitios donde podes encontrar excelente soporte gratuito.
- Docenas de herramientas de terceros, como herramientas de administración gráficas, herramientas de replicación, etc.
- Escritura segura - recuperación rápida, ¡sin requerir logs de transacciones!
- Muchas formas de acceder a tu base de datos: nativo/API, drivers dbExpress, ODBC, OLEDB, proveedor .Net, driver JDBC, módulo Python, PHP, Perl, etc.
- Soporte nativo para todos los principales sistemas operativos, incluyendo Windows, Linux, Solaris, MacOS.
- Copias de seguridad incrementales.
- Disponibilidad de binarios en arquitectura de 64bits.

1.5.4 Apache Derby

La Fundación Apache lanzó "Apache Derby", la base de datos basada en Java. "Derby" es una copia de la base de datos relacional de IBM "Cloudscape" que el gigante azul aportó a la comunidad de código

abierto. IBM aportó el código a Apache bajo el otorgamiento de licencia de contribuidor corporativo ASF (24).

El proyecto está siendo gestionado por "Apache", que ha sido responsable de inspeccionar el código para asegurarse que se adecue a los estándares de la organización en cuanto a licenciamiento e integridad de código, siendo además responsable de supervisar la formación de la comunidad de desarrollo.

Derby es una base de datos relacional basada en Java, sin soporte de administración, ideal para desarrolladores que no necesitan un sistema de base de datos de clase empresarial (25).

Derby apunta al 30% de soluciones que no requieren una base de datos empresarial, por ejemplo, aplicaciones de pequeños sitios web, sistemas de punto de ventas, registros y repositorios locales, pequeñas aplicaciones.

1.6 Comparación entre Sistemas de Bases de Datos Libres

En la sección siguiente se mostrarán las características más importantes que tienen los sistemas Gestores de Bases de Datos MySQL y PostgreSQL. No se analizarán Apache Derby ya que es de propósito específico además que no se recomienda para grandes sistemas. Tampoco se analizará Firebird ya que su presencia en entornos empresariales no supera la de PostgreSQL ni MySQL.

1.6.1 MySQL vs PostgreSQL

Elegir entre MySQL y PostgreSQL es una decisión la cual muchos deben hacer cuando a bases de datos relacionales libres se refiere. Ambos servidores son soluciones comprobadas al paso del tiempo y que compiten fuertemente con la base de datos de software propietario. **MySQL** ha sido durante mucho tiempo la más rápida pero es la que cuenta con menos funciones de los dos sistemas de bases de datos, mientras que **PostgreSQL** es un sistema de base de datos más denso, la cual a menudo se describe como la versión de código abierto de Oracle.

La Tabla 1 muestra una comparación entre estos 2 Sistemas Gestores de Bases de Datos en cuanto a características usadas por aplicaciones de mediana o gran envergadura, y para uso empresarial, lo que ilustrará como se encuentran estos SGBD en cuanto a la implementación de funcionalidades que poseen sus competidores comerciales.

Características	Mysql	PostgreSQL
MVCC[3]	Si	Si
Bloqueo a nivel de fila	Si	Si
Máximo tamaño de la Base de Datos	Ilimitado*	Ilimitado*
Máximo tamaño de una Tabla	Ilimitado*	Ilimitado*
Máximas filas en una Tabla	Ilimitado*	Ilimitado*
Máximas columnas en una Tabla	~ 1000 columnas (InnoBD)	~ 1600 o más, depende del tipo de datos de las columnas
Cantidad de índices en una Tabla	Ilimitado	Ilimitado
Almacena la información de transacciones en el mismo archivo de datos	No	No
Herencia entre tablas	No	Si
Dominios	No	Si
Particionado de Tablas	No	Si(Básico)
Tablas Temporales	Si	Si
Variedades de Funciones	SI	Si

[3] Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control o MVCC, es la tecnología para evitar bloqueos innecesarios de tablas en una base de datos.

*Limitado solamente por el espacio disponible.

Procedimientos almacenados de varios idiomas	No	Si
Manejo de excepciones en los procedimientos almacenados	Si (declarando manejadores para cada tipo de excepción)	Si
Soporte nativo de para SSL[4]	Si	Si
Múltiples métodos de autenticación	No	Si
Índices Únicos	Si	Si
Índices Parciales	Si	Si
Múltiples tipos de almacenamiento de índices	Si	Si
Soporte para ANSI-SQL 92/99	SI	Si
Crear tipos definidos por el usuario	No	Si
Crear operadores definidos por el usuario	No	Si
Escribe antes de registrar los logs (importante para la recuperación y seguimiento de los logs)	SI	Si
Tablespaces	No	Si

[4] El protocolo SSL es un sistema diseñado para el intercambiado de datos encriptados entre el servidor y el cliente con un algoritmo de cifrado simétrico.

Software para replicación asincrónica de código abierto	SI	Si
Online/Hot Backups	SI	Si
Cumple plenamente con ACID	SI(InnoDB)	Si
Soporte para objetos binarios grandes	SI	Si
Soporte para UTF-8	SI	Si
Definir conjuntos de caracteres para cada columna caracteres para cada columna	SI	No
Llaves Foráneas	SI(InnoDB)	Si
Verificación de restricciones	SI(InnoDB)	Si
Múltiples niveles de aislamiento de transacciones	SI	Si
Soporte nativo para GIS[5].	Si	Si (PostGIS)
Monitoreo de Base De Datos	Si	Si
Capacidad de consulta de bases de datos en otros servidores locales o remotos	SI	Si (DBLink)

[5] Sistema de Información Geográfica son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones

Actualizaciones regulares	Si	Si Cambios mayores una vez al año, con frecuentes cambios menores.

Tabla 1: Características de MySQL, PostgreSQL

La tabla anterior muestra que PostgreSQL en cuanto a funcionalidades es más completo que MySQL. Por la amplia gama de funcionalidades que implementa PostgreSQL es reconocido como uno de los mejores motores de bases de datos del mundo.

La Tabla 2 muestra algunas funcionalidades que están presentes en Oracle que el SIGESC 171 utiliza, que están presentes en MySQL o PostgreSQL.

Características de Oracle	Mysql	PostgreSQL
Múltiple control de Acceso	No	Si
Secuencias	No	Si
Lenguaje Procedural	Si	Si
Transaccional	Si(InnoDB)	SI
Paquetes	No	No
Esquemas	No	Si
Tratamiento de Excepciones	Si	Si
Tablespaces	No	Si
Triggers	Si	Si

Roles	No	Si
Índices Únicos	Si	Si
Soporte para objetos binarios grandes	Si	Si
Soporte para múltiples codificaciones	Si	Si
Actualizaciones regulares	SI	Si

Tabla 2: Funcionalidades de Oracle presentes en MySQL o PostgreSQL.

Se puede apreciar que una vez más PostgreSQL supera a MySQL en cuanto a funcionalidades similares con Oracle.

Rendimiento

Los Sistemas de Bases de Datos pueden ser optimizados de acuerdo con el entorno en el que corren. Por lo tanto, es muy difícil dar una comparación precisa en el rendimiento, sin prestar atención a la configuración y el medio ambiente. PostgreSQL así como MySQL emplean diversas tecnologías para mejorar el rendimiento en el nivel básico:

Comienzos

MySQL comenzó su desarrollo con un enfoque en la velocidad, mientras que PostgreSQL comenzó a desarrollar con un enfoque sobre las características y normas. Por lo tanto, MySQL es a menudo considerado como el más rápido de los dos. La configuración por defecto de PostgreSQL fue diseñada para ejecutarse en sistemas con poca memoria.

Raw Speed

El motor MyISAM de MySQL funciona más rápido que PostgreSQL sobre las consultas simples y cuando la concurrencia es baja o sigue ciertos patrones (por ejemplo, las inserciones que se realizan en tablas optimizadas y sin bloqueos).

El costo de la velocidad del motor MyISAM viene de no brindar soporta a las transacciones, **llaves foráneas,** y no ofrece durabilidad garantizada en los datos.

Compresión de los datos

PostgreSQL puede comprimir y descomprimir sus datos sobre la marcha con un rápido sistema de compresión para encajar más datos en un espacio de disco asignado. La ventaja de los **datos comprimidos**, además de ahorrar espacio en disco, es que la lectura de datos tarda menos, por lo que lee datos con mayor rapidez. (26)

Hasta la versión 5.1 de MySQL sus motores de almacenamiento de alto rendimiento no soportan compresión sobre la marcha.

La versión de MySQL 6.0 tendrá soporte de compresión sobre la marcha con su nuevo motor de almacenamiento Falcon:

> Los datos almacenados en las tablas de Falcon están comprimidos en el disco, pero se almacena en un formato sin comprimir en la memoria. La compresión se produce automáticamente cuando los datos se guardan al disco.

Con InnoDB instalado, MySQL 5.1 soporta compresión sobre la marcha de tablas InnoDB:

Velocidad

PostgreSQL proporciona características que pueden conducir a un rendimiento más rápido en algunas consultas:

- Indexación parcial.
- TOAST de compresión de datos.
- Una mayor asignación de memoria de forma predeterminada en las últimas versiones del sistema.
- Mejora de la gestión de caché en las versiones 8.1 y 8.2.

MySQL también soporta la indexación parcial utilizando el motor InnoDB, pero no con el motor MyISAM. Incluso cuando se utiliza el motor InnoDB, sin embargo, las tablas del sistema utiliza el motor MyISAM.

Si bien es cierto que con una instalación por defecto de MySQL, este por lo regular le gana a PostgreSQL en muchos parámetros de rendimiento, algunos benchmarks que muestran un mayor rendimiento de MySQL ante PostgreSQL tienden a sufrir una serie de problemas:

- No era raro ver a un servidor MySQL afinado para **rendimiento** el cual se enfrentara contra un servidor PostgreSQL con la configuración por defecto y sin afinar.
- Muchos de los benchmarks a menudo agrupan transacciones modales poco realistas que no reflejan el comportamiento de las aplicaciones en el mundo real, lo que lleva a un desajuste en el número de operaciones discretas que se están llevando a cabo en un tiempo y las relaciones entre ellos de una base de datos a la siguiente.
- Las “transacciones” de MyISAM que no cumplen con el estándar ACID a menudo son comparadas contra PostgreSQL ejecutando transacciones compatibles con ACID.
- Si bien esto resultaba a menudo en un mayor rendimiento en los benchmarks, también involucraba sistemas de pruebas haciendo cosas muy diferentes.
- Si la **integridad transaccional** es de suma importancia, mayor rendimiento sin garantía en la integridad transaccional no son una opción en absoluto.
- El rendimiento de MyISAM ha sido optimizado para un solo un usuario. Esto significó una victoria para MySQL usando MyISAM en muchos puntos de referencia.
- En virtud del uso, sin embargo, con muchos usuarios concurrentes, la utilización del control de bloqueos de tablas de MyISAM tiene un dramático efecto negativo en el rendimiento.

Concurrencia

PostgreSQL escala mucho mejor, tanto en términos de la utilización de un hardware de alto rendimiento, y al hacer frente a la concurrencia. MySQL, por otra parte, se centra en tecnologías comunes de bajo rendimiento y el uso de hardware básico.

Cumplimiento de ACID

ACID (Atomicidad, Coherencia, Aislamiento y Durabilidad), este modelo se utiliza para evaluar la integridad de los datos a través de los sistemas de gestión de bases de datos. La mayoría de sistemas de bases de datos cumplen con el estándar ACID mediante el uso de las transacciones.

PostgreSQL es plenamente compatible con ACID, mientras que el motor de almacenamiento InnoDB de MySQL proporciona cumplimento de ACID a nivel del motor.

MySQL Server (versión 3.23-max y todas las versiones 4.0 y superiores) soportan las transacciones con los motores de almacenamiento transaccionales InnoDB y BDB.

InnoDB proporciona pleno cumplimiento de ACID.

MySQL Cluster también es un motor de almacenamiento de transacciones seguro.

Para utilizar el motor compatible con ACID en MySQL por defecto, basta con establecer **default-storage-engine**=innodb en su archivo de configuración.

Innobase, la empresa que desarrolló InnoDB, fue adquirida por Oracle en octubre de 2005. Oracle y MySQL AB firmaron un contrato para extender la concesión de licencias para el motor InnoDB en 2006, pero algunos temen que esa licencia cuando finaliza el período la competencia comercial entre Oracle (propietario del motor de almacenamiento InnoDB) y Sun (que compró MySQL AB en febrero de 2008) puede conducir a la futura concesión de licencias y costos para los usuarios de MySQL.

Características

MySQL y PostgreSQL tienen un impresionante conjunto de características que aumentan la integridad de los datos, funcionalidad y rendimiento. Las características incluidas en una base de datos pueden ayudar a mejorar el rendimiento, la funcionalidad, o la estabilidad.

Limitaciones

Tanto MySQL y PostgreSQL soportan Not-Null, Unique, Llave primaria y llaves foráneas. Sin embargo MySQL no soporta comprobación de limitación (Check constraint) mientras que PostgreSQL lo ha soportado durante mucho tiempo.

Tablas InnoDB soportan la comprobación de llaves foráneas… Para otros motores de almacenamiento, MySQL Server analiza y hace caso omiso de FOREIGN KEY y REFERENCES en la sintaxis de CREATE TABLE.

Procedimientos Almacenados

Tanto MySQL y PostgreSQL soportan procedimientos almacenados. El primer lenguaje de consulta para PostgreSQL, PL/PgSQL, es similar a la de Oracle PL/SQL. PostgreSQL también soporta procedimientos almacenados en muchos otros lenguajes, entre ellos Python, Perl, TCL, Java y C - en particular la norma ISO SQL:2003 PSM.

MySQL sigue el SQL:2003 para la sintaxis de rutinas almacenadas, que también es utilizado por IBM DB2.

Disparadores

Tanto MySQL y PostgreSQL soportan disparadores. Un disparador PostgreSQL puede ejecutar cualquier función definida por el usuario desde cualquiera de sus lenguajes de procedimiento, no sólo PL/pgsql.

Los disparadores de MySQL son activados solamente por comandos SQL. Estos no son activados por cambios en las tablas realizados por APIs que no transmiten las declaraciones de SQL al servidor MySQL, en particular, no son activadas por las actualizaciones hechas utilizando el NDB API.

PostgreSQL también soporta las “reglas”, que permiten operar en el árbol de sintaxis de la consulta, y puede hacer algunas operaciones más simplificadas que tradicionalmente son realizadas por disparadores (27).

La sintaxis para la definición de los disparadores en PostgreSQL no es tan sencilla como en MySQL. En PostgreSQL se requiere una definición de una función con la devolución del tipo de datos específico.

Replicación y Alta Disponibilidad (HA)

La replicación es la capacidad de un sistema de gestión de base de datos de duplicar sus datos almacenados a efectos de brindar copias de seguridad y una manera de prevenir la inactividad de la base de datos de inactividad (28). Ambos PostgreSQL y MySQL soportan replicación:

PostgreSQL es modular por su diseño, y la replicación no está en el núcleo. Hay varios paquetes que permiten la replicación en PostgreSQL:

- PGCluster
- Slony-I
- DBBalancer
- pgpool
- PostgreSQL table comparator
- SkyTools
- Sequoia
- Bucardo

Es un error común pensar que estos “paquetes de terceros” de alguna manera son menos bien integrados. Slony, por ejemplo, fue diseñado y construido por Jan Weick, un miembro del equipo del núcleo de PostgreSQL, y tiene otros miembros de la comunidad PostgreSQL que participan en su diseño continuo y mantenimiento.

Sin embargo, Slony es considerablemente más lento y utiliza más recursos que MySQL y su replicación incorporada, ya que utiliza SQL y disparadores en lugar de un registro binario de envío para replicar los datos a través de los servidores.

Lo cual lo puede hacer menos adecuado para grandes instalaciones de clusters con necesidades de alto rendimiento. Recientemente, el equipo del núcleo de PostgreSQL anuncio que la replicación básica se ha previsto como parte de la liberación 8.4.

MySQL brinda soporte para replicación. A partir de la versión 5.1, MySQL soporta dos formas de replicación; replicación basada en declaración (SBR) y replicación basada en la fila (RBR). SBR recolecta las consultas SQL que realizan cambios a la base de datos en un registro binario a los cuales los servidores esclavos se conectan para copiar sus cambios (29).

A diferencia RBR registra los cambios incrementales a las filas en el registro binario que luego son aplicados a los esclavos. Algunos motores de almacenamiento, tales como NDB y Falcon sólo soportan la replicación usando este nuevo formato basado en la fila.

Tipos de datos

PostgreSQL no tiene un tipo de dato entero sin signo, pero tiene mucho más soporte de tipos de datos en varios aspectos: el cumplimiento de las normas, la lógica fundamental del tipo de datos BOOLEAN, mecanismos de tipo de datos definidos por el usuario, tipos nativos y contribuidos.

Arreglos de cualquier tipo básico nativo o definido, tipo enumerativo, o tipo compuestos se pueden crear. Arreglos de dominios no son soportados todavía.

Subconsultas

Tanto MySQL y PostgreSQL soportan subconsultas, pero en MySQL algunas formas puede ser un gran impacto en el rendimiento. Esto será corregido en la versión 6.0.

Indexación Avanzada

Métodos avanzados de indexación permiten a los sistemas de bases de datos optimizar las consultas para lograr un mayor rendimiento.

- Índices hash: Sólo InnoDB y MEMORY soporta índices hash. PostgreSQL soporta índices hash, aunque nunca son más rápido que los índices de árbol-Binario.
- Índices múltiples: MySQL soporta múltiples índices por tabla y consultas desde 5.0. PostgreSQL soporta múltiples índices por consulta.

- Full-Text Índices: MySQL viene con búsqueda de texto completo, pero sólo se puede ejecutar sobre el (transacción no segura) motor de almacenamiento MyISAM.
- Un agregado de terceros para MySQL, Sphinx Fulltext Search Engine permite el soporte a búsquedas de texto completo sobre tablas InnoDB. El texto de la indexación integrada no puede ser más de 255 caracteres esto significa que usted no puede garantizar una única columna de texto de más de 255 caracteres. PostgreSQL 8.2 tiene búsqueda de texto completo en el modulo tsearch2.
- PostgreSQL 8.3 integra tsearch2 en el núcleo: TSearch2, la herramienta de búsqueda texto completo de vanguardia, se ha integrado plenamente en el núcleo de código, y también tiene un API más limpia.
- Índices parciales: MySQL no es compatible con los índices parciales. PostgreSQL soporta índices parciales:

  Un índice parcial es un índice construido sobre un subconjunto de una tabla, el subconjunto se define por una expresión condicional (llamado el predicado del índice parcial). El índice contiene entradas de la tabla sólo aquellos registros que satisfacen el predicado. Los índices parciales son una función especializada, pero hay varias situaciones en las que son útiles.

Una de las principales razones para la utilización de un índice parcial es evitar la indexación de los valores comunes. Ya que una consulta en busca de un valor común (uno que representa más de unos pocos por ciento de todas las filas de tabla) no utilizara el índice de todos modos, no hay ningún motivo para mantener esas filas en el índice en absoluto.

Esto reduce el tamaño del índice, lo que acelerará las consultas que si utilizan el índice. Así mismo, acelerara muchas de las operaciones de actualización de la tabla porque el índice no necesita ser actualizado en todos los casos.

Motores de Almacenamiento de Datos

Los Motores de almacenamiento de Datos tienen en cuenta el medio que se está utilizando (para la mayoría de los propósitos, las bases de datos se almacenan en los discos para proporcionar la persistencia de datos) para maximizar el rendimiento de lectura/escritura.

PostgreSQL soporta un motor, por defecto su sistema de almacenamiento es Postgres (Postgres Storage System). Hay una serie de formas de aumentar el rendimiento de PostgreSQL.

Para los datos no críticos, puede poner su directorio de almacenamiento en un disco RAM LINK. Esto, por supuesto, plantea la pregunta, de por qué usted quiere ponerlo en una base de datos de todos modos. Hay DSMS (Sistemas de Gestión de flujo de datos - Data Stream Management Systems) diseñados específicamente para manejar los datos transitorios en una forma muy eficiente.

MySQL 5.1 soporta nativamente 9 motores de almacenamiento.

MyISAM

- InnoDB
- NDB Cluster
- MERGE
- MEMORY (HEAP)
- FEDERATED
- ARCHIVE
- CSV
- BLACKHOLE

InnoDB es desarrollado por la empresa externa InnoBase, que ha sido adquirida por Oracle. Conserva su lugar en las distribuciones estándar de MySQL como el principal motor transaccional. MySQL tiene previsto introducir nuevos motores de nombre María y Falcon en una próxima versión 6.x.

Entre las nuevas características figura la recuperación. Se espera reemplazar y completar los motores MyISAM e InnoDB, respectivamente.

Hay varios motores de almacenamiento desarrollados externamente, algunos de los más populares son:

- SolidDB
- NitroEDB
- BrightHouse

MySQL tiene motores de almacenamiento a la medida y de la comunidad en desarrollo:

- PrimeBase XT
- FederatedODBC
- FederatedX
- IBM DB2
- memcached

En algunas distribuciones, el motor de almacenamiento por defecto es MyISAM, que no es seguro para las transacciones. Ajustar los valores para configurar el motor transaccional como InnoDB, es sin embargo, trivial.

Licencias

PostgreSQL viene con un estilo de licencia BSD, que se inscribe en la definición de Software Libre y Código Abierto, y se ajusta tanto a la *Debian Free Software Guidelines* y al estándar Copyfree.

El código fuente de MySQL está disponible bajo los términos de la Licencia Pública General de GNU, que también se inscribe en las definiciones de Software Libre y Código Abierto y se ajusta a la Debian Free Software Guidelines (pero no a la Copyfree Standard).

También está disponible bajo un acuerdo de licencia de propietaria, que es normalmente destinado a ser utilizados por aquellos que desean incorporar código de MySQL sin tener que liberar el código fuente para toda la aplicación.

En términos prácticos, esto significa que MySQL se puede distribuir con o sin código fuente, como PostgreSQL, pero para no distribuir el código fuente en el caso de MySQL se requiere el pago de MySQL AB para una licencia comercial de MySQL.

Incluso la biblioteca cliente de MySQL es GPL (no LGPL), lo que significa que el uso (y, por tanto, enlace a) la biblioteca cliente de MySQL el programa en sí debe ser GPL o debe tener una licencia comercial de MySQL AB.

Desarrollo

PostgreSQL no es controlada por una sola empresa, sino que se basa en una comunidad global de desarrolladores y empresas para desarrollarlo.

MySQL es propiedad y está patrocinado por una sola empresa con fines de lucro, la empresa sueca MySQL AB, que posee los derechos de autor a la mayoría del código.

El 16 de enero de 2008 MySQL AB anunció un acuerdo para ser adquirida por Sun Microsystems, por aproximadamente US $1 mil millones.

MySQL es PRODUCTO de código abierto.

PostgreSQL es proyecto de código abierto.

Muchas son las características que ofrece PostgreSQL, su estabilidad, escalabilidad, variedad de tipos de datos soportados, rendimiento excelente unido a que posee una licencia que permite su uso para cualquier fin lo convierte en un poderoso sistema manejador de bases de datos, que tiene la fama de ser la base de datos de código abierto (Open Source) más avanzada del mundo. PostgreSQL se ha ganado la admiración y el respeto de sus usuarios, así como el reconocimiento de la industria (ganador del Linux New Media Award for Best Database System y 3 veces ganador del The Linux Journal Editors' Choice

Award for best DBMS) (30). Por antes expuesto se utilizará PostgreSQL como nuevo SGBD para el SIGESC 171.

Conclusiones del capítulo

Después de realizado el estudio de los conceptos más importantes y comparar algunos SGBD libres, se definió que el SGBD PostgreSQL cumple con todas las características necesarias para llevar a cabo la migración del diseño y la administración del SIGESC 171.

2 La Migración

Este capítulo abarca:

- Migración de la estructura de la Base de Datos
- Creación de los Objetos de la Base de Datos
- Creación de Políticas de Salva y Recuperación

CAPÍTULO II
La Migración

2.1 Introducción

El proceso de migración puede aparecer complicado. Son muchas las diferencias entre cada Sistema Gestor de Bases de Datos. Las palabras y las terminologías usadas para describir la arquitectura de Oracle a menudo tienen significados totalmente diferentes en PostgreSQL. Desde la perspectiva de los desarrolladores de aplicaciones, Oracle y PostgreSQL gestionan los datos de forma similar. Las diferencias internas entre Oracle y PostgreSQL son significativas, pero si son manejadas apropiadamente, tienen impactos mínimos sobre la aplicación migrada.

En este capítulo se detallan las especificaciones del funcionamiento y administración de la Base de Datos del SIGESC 171. Se realizará además una clasificación de los elementos a migrar así como los pasos para realizar la migración al SGBD seleccionado.

Para realizar el proceso de migración se cuenta con varios ficheros SQL generados desde Oracle:

secuencias.sql: Este script tiene las sentencias de creación de todas las secuencias del SIGESC 171.

tablas.sql: En este script están las sentencias de creación de todas las tablas del SIGESC 171.

paquetes.sql: Sentencias de creación de los paquetes del SIGESC 171.

usuarios.sql: Sentencia de creación de los usuarios y roles iniciales con los que contará la base de datos del SIGESC 171.

Se cuenta además con un servidor PostgreSQL con un usuario creado llamado *sigesc* con privilegios de administración sobre el servidor de bases de datos.

2.2 Pasos para la migración

Para realizar el proceso de migración se siguieron los siguientes pasos:

- Creación de las Estructuras de Organización Físicas y Lógicas. Creación de la Base de Datos.

- Creación de los Objetos de la Base de Datos.
- Seguridad del Servidor (usuarios y roles).
- Creación de políticas de Salva y Recuperación.
- Creación de políticas de mantenimiento de la Base de Datos.

2.3 Creación de las Estructuras de Organización Físicas y Lógicas. Creación de la Base de Datos

2.3.1 Tablespaces (Espacios de Tablas)

En Oracle toda base de datos tiene uno o más archivos de datos físicos asociados. En estos archivos de datos es donde se almacenan físicamente todos los datos de la base de datos. Para contribuir a una mejor administración de estos archivos de datos, se agrupan en unos objetos de base de datos denominados *tablespaces* (espacios de tablas).

Un *tablespace* es una colección de uno o más archivos de datos. Las tablas, índices y otros objetos de base de datos se colocan dentro de estos tablespaces. (31)

Los tablespaces en PostgreSQL permiten al Administrador de Base de Datos (DBA[6]) definir localizaciones en el sistema de ficheros donde los archivos que representan objetos en la base de datos pueden ser almacenados.

Mediante el uso de tablespaces, un DBA puede controlar la estructura en el disco de la instalación de PostgreSQL. Esto es útil en al menos dos formas:

En primer lugar, si la partición o volumen en el que la base de datos se ha inicializado se queda sin espacio y no puede extenderse, se puede crear un tablespace en una partición diferente y se utiliza hasta que el sistema pueda ser reconfigurado. (31)

[6] Database Administrador (Administrador de Bases de Datos)

En segundo lugar, los tablespaces permiten la utilización de los patrones de uso de los objetos de la base de datos para optimizar el rendimiento. Por ejemplo, un índice que es muy usado puede ser colocado en un disco altamente disponible. Al mismo tiempo una tabla que almacena datos que rara vez se utiliza o no participa en operaciones críticas, podría estar almacenada en un disco más lento y menos costoso. (31)

Existen varias razones que justifican este modo de organización de las tablas en tablespaces:

- Permiten distribuir a nivel físico los distintos objetos de las aplicaciones.
- Un tablespace puede quedarse *offline* debido a un fallo de disco, permitiendo que el Sistema Gestor de Bases de Datos continúe funcionando con el resto.
- Como son una unidad física de almacenamiento, pueden usarse para aislar completamente los datos de diferentes aplicaciones.

Para la creación de un tablespace se utiliza en comando *CREATE TABLESPACE*, por ejemplo:

```
CREATE TABLESPACE ts_sigesc LOCATION '/mnt/almacenamiento1/sigesc171/data';
```

La localización del tablespace debe existir y el propietario de ese directorio debe ser el usuario del sistema postgres. Posteriormente, todos los objetos creados dentro del tablespace se almacenarán en archivos bajo ese directorio.

Tablas, índices y bases de datos enteras pueden ser asignadas a un tablespace. Para hacer esto, un usuario con el privilegio de *CREATE* sobre un tablespace dado debe pasar el nombre del tablespace como un parámetro al comando. Por ejemplo:

```
CREATE TABLE dllamada(i int) TABLESPACE ts_sigesc;
```

Para remover un tablespace se utiliza en comando DROP TABLESPACE.

Para determinar el conjunto de tablespaces existentes, se consulta en catálogo del sistema *pg_tablespace*, por ejemplo:

```
SELECT spcname FROM pg_tablespace;
```

Definición de los tablespaces

En el SIGESC 171 se definieron tres tablespaces:

- *ts_sigesc_sigesc*: Utilizado para el almacenamiento de todos los objetos de la base de datos.
- *ts_sigesc_indices*: Utilizado para el almacenamiento de los índices.
- *ts_sigesc_tmp:* Este tablespace utilizará como temporal para que PostgreSQL lo utilice para la colocación temporal de tablas e índices, así como los archivos temporales que usa para fines tales como la ordenación de grandes conjuntos de datos.

Una vez dentro del servidor PostgreSQL como el usuario sigesc, el proceso de creación de los tablespaces para el SIGESC 171 sería:

```
CREATE TABLESPACE ts_sigesc LOCATION '/ruta/almacenamiento1/data';
CREATE TABLESPACE ts_sigesc_indices LOCATION '/ruta/almacenamiento2/data';
CREATE TABLESPACE ts_sigesc _tmp LOCATION '/ruta/almacenamiento3/data';
```

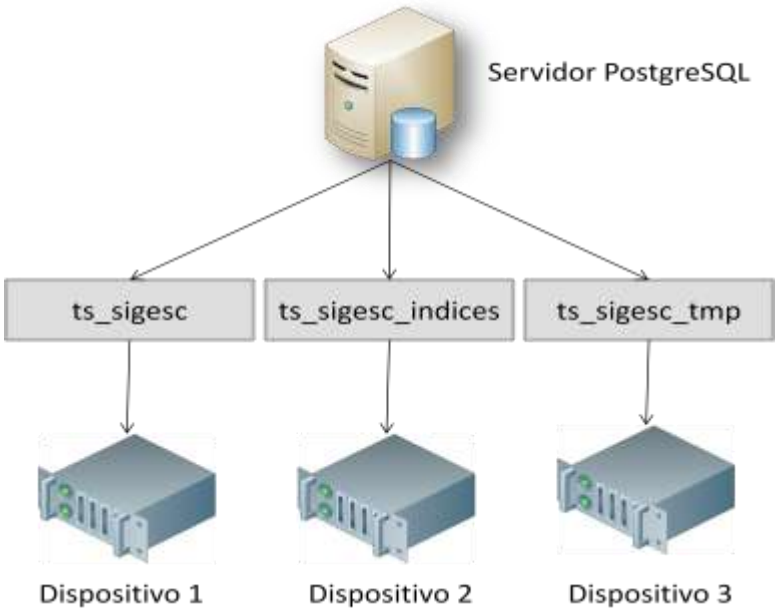

Figura 1: Representación de los Tablespaces

Para el aprovechamiento por parte del servidor PostgreSQL del tablespace *ts_sigesc_tmp*, se necesitará realizar una modificación en unos de los archivos de configuración del servidor. El archivo de configuración a modificar es el *postgresql.conf*, el cual se encuentra en el directorio donde se instaló PostgreSQL.

Una vez encontrado el fichero se modificará el parámetro *temp_tablespaces* como se muestra a continuación:

```
temp_tablespaces = 'ts_sigesc_tmp'
```

Para que surja efecto el cambio realizado se procederá a reiniciar el servidor PostgreSQL.

Como se muestra en la Figura 1, cada uno de estos tablespaces se almacenarán en discos diferentes, esto le brindará al sistema ventajas de **disponibilidad**, **velocidad de respuesta** y **seguridad de la información**.

Condiciona la **disponibilidad** de la información del sistema porque los tablespaces pueden quedar fuera de servicio (offline) por falta de capacidad de almacenamiento, por ejemplo: El tablespaces de los índices puede llenarse y quedar fuera de servicio que no afecta al de los datos y por tanto se puede continuar accediendo a la información.

Condiciona la **velocidad de respuesta** porque al estar la información en dispositivos diferentes las operaciones de entrada/salida sobre datos almacenados en diferentes tablespaces se realzarán con recursos físicos diferentes, la cantidad de información será menos en cada tablespaces algo que también brindara mayor velocidad de ejecución.

Condiciona la **seguridad** porque mejora las operaciones de backup y recuperación de los datos, ya que estas operaciones pueden ser hechas a nivel de tablespaces y dispositivos. Si alguno de los dispositivos se daña físicamente esto no afecta el funcionamiento del sistema ni en la disponibilidad del resto de los datos.

Una vez creadas las estructuras físicas sobre las que se distribuirá nuestra base de datos, se está en condiciones de pasar a la creación de la base de datos del SIGESC 171.

2.3.2 Creación de la Base de Datos

En PostgreSQL una base de datos es una colección de objetos SQL (objetos de base de datos). Generalmente, cada objeto de base de datos (tablas, funciones, etc.) pertenece a una y solo una base de datos.

Al conjunto de base de datos que maneja un servidor PostgreSQL se le denomina *cluster*.

Las bases de datos son creadas con el comando CREATE DATABASE:

```
CREATE DATABASE name
    [ [ WITH ] [ OWNER [=] dbowner ]
           [ TEMPLATE [=] template ]
           [ ENCODING [=] encoding ]
           [ TABLESPACE [=] tablespace ]
           [ CONNECTION LIMIT [=] connlimit ] ]
```

name: Nombre de la nueva base de datos.

dbowner: El nombre del usuario de base de datos que será propietario de la nueva base de datos. Si es omitido se tomará el nombre del usuario que ejecuta el comando.

template: El nombre de la base de datos plantilla desde la cual se creará la nueva base de datos. Si es omitido, se usará la base de datos plantilla por defecto (*template1*).

tablespace: El nombre del tablespace que será asociado a la nueva base de datos. Si es omitido, se usará el tablespace de la base de datos plantilla. Este tablespace será el tablespace por defecto de los objetos creados en la base de datos.

encoding: Es la codificación del conjunto de caracteres que se utilizará en la nueva base de datos. Se puede especificar mediante una cadena (ejemplo: 'utf-8'). Si se omite, se utilizará por defecto la codificación de la base de datos plantilla.

connlimit: Cantidad de conexiones concurrentes que pueden ser realizadas a la nueva base de datos.

Creación de la base de datos

El nombre que se utilizó para la base de datos del SIGESC 171 fue *dbsigesc* y la codificación utilizada en el servidor Oracle fue *UTF-8,* la cual se debe mantener en la creación de la base de datos en el servidor PostgreSQL.

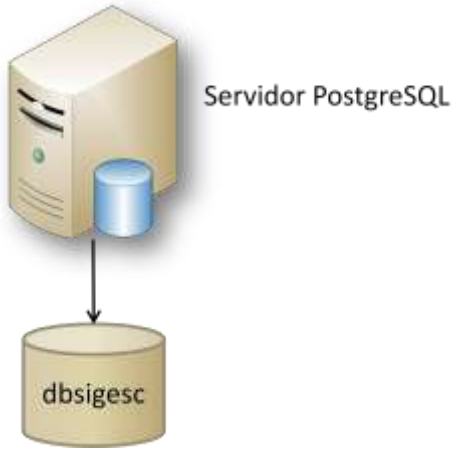

Figura 2: Base de datos DBSIGESC

El usuario propietario de la base de datos y de todos los objetos creados en ella será el usuario *sigesc*.

Una vez definido el propietario, codificación y nombre de la base de datos, se procederá a la creación de esta en PostgreSQL:

```
CREATE DATABASE dbsigesc
  WITH ENCODING= 'UTF-8'
  OWNER=sigesc
  TABLESPACE=ts_sigesc;
```

Nótese que se ha especificado como tablespace por defecto el nombrado *ts_sigesc*, al cual por defecto irán todos los objetos que se crearán dentro de esa base de datos.

Para que los nuevos objetos creados en la base de datos usen otro tablespace, el nombre del tablespace debe ser especificado explícitamente en sus respectivos comandos de creación.

Una vez creada la base de datos se accede a esta mediante:

- Ejecución de la terminal interactiva de PostgreSQL, llamada *psql,* la cual permite de forma interactiva entrar, editar y ejecutar comandos SQL.

- Usando una herramienta gráfica tal como el pgAdmin o cualquier suite que soporte ODBC para crear manipular bases de datos.

Para la conexión a la base de datos creada se utilizará la consola del PostgreSQL *psql*, ejecutando el siguiente comando:

```
$ psql –d dbsigesc –U sigesc
```

Se introducirá la contraseña del usuario sigesc para acceder a la base de datos. Si todo marcha correctamente saldrá un mensaje de bienvenida al servidor PostgreSQL y el indicador de que estamos conectados:

```
dbsigesc=#
```

Llegado a este punto, se puede comenzar a realizar la creación de la estructura lógica de la base de datos para el SIGESC 171.

2.3.3 Esquemas

Las bases de datos organizan los objetos relacionados dentro de un *esquema* de base de datos. Por ejemplo, es normal organizar dentro de un esquema de base de datos sencillo todas las tablas y los objetos de base de datos necesarios para soportar una aplicación. La Figura 3 muestra cómo podría ser un esquema de una aplicación.

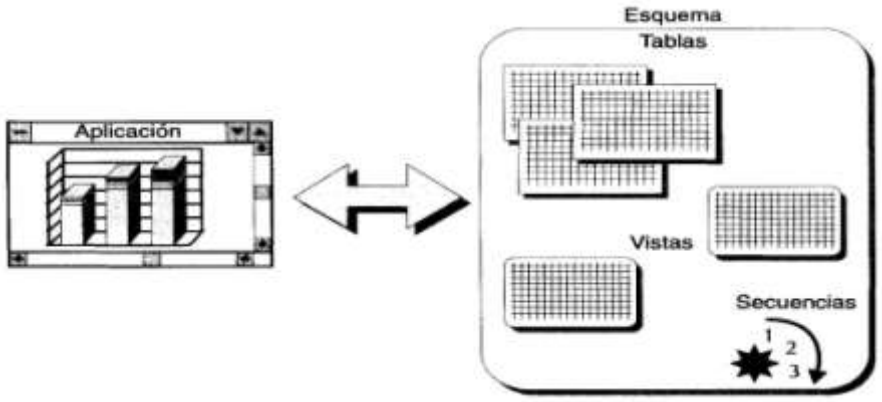

Figura 3: Ejemplo de esquema

Tanto en Oracle como en PostgreSQL los esquemas no organizan físicamente el almacenamiento de los objetos. Por el contrario, los esquemas organizan lógicamente los objetos de base de datos relacionados. En otras palabras, en Oracle y en PostgreSQL la organización lógica de los objetos de base de datos dentro de los esquemas sirve inicialmente para la organización y no tiene nada que ver con el almacenamiento físico de los objetos de base de datos.

Una base de datos en PostgreSQL contiene uno o más esquemas, el cual contiene tablas. Los esquemas pueden contener además otros tipos de objetos, incluyendo tipos definidos por el usuario, funciones y operadores.

Existen varias razones que por las cuales se recomienda el uso de los esquemas:

- Permiten a muchos usuarios usar una base de datos sin interferir unos con otros.
- Organizar los objetos de la base de datos en grupos lógicos los hace más manejable.
- Los datos y objetos de terceras aplicaciones pueden ubicarse en esquemas separados, por lo que los nombres de dichos objetos no entrarían en conflicto con los de nuestra base de datos.

La sentencia de creación de esquemas en PostgreSQL tiene la siguiente estructura:

```
CREATE SCHEMA nombre_esquema [AUTHORIZATION propietario]
```

nombre_esquema: Nombre del esquema que será creado. Si el nombre es omitido, será utilizado el nombre del usuario como nombre del esquema.

propietario: Nombre del usuario que será propietario del esquema. Si es omitido, será por defecto el usuario que ejecuta el comando.

La creación o acceso a los objetos en un esquema, se realiza mediante un nombre, compuesto por el nombre del esquema y el nombre del objeto separado por un punto, ejemplo:

```
sigesc=# CREATE esquema.tabla(i integer);
```

Cada nueva base de datos en PostgreSQL se crea con un esquema especial nombrado "*public*". Por defecto, todos los objetos que se creen en la base de datos irán para ese esquema si NO existe un esquema creado, con el mismo nombre del usuario que se conecta a la base de datos.

Por defecto, un usuario no puede acceder a los objetos en los esquemas en que no es propietario. Para ello, el propietario debe conceder el permiso de USAGE sobre el esquema.

A un usuario se le puede permitir crear objetos en cualquier esquema. Para ello, se le debe conceder el privilegio de CREATE. Por defecto, todos los usuarios tienen los privilegios de USAGE y CREATE sobre el esquema *public.* Esto permite que todos los usuarios sean capaces de conectarse a la base de datos para crear objetos en el esquema *public*. Si no se desea esto, se puede revocar ese privilegio:

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

El primer "*public*" es el esquema, el segundo "*public*" significa "todos los usuarios". El primero es un identificador, el segundo es una palabra clave, por tanto la diferencia en la capitalización.

Definición de los esquemas

La base de datos del SIGESC 171 cuenta con un único esquema llamado SIGESC, en el cual se almacenan todos los datos y objetos de la base de datos que utilizan cada una de las aplicaciones que conforma el sistema.

Una vez dentro del servidor PostgreSQL como usuario *sigesc*, el proceso de creación del esquema para el SIGESC 171 sería:

```
CREATE SCHEMA sigesc
  AUTHORIZATION sigesc;
GRANT ALL ON SCHEMA sigesc TO sigesc;

REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

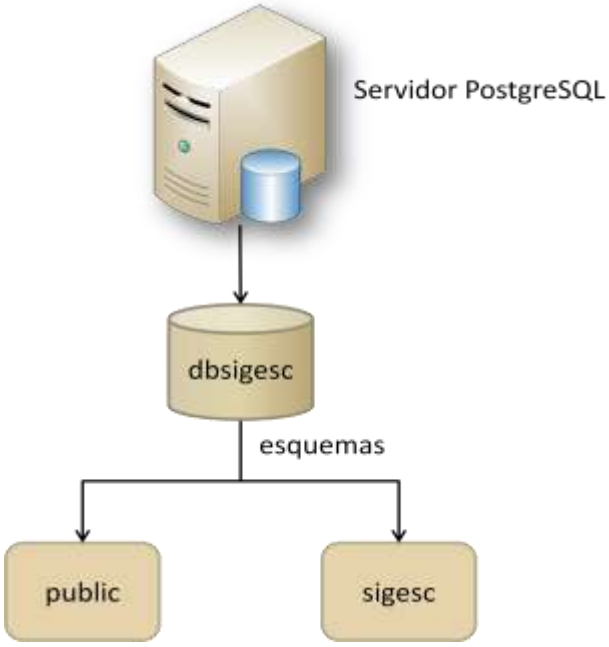

Figura 4: Esquemas en la base de datos DBSIGESC

El usuario sigesc será el propietario del esquema creado y tendrá además todos los permisos sobre este.

El nombre del esquema y el del usuario es el mismo, en este caso *sigesc*, por lo tanto, todos los objetos que se crearán a partir de este momento se colocarán en el esquema *sigesc*, y NO en el esquema *public.*

2.4 Creación de los Objetos de la Base de Datos

Los objetos de la base de datos Oracle (identificadores, tablas, disparadores, secuencias) se pueden migrar fácilmente a PostgreSQL porque ambos Sistemas Gestores de Bases de Datos siguen de cerca el estándar SQL-92 (32).

2.4.1 Identificadores de los objetos de la Base de Datos

La siguiente tabla compara cómo Oracle y PostgreSQL manejan los identificadores de los objetos. En la mayoría de los casos no es necesario cambiar los nombres de los objetos cuando se migra a PostgreSQL.

Oracle	*PostgreSQL*
Identificadores: de 1-30 caracteres. Nombre de bases de datos: > 8 caracteres.	Identificadores: de 1-63 caracteres. Nombre de bases de datos: 1-63 caracteres.
Los identificadores deben comenzar con una letra y contener caracteres alfanuméricos, o los caracteres _, #, y $.	Los identificadores deben comenzar con una letra y contener caracteres alfanuméricos, o los caracteres _, #, y $.
Almacena los identificadores en mayúsculas.	Almacena los identificadores en minúsculas.
Los nombres de los tablespaces deben de ser únicos.	Los nombres de los tablespaces deben de ser únicos.
Los nombres de las columnas deben de ser únicos en las tablas y las vistas.	Los nombres de las columnas deben de ser únicos en las tablas y las vistas.
Los nombres de los índices tienen que ser únicos en el Esquema del usuario.	Los nombres de los índices tienen que ser únicos dentro de cada Esquema de la base de datos.

Tabla 3: Manejo de identificadores de objetos en Oracle y PostgreSQL

El almacenamiento en minúsculas de los identificadores en PostgreSQL es incompatible con el estándar SQL el cual menciona que los identificadores se deben almacenar en mayúsculas.

Identificadores en el SIGESC 171

Como se ha expuesto en la Tabla 3, la longitud máxima de los identificadores en PostgreSQL es mayor que en Oracle, esto garantiza que no se requerirán cambios en ninguno de los identificadores de los objetos en el momento de la generación del script para PostgreSQL.

2.4.1 Tipos de datos

Aunque algunas de las conversiones de tipo de datos de Oracle a PostgreSQL son sencillas, las conversiones de otro tipo de datos exigirá la evaluación de varias opciones como se muestra en la Tabla 4.

Oracle	***PostgreSQL***
VARCHAR2, CLOB, LONG, NCHAR, NVARCHAR2, NCLOB	***varchar*** *ó* ***text.*** *(Si la longitud de los datos en la columna es menor de 8000 bytes, usar* ***varchar****; sino, se debe usar* ***text.****)*
NUMBER	**smallint, integer, bigint, decimal, numeric, real, double precision.**
BINARY FLOAT, BINARY DOUBLE	**real precision, double precision.**
BLOB, RAW, LONG RAW	***bytea, lo.*** (Requiere instalación de un módulo de PostgreSQL.)
DATE	***date*** *ó* ***timestamp.***

Tabla 4: Tabla de equivalencia de tipos de datos.

La equivalencia entre los tipos de datos entre Oracle y PostgreSQL es casi perfecta, en algunos casos se tienen varias opciones, las cual se deberán evaluar cuidadosamente, como en el caso de los valores numéricos, que dependiendo del dominio de valores, será el tipo de dato a usar.

La conversión entre los tipos de datos se recomienda realizar a través de herramientas de modelado de base de datos, para automatizar la creación las tablas de la base de datos. Estas herramientas nos proveerán las conversiones recomendadas de tipos de datos, las cuales se cambiarán cambiar en caso de ser necesario.

2.4.1 Secuencias

Una secuencia es un objeto de esquema que genera una serie de números enteros únicos, y es apropiada sólo para las tablas que utilizan columnas numéricas sencillas como claves (33). Cuando una aplicación

inserta una nueva fila en una tabla, la aplicación solicita simplemente una secuencia de base de datos para proporcionar el siguiente valor disponible de la secuencia para el valor de clave primaria de la nueva fila. Lo que es más, la aplicación puede volver a utilizar posteriormente un número de secuencia generada para coordinar los valores de clave ajena de las filas secundarias relacionadas.

Tanto Oracle como PostgreSQL gestionan la generación de secuencias con una cantidad insignificante de actividad, permitiendo que funcionen muy bien incluso las aplicaciones de procesamiento de transacciones en línea más exigentes.

Para crear una secuencia, se utiliza el comando SQL CREATE SEQUENCE como se muestra en la Tabla 4.

Oracle	***PostgreSQL***
CREATE SEQUENCE [*esquema.*] *secuencia* [START WITH *entero*] [INCREMENT BY *entero*] [MAXVALUE *entero* | NOMAXVALUE] [MINVALUE *entero* | NOMINVALUE] [CYCLE I NOCYCLE] [CACHE *entero* I NOCACHE] [ORDERI NOORDER]	CREATE [TEMPORARY | TEMP] SEQUENCE [*esquema.*] *seuencia* [INCREMENT [BY] *entero*] [MINVALUE *entero* | NO MINVALUE] [MAXVALUE *entero* | NO MAXVALUE] [START [WITH] *entero*] [CACHE *entero*] [[NO] CYCLE] [OWNED BY { *table*.*column* | NONE }]

Tabla 5: Sentencia de creación de secuencias

Cuando se crea una secuencia se puede personalizar para que se adecue a las necesidades especificas de una aplicación; por ejemplo, una secuencia de PostgreSQL puede ascender o descender en uno o más números enteros, tener un valor máximo o mínimo, etc.

Si es necesario, se puede cambiar posteriormente las propiedades de una secuencia utilizando el comando ALTER SEQUENCE de SQL. El comando ALTER SEQUENCE soporta las mismas opciones y los mismos parámetros que el comando CREATE SEQUENCE, con la excepción del parámetro START WITH.

Creación de las secuencias

Para la generación de las secuencias se utilizará el script SQL generado desde Oracle llamado secuencias.sql. A continuación se muestra un ejemplo de transformación de una secuencia:

```
CREATE SEQUENCE scolor
MINVALUE 0
MAXVALUE 999999999999999999999999999
START WITH 11
INCREMENT BY 1
CACHE 20
ORDER;
```

La misma secuencia en PostgreSQL quedaría:

```
CREATE SEQUENCE scolor
INCREMENT 1
MINVALUE 0
MAXVALUE 9223372036854775807
START 11
CACHE 20;
```

Para la creación de todas las secuencias en el servidor PostgreSQL, se modificará las sentencias de creación de las mismas en el fichero SQL *secuencias.sql*, quedando adaptadas al lenguaje de PostgreSQL. El procedimiento se puede realizar manualmente en un sencillo editor de texto.

Una vez modificado el fichero, se procederá a la ejecución de este script en la base de datos dbsigesc en el servidor PostgreSQL.

Dentro de la base de datos dbsigesc, ejecutamos el siguiente comando:

```
dbdigesc=# \i /ruta/fichero/secuencias.sql;
```

Para visualizar todas las secuencias creadas, se ejecuta el siguiente comando:

```
SELECT v.sequence_name FROM information_schema.sequences v;
```

2.4.2 Tablas

La adaptación de las sentencias de creación de las tablas de Oralce a PostgreSQL es relativamente sencilla que los dos gestores de bases de datos cumplen con el estándar SQL92.

Oracle	*PostgreSQL*
CREATE [GLOBAL TEMPORARY] TABLE [schema.]table (column datatype [DEFAULT expr] [column_constraint(s)[,…]] [,column datatype [,…]]) [table_constraint [,…]] [table_ref_constraint [,…]] [ON COMMIT {DELETE|PRESERVE} ROWS] storage_options [COMPRESS int|NOCOMPRESS])	CREATE [[GLOBAL | LOCAL] { TEMPORARY | TEMP }] TABLE [schema.]table_name([{ column_name data_type [DEFAULT default_expr] [column_constraint [...]] | table_constraint | LIKE parent_table [{ INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS | INDEXES }] [, ...]]) storage_options

Tabla 6: Sentencia de creación de tablas

El proceso de creación de las tablas en uno u otro gestor de bases de datos, se puede automatizar, ya que existen herramientas que nos genera mucho código de forma automática, además, algunas de ellas nos permiten la conexión a la base de datos, y crearnos todos los objetos en ella de forma directa.

Generación del script de creación de las tablas

Para la creación de las tablas se utilizará el script generado desde Oracle nombrado *tablas.sql*. A continuación un ejemplo de transformación de la sentencia de creación de una tabla:

```
create table DUSUARIO
(
  IDUSUARIO          VARCHAR2(32) default rawtohex(sys_guid()) not null,
  NOMUSUARIO         VARCHAR2(20) not null,
  ELIMINADO          NUMBER(1) not null,
  IDESTADOACTIVACION NUMBER(4) not null,
  IDPERSONA          VARCHAR2(32) not null,
```

```
  TELEFONO           VARCHAR2(11)
)
tablespace TS_SIGESC;
alter table DUSUARIO add constraint PKDUSUARIO primary key (IDUSUARIO)
using index tablespace TS_SIGESC_INDEX;
```

La misma tabla en PostgreSQL quedaría:

```
CREATE TABLE dusuario (
 idusuario VARCHAR(36) DEFAULT (uuid_generate_v4())::VARCHAR NOT NULL,
 nomusuario VARCHAR(20) NOT NULL,
 eliminado BOOLEAN DEFAULT false NOT NULL,
 idestadoactivacion NUMERIC NOT NULL,
 idpersona VARCHAR(36) NOT NULL,
 telefono VARCHAR(11),
 CONSTRAINT pkdusuario PRIMARY KEY(idusuario) USING INDEX TABLESPACE
ts_sigesc_indices
);
```

Note que cuando se realiza la transformación no se especifica en la sentencia, el tablespace hacia donde irá la tabla, ya que será el que se definió para la base de datos. En el caso del índice sí, ya que se creó un tablespace para su almacenamiento, en este caso ts_sigesc_indices.

Una vez adaptado el script para PostgreSQL, se procederá a su ejecución en el servidor:

```
dbdigesc=# \i /ruta/fichero/tablas.sql;
```

Para la visualización de todas las tablas creadas, se ejecuta el siguiente comando:

```
SELECT table_name, table_schema FROM information_schema.tables
  WHERE table_type = 'BASE TABLE' AND
  table_schema NOT IN('pg_catalog', 'information_schema');
```

2.4.3 Paquetes

Para la separación de grupo de funciones relacionadas, en la base de datos del SIGESC 171, se utilizó un recurso de Oracle llamado "*paquete*".

Un paquete es un conjunto de procedimientos, funciones, variables y comandos SQL creado como una única unidad. Se utiliza para almacenar juntos objetos relacionados.

Entre las ventajas que ofrecen los paquetes en Oracle se encuentran:

- Permiten agrupar los temas relacionados, tipos y procedimientos en PL/SQL como un módulo.
- Cuando un procedimiento en un paquete que se invoca, todo el paquete se carga, aunque pasa a ser costosa la primera vez, la respuesta es más rápida en las próximas invocaciones.
- Los paquetes permite crear tipos, variables y procedimientos que son privados o públicos.

Las llamadas a los procedimientos que se encuentran dentro de los paquetes tiene la siguiente estructura:

```
paquete.procedimiento([param1, param2, ….])
```

Dado que PostgreSQL no implementa la estructura *paquete* que implementa Oracle, se hizo necesario utilizar una estructura que sirva para agrupar conjuntos de funcionalidades. Para lograrlo, se utilizaron los esquemas. Ver sección 2.3.3 Esquemas.

Por cada paquete existente en la base de datos del SIGESC 171, se creó en PostgreSQL un esquema, los cuales contienen todos los procedimientos y funciones del paquete que representan.

En la base de datos Oracle del SIGESC 171 se crearon los siguientes paquetes:

- pkg_configoperaciones
- pkg_adminf
- pkg_admrecurso
- pkg_aplicacion
- pkg_avl

- pkg_comunes
- pkg_controllistar
- pkg_despacho
- pkg_mapas
- pkg_nomenclador
- pkg_operador
- pkg_reporte
- pkg_supervisiondespacho
- pkg_supervisiongeneral
- pkg_supervisionoperador
- pkg_utilidades

Luego, la base de datos en PostgreSQL quedaría:

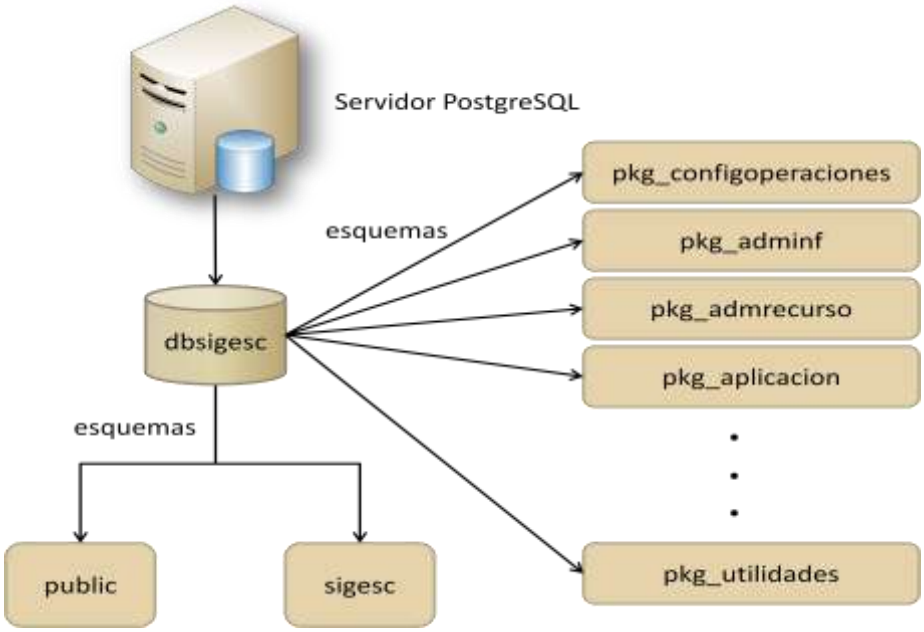

Figura 5: Esquemas por paquetes en la base de datos DBSIGESC

La creación de los esquemas como muestra la figura Figura 5, tiene entre sus ventajas:

- Mantiene agrupados y relacionados los procedimientos de la misma forma que hacen los paquetes en Oracle.
- Es la misma notación para invocar a un procedimiento contenido dentro de cualquiera de esos esquemas, que la invocación de procedimientos dentro de paquetes en Oracle. Esto minimiza los cambios a la hora de la migración de código.

Transformación de procedimientos desde Oracle (Lenguaje Procedural utilizado: PL/SQL) hacia PostgresSQL (Lenguaje Procedural utilizado: PL/pgSQL).

Hasta el momento no se ha encontrado una herramienta de conversión automática de código PL/SQL hacia PL/pgSQL, por tanto que hay que codificar todos los procedimientos existentes en la base de datos del SIGESC 171 manualmente hacia PostgreSQL.

PostgreSQL ofrece varios lenguajes para la programación de procedimientos en el servidor. Entre todos, el que recomiendan para realizar la migración desde Oracle es el PL/pgSQL. Aunque no es 100% compatible, las especificaciones de ese lenguaje son bastante similares a las del PL/SQL de Oracle (34).

PL/pgSQL es similar a PL/SQL en muchos aspectos. Es un lenguaje estructurado, imperativo, y todas las variables tienen que declararse. Asignaciones, ciclos y condicionales son similares. Las principales diferencias que se deben tener en cuenta a la hora de portar de PL/SQL a PL/pgSQL son:

- No existen parámetros por defecto en PL/pgSQL.
- Se puede sobrecargar las funciones en PL/pgSQL. Esto se utiliza a menudo para evitar la falta de parámetros por defecto.
- Asignaciones, ciclos y condicionales son similares.
- En lugar de paquetes, la utilización de esquemas para organizar las funciones en grupos.
- Como no hay paquetes, tampoco existen las variables internas de paquetes. Esto es algo molesto. Se puede solucionar guardando el estado en tablas temporales.
- PL/pgSQL no tiene procedimientos, solo tiene funciones. Es decir, tanto las funciones como los procedimientos Oracle, pasan a ser funciones en PostgreSQL.

2.5 Seguridad del Servidor

Para la implementación de la seguridad en la base de datos del SIGESC 171, se utilizaron varios métodos de seguridad de la información y a varios niveles.

- *Seguridad a objetos*: Establece la autorización para que determinadas operaciones o comandos puedan ser ejecutados sobre objetos en el servidor de bases de datos (…) (35). Los permisos son asignados mediante el comando GRANT, y retirados con el comando REVOKE.
- *Seguridad por usuarios*: Establece seguridad a nivel de cuentas de usuarios en el servidor. Solamente se autoriza a acceder a los datos a los usuarios que tengan cuenta en el servidor de base de datos. Cada cuenta en el servidor se conforma de clave y nombre usuario (…) (35).

2.5.1 Seguridad a objetos

Cuando un objeto es creado, este es asignado a un propietario. El propietario es usualmente un usuario que ejecutó una sentencia de creación. Para los objetos de la base de datos, su estado inicial es que solamente el propietario (o un superusuario) puede hacer cualquier cosa sobre el objeto. Para que otros usuarios (o roles) puedan usarlos, se les deben conceder *privilegios* (36). Existen diferentes tipos de privilegios: SELECT, INSERT, UPDATE, DELETE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE, y USAGE.

Para la asignación de privilegios se utiliza el comando GRANT. Entonces, si existe un usuario *operador*, y una tabla *dpersona*, el privilegio para seleccionar en la tabla puede otorgarse de la siguiente manera:

```
GRANT SELECT ON dpersona TO operador;
```

Si no se quiere otorgar permiso directo de selección a la tabla dpersona, se puede realizar mediante una vista que seleccione sobre la tabla en cuestión, ejemplo:

```
GRANT SELECT ON viewpersona TO operador;
```

Si se desea realizar una inserción sobre la tabla dpersona y no se desea que el usuario tenga directamente el privilegio de INSERT sobre esta, se realizará mediante una función que inserte sobre la tabla en cuestión, ejemplo:

```
GRANT EXECUTE ON FUNCTION insertarpersona(nombre VARCHAR) TO operador;
```

Cuando se le otorga permisos sobre una función a un usuario (o rol), a diferencia de Oracle, en PostgreSQL se tiene que especificar los parámetros de la función, ya que en PostgreSQL existe la sobrecarga de funciones (funciones con el mismo identificador pero con parámetros diferentes).

El acceso a las tablas a través de funciones se aprovechó en el SIGESC 171 para evitar que las aplicaciones interactuaran directamente con las tablas donde se almacenan los datos.

2.5.2 Seguridad por usuarios

Como parte de la implementación de la seguridad, en la base de datos del SIGESC 171 se ha diseñó además una seguridad a nivel de roles y usuarios que permitan controlar el acceso a la información. Para ello se han creado los siguientes roles:

- dbsession
- dbactivacion
- dbaplicacion
- dbnomeclador
- dbsincronizacion
- dbadministradorrecurso
- dbestadistica
- dbavl
- dbmapificacion
- dboperaciones
- dboperador
- dbdespachador
- dbsupervisoroper

- dbsupervisordesp
- dbsupervisorgen
- dbadministrador

Hay roles, como *dbsesion* que se relacionan con otros, permitiendo que el otro rol pueda tener los mismos privilegios del rol *dbsesion*. La forma de relacionar los roles se realiza mediante el comando GRANT.

Figura 6: Ejemplo de relación entre roles

En la Figura **6** el sentido de la flecha representa que todos los privilegios del rol *dbsesion* se le conceden al rol *dbestadística* mediante la ejecución del comando GRANT.

La Figura 6 representa la ejecución del siguiente comando:

```
GRANT dbsesion TO dbestadistica;
```

Visto lo anterior si la definición del rol *dbsesion* en PostgreSQL es la siguiente:

```
-- Create the role
CREATE ROLE dbsession;
-- Grant/Revoke object privileges
GRANT EXECUTE ON bloquearsesion(idsesion VARCHAR, fecha DATE, prmnombreusuario VARCHAR) TO dbsession;
GRANT EXECUTE ON cerrarsesion(idsesion VARCHAR, fecha DATE, prmnombreusuario VARCHAR) TO dbsession;
GRANT EXECUTE ON desbloquearsesion(idsesion VARCHAR, fecha DATE, prmnombreusuario VARCHAR) TO dbsession;
GRANT EXECUTE ON iniciarsesion(nombusu VARCHAR, prmidpunto VARCHAR, prmidaplicacion VARCHAR, fecha DATE) TO dbsession;
```

Una vez relacionado el rol *dbsesion* con el rol *dbestadistica*, el rol dbestadistica podrá ejecutar las funciones bloquearsesion, cerrarsesion, desbloquearsesion e iniciarsesion.

La Figura 7 representa cómo se relacionan los roles en el SIGESC 171.

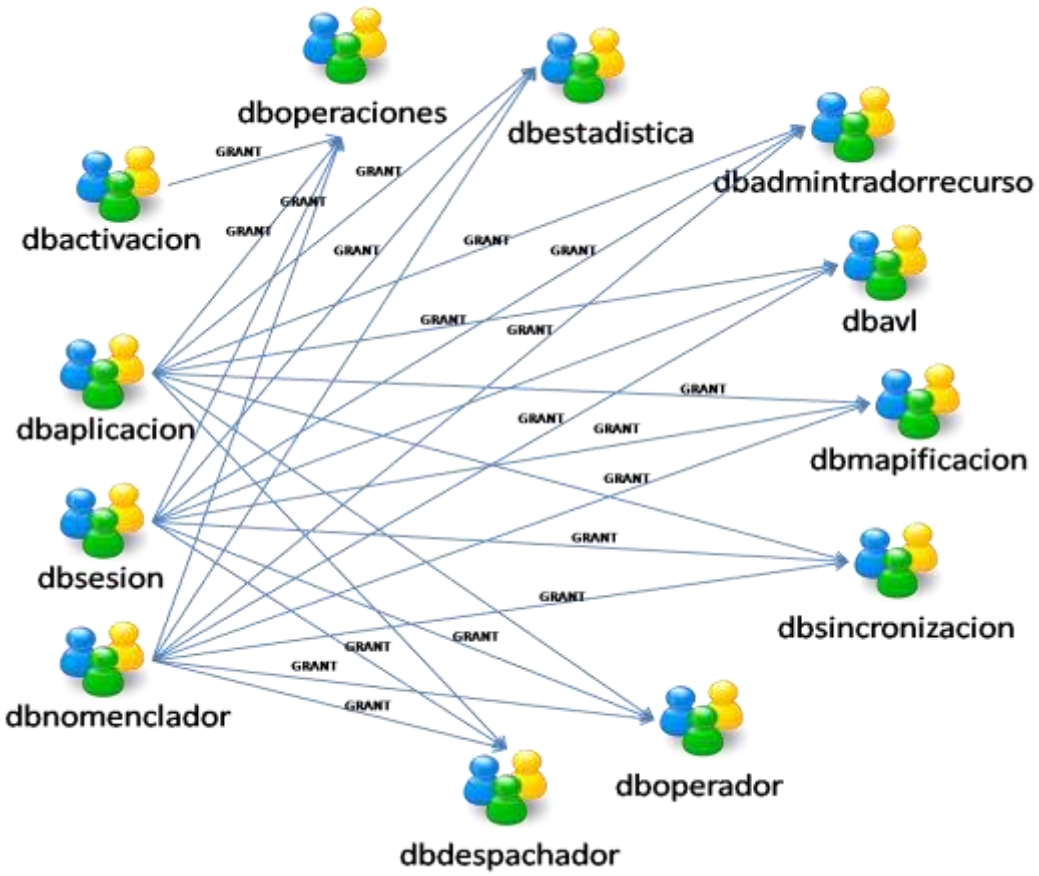

Figura 7: Relaciones entre roles en el SIGESC 171

Como se puede apreciar en la Figura 7, existe una fuerte relación entre los roles *dbsesion, dbnomenclador* y *dbaplicacion* con los demás roles, debido a esto en el momento de la creación de los otros roles, explícitamente se deberá relacionarlos, con el comando GRANT. Es importante destacar que este es un diseño complejo de mantener por parte del DBA del sistema, ya que no se distingue una jerarquía entre los roles del sistema, además dificulta la comprensión.

Para resolver el problema de la relación directa que existe entre los roles *dbsesion, dbnomenclador* y *dbaplicacion* y los demás roles, se propone la creación de un rol llamado *dbmodulo*, el cual agrupará los privilegios de los roles *dbsesion, dbnomenclador* y *dbaplicacion.*

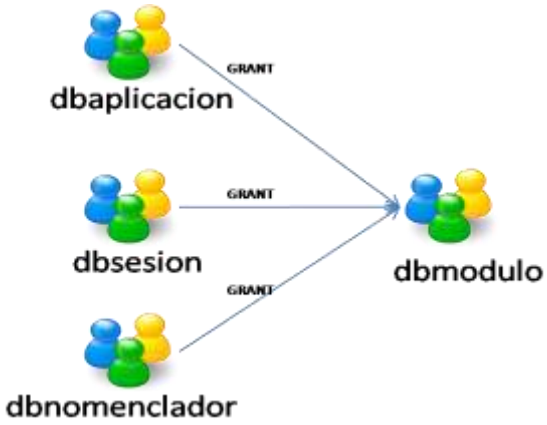

Figura 8: Creación del rol dbmodulo

La definición del rol en PostgreSQL quedaría:

```
-- Create the role
CREATE ROLE dbmodulo;
-- Grant privileges
GRANT dbaplicacion TO dbmodulo;
GRANT dsesion TO dbmodulo;
GRANT dnomenclador TO dbmodulo;
```

Una vez creado el rol *dbmodulo*, se eliminarían las relaciones directas como se muestra en la Figura 9:

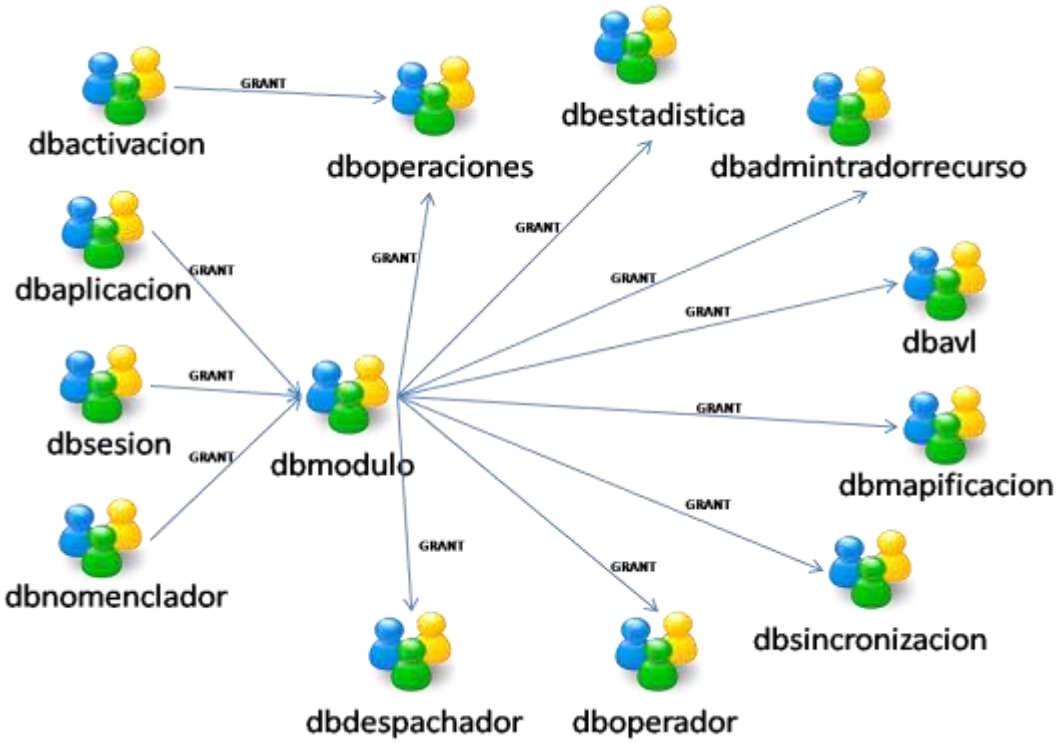

Figura 9: Propuesta de relaciones entre roles para el SIGESC 171.

El nuevo diseño tiene ventajas:

- El diseño propuesto se puede apreciar una jerarquía bien definida de los roles que intervienen en el sistema, además de un bajo acoplamiento entre los roles, lo que permite la fácil reutilización y administración.
- El diseño es más comprensible por la eliminación del las relaciones existentes entre muchos de los roles.

2.5.3 Seguridad basada en la autenticación del cliente

Como propuesta para aprovechar los diferentes métodos de seguridad que ofrece PostgreSQL y así aumentar la seguridad del SIGESC 171, se propone que se realice por parte del servidor de bases de datos una *Seguridad basada en la autenticación del cliente.*

La Seguridad basada en la autenticación del cliente es el proceso mediante el cual el servidor de base de datos establece la identidad del cliente y, por extensión, determina si la aplicación cliente (o el usuario que ejecuta la aplicación cliente) está autorizado a conectarse con la base de datos con el nombre de usuario que fue solicitado. (37)

PostgreSQL ofrece una serie de diferentes métodos de autenticación de cliente. El método utilizado para autenticar a un cliente en particular se puede seleccionar sobre la base de (cliente) dirección de host, base de datos, y el usuario.

La autenticación del cliente es controlado por un archivo de configuración, que tradicionalmente se denomina *pg_hba.conf* y se almacena en el directorio de instalación del servidor PostgreSQL.

El formato general del fichero pg_hba.conf es un conjunto de registros, uno por línea. Las líneas en blanco se ignoran, al igual que cualquier texto después del carácter #. Un registro se compone de una serie de campos que están separadas por espacios y/o tabs. Los campos pueden contener espacios en blanco si el valor del campo es delimitado por el carácter (‘).

En cada registro se especifica un tipo de conexión, un rango de direcciones IP del cliente (si es relevante para el tipo de conexión), un nombre de una base de datos, un nombre de usuario, y el método de autenticación que se utilizará para las conexiones que coincidan con estos parámetros.

Los registros pueden tomar una de estas formas:

```
local DATABASE  USER  METHOD  [OPTION]
host DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
hostssl DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
hostnossl DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
```

local: Este registro coincide con los intentos de conexión usando sockets de dominio Unix. Sin un registro de este tipo, Unix socket conexiones son rechazados.

host: Este registro coincide con los intentos de conexión utilizando TCP / IP.

hostssl: Este registro coincide con los intentos de conexión utilizando TCP / IP, pero sólo cuando la conexión se realiza con cifrado SSL.

hostnossl: Este tipo de registro se ha opuesto a la lógica hostssl: que sólo coincide con los intentos realizados a lo largo de la conexión TCP / IP que no utilizan SSL.

database*:* Especifica a que base de datos se tiene acceso. Si se coloca el valor *all* se especifica que se tendrá acceso a todas las bases de datos. El valor *sameuser* especifica que se tendrá acceso a la base de datos que tenga el mismo nombre del usuario solicitado. En caso contrario, se coloca un nombre de una base de datos PostgreSQL.

users: Especifica a que nombre de usuario de la base de datos el registro es aplicable. El valor *all* especifica que se aplicará a todos los usuarios. En caso contrario, puede ser el nombre de un usuario específico de la base de datos, o de un rol precedido del signo (+). Pueden ser especificados diferentes nombres de usuarios separándolos con el signo (,). Puede especificarse un archivo que contiene los nombres de los usuarios mediante el nombre del archivo precedido del signo (@).

CIDR-address: Especifica a que rango de IP el registro es aplicable.

method: Especifica el método de autenticación que será utilizado por las conexiones que coincidan con este registro. Permite diferentes valores:

Método de Autenticación	***Descripción***
trust	Permite la conexión incondicional. Este método permite que cualquiera pueda conectarse a la base de datos como un usuario válido sin requerir contraseña. Por supuesto, las restricciones realizadas en la base de datos, tablas y objetos en general siguen siendo aplicadas.
reject	Rechaza la conexión incondicionalmente. Es usado para filtrar algunos usuarios y hosts.
md5	Se realizará la autenticación mediante la contraseña del usuario de la base de datos. La contraseña viajará por la red encriptada.
crypt	Funciona de forma similar a *md5*. Se utiliza cuando se tiene clientes que no soportan versiones de PostgreSQL mayores que la 7.2.

password	Requiere por parte del cliente de una contraseña sin encriptar. Se envía la contraseña en texto claro a través de la red, esto no debe ser utilizado en redes no confiables.
gss	GSSAPI es un protocolo estándar de la industria para la autenticación segura definido en el RFC 2743. PostgreSQL soporta GSSAPI con la autenticación Kerberos de acuerdo con RFC 1964. GSSAPI proporciona autentificación automática (Single Sign-On) para los sistemas que lo soporten. La autenticación en sí es seguro, pero los datos enviados a través de la conexión será texto claro a menos que se utiliza SSL.
sspi	Usa SSPI para autenticar el usuario. Solamente es está disponible para Windows.
ldap	Este método de autenticación funciona de manera similar a *password,* excepto que se utiliza LDAP como método de autenticación. LDAP se usa sólo para validar pares nombre de usuario / contraseña. Por lo tanto, el usuario ya debe existir en la base de datos antes de que LDAP pueda utilizarlos para la autenticación.

Tabla 7: Métodos de autenticación en PostgreSQL

Dado que los registros de pg_hba.conf son examinados secuencialmente para cada intento de conexión, el orden de los registros es importante. Normalmente los primeros registros son lo que tendrán escasos parámetros de coincidencia y métodos de autenticación más flexibles, mientras que últimos registros tendrán parámetros de coincidencia más flexibles y métodos de autenticación más fuertes. Por ejemplo, se podría desear conexiones de confianza (*trust*) para conexiones TCP/IP locales, pero que las conexiones remotas requieran contraseña. En este caso el registro que especifica la conexión de confianza debe aparecer antes que el registro que especifica las conexiones con contraseña para un amplio rango de direcciones IP.

Como propuesta de configuración de autenticación de los clientes para el servidor del SIGESC 171 se propone la siguiente configuración del fichero pg_hba.conf

```
# Allow any user on the local system to connect to any database under
# any database user name using Unix-domain sockets (the default for local
# connections).
```

```
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
Local   dbsigesc    sigesc                             trust

# The same using local loopback TCP/IP connections.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    dbsigesc    sigesc       127.0.0.1/32           trust

# Entrada controlada a la base de datos del SIGESC 171
#
# TYPE     DATABASE        USER        IP-ADDRESS      METHOD
host     dbsigesc      @acces_file  <rango IP>     md5
```

Figura 10: Propuesta de fichero pg_hba.conf para el SIGESC 171

En el fichero de configuración, con la primera y segunda entrada se asegura que las conexiones locales a la base de datos no requieran contraseña, se utilizará para fines de administración exclusivamente. Se puede apreciar también que existe una entrada controlando las conexiones a la base de datos *dbsigesc*, la cual contiene un listado de control de acceso almacenado en un fichero llamado *acces_file*, el cual debe estar en el mismo directorio que el fichero pg_hba.conf. Se sustituirá <rango IP> por las direcciones (o rangos) IP desde las cuales se podrán conectar a la base de datos con una contraseña, separadas por coma.

El fichero acces_file tendrá el siguiente contenido:

```
+dboperaciones,+dbestadistica,+dbadministradorrecurso,+dbmapificación,+dbsincronización,+dbperador,
+dbdespacharor,+dbavl,+dbestadistica
```

Figura 11: Contenido del fichero acces_file

Como último paso, se realiza el ajuste de permisos del fichero acces_file, modificándolo para que solamente el usuario administrador del servidor y el usuario que está ejecutando el proceso del PostgreSQL tengan acceso de lectura y modificación del mismo.

Con la implementación de estas estrategias de seguridad se garantiza el control de todos los usuarios que se conectan a la base de datos, así como cuales usuarios ejecutarán determinado procedimiento, quienes tendrán acceso a tabla, esquema, vista, entre otros objetos que se crearon en la base de datos del SIGESC 171.

2.6 Copia de Seguridad (Backup)

La capacidad de mantener un sistema constante de datos es probablemente la parte más importante de un trabajo de los administradores de la base de datos. En caso de un fallo del sistema, la necesidad de la recuperación de los datos se convierte en una prioridad superior. Sin un procedimiento de reserva definido, la capacidad de restaurar datos a un estado constante es obstaculizada seriamente o imposible. El propósito epígrafe es contornear los procedimientos de salva y de recuperación que se utiliza en PostgreSQL.

Una copia de seguridad (backup) válida es una copia de la información sobre la base de datos necesaria para reconstruirla a partir de un estado no utilizable de la misma. Normalmente, si la estrategia de backup se basa en la copia de los ficheros de datos y en el archivado de los ficheros de log (WAL), se han de tener copias de los ficheros de datos, y también de los ficheros log archivados. Si se pierde uno de los ficheros de log se dice que se tiene un agujero en la secuencia de ficheros. Esto invalida el backup, pero permite a la base de datos ser llevada hasta el principio del agujero realizando una recuperación incompleta.

Antes de nada, es muy importante entender ciertas reglas que determinan la situación de los ficheros y otras consideraciones que afectarán al esquema de backup:

- Es recomendable archivar los ficheros de log en disco, y luego copiarlos a cinta, pero siempre en un disco diferente del que soporta los ficheros de datos.
- Los ficheros copias no deben estar en el mismo dispositivo que los originales. No siempre hay que pasar las copias a cinta, ya que si se dejan en disco se acelera la recuperación. Además, si se

copian las copias a cinta y se mantienen en el disco, se puede sobrevivir a diversos fallos de dispositivo.

Teniendo en cuenta las reglas anteriores, los siguientes puntos pueden considerarse un ejemplo de estrategia de *backup*:

- Activar el archivado de log.
- Realizar un backup al menos una vez a la semana si la BD se puede parar. En otro caso, realizar backups en caliente cada día.
- Copiar todos los ficheros log archivados. El tamaño y el número de ellos dependerá de la tasa de transacciones.
- Efectuar un export de la BD semanalmente.

Existen tres formas principales de realizar copias de seguridad:

- Copia de seguridad de ficheros del SO.
- Volcado SQL usando las herramientas PostgreSQL: *pg_dump (pg_dumpall) y pg_restore*.
- Volcado en línea y recuperación en el punto (PITR).

2.6.1 Copias de seguridad de ficheros del SO

Es el método más sencillo, pero el más ineficaz, se trata de realizar una copia de todos los ficheros de un cluster, por ejemplo:

```
$ su – postgres
$ cd /tmp/backup
$ tar -cvfz copia.tar.gz $PGDATA7
$ --- > copiar el fichero anterior a cinta …
```

Desventajas de este método:

- La base de datos debe estar parada.

[7] Directorio que contiene los datos del cluster de base de datos en PostgreSQL.

- No se pueden recuperar partes del cluster (bases de datos, esquemas, tablas, etc.)

La recuperación consiste en borrar todo el cluster y descomprimir el fichero de copia de seguridad, con lo que se pierden los datos que se hayan modificado desde la última copia de la base de datos.

2.6.2 Volcado SQL

Los volcados de este tipo se realizan usando las herramientas que proporciona PostgreSQL. Estos volcados son muy flexibles y de gran utilidad, permitirán hacer copias de seguridad de toda la base de datos o de partes de ella, y luego, dada una copia de seguridad, permitirán restaurar lo que se desee.

Además, estas herramientas sirven para la transmisión de datos entre bases de datos. Las herramientas que se van a utilizan son:

pg_dump: Vuelca una base de datos o parte de ella a un fichero, bien en texto plano o en un formato propio de PostgreSQL. Se puede recuperar cualquier objeto que esté en el fichero aisladamente. El servidor debe estar en marcha.

pg_dumpall: Vuelca un cluster completo.

pg_restore: Recupera los objetos volcados en una copia de seguridad que no se realizó en texto plano sino en un formato propio de PostgreSQL.

psql: se usa para recuperar los volcados en texto plano.

Desventajas de esté método:

- En grandes bases de datos el proceso de volcado hacia un fichero puede demorar.
- Dependiendo del tamaño de la base de datos, el proceso puede generar un fichero de mayor tamaño que es que soporta el sistema de ficheros.
- Se pierde toda la información sobre las transacciones que se están realizando en el momento de la realización del backup.

2.6.3 Volcado en línea y recuperación PITR

Hasta la versión 7.X no había recuperación total, si se producía una pérdida dentro del $PGDATA, había que recurrir a la copia de seguridad física más reciente y desde ese momento las actualizaciones se perdían, aunque tuviéramos los ficheros de log. Aquí es donde, en la versión 8.X se incorpora el volcado en línea y la recuperación PITR.

A partir de la versión 8.0.0, PostgreSQL permite actualizar la copia de seguridad con los cambios que proporcionan en los ficheros de log que hayan sobrevivido.

Para poder utilizar esta opción tiene que estar habilitado y en funcionamiento el archivado WAL. Los ficheros WAL son segmentos de 16Mb que se nombran secuencialmente en el cual el sistema va copiando los cambios en la base de datos. El sistema recicla los ficheros de log que no van a ser necesitados renombrándolos a números superiores dentro de la secuencia. Para activar el archivado, en el fichero *postgresql.conf* debemos indicar el comando de copia para preservar los ficheros de log, en el parámetro archive_command, haciendo, por ejemplo:

```
archive_command = 'cp –i %p /mnt/server/archivedir/%f < /dev/null'
```

donde '%p' representa el nombre del fichero de log con la ruta absoluta y '%f' sin la ruta.

Para realizar copias de seguridad en línea se siguen los siguientes pasos:

1. Activar el archivado WAL.
2. Antes de empezar y desde una consola SQL hay que ejecutar:
   select pg_start_backup('nombre_copia');
3. Con el servidor en marcha, hacemos la copia desde el sistema operativo, no hace falta parar el servidor, por ejemplo:
   $ tar –cvf backup_nombre_copia.tar $PGDATA.
4. Cuando acaba, desde la consola SQL, marcamos el final, ejecutamos:
   select pg_stop_backup();
5. Se crea así un fichero de marca en el directorio $PGDATA/pg_xlog/archive_status y copia los logs que se reciclan en donde se haya indicado en archive_command.

Este método tiene como principal ventaja que almacena la información de las transacciones confirmadas cuando se está realizando el proceso de salva.

Desventajas de este método:

- Como este método salva todos los ficheros de log, más el directorio de datos del servidor PostgreSQL, tiene como principal desventaja el tamaño que puede alcanzar el backup.

2.6.4 Política y planes de copias de seguridad para el SIGESC 171

En el SIGESC 171 se realiza un backup completo de la base de datos los fines de semana. Para su preparación y realización en PostgreSQL se propone seguir el siguiente procedimiento:

1. Se establece el parámetro de configuración `archive_mode` a `on`, y se crea un comando para la opción `archive_command` que realice el archivado de los archivos de log. Por ejemplo.

   ```
   archive_command = 'test ! -f /var/lib/pgsql/backup_in_progress || cp -i %p /var/lib/pgsql/archive/%f < /dev/null'
   ```

   Este comando realizará el archivado cuando exista el fichero /var/lib/pgsql/backup_in_progress que funciona como “archivo interrupor”, si no existe este archivo, silenciosamente retornará cero como código de salida (permitiendo a PostgreSQL reciclar el fichero de log).
2. Luego de esta preparación, se puede realizar el backup usando un script como el siguiente:

   ```
   touch /var/lib/pgsql/backup_in_progress
   psql -c "select pg_start_backup('hot_backup');"
   tar -cf /var/lib/pgsql/backup.tar /var/lib/pgsql/data/
   psql -c "select pg_stop_backup();"
   sleep 20
   rm /var/lib/pgsql/backup_in_progress
   tar -rf /var/lib/pgsql/backup.tar /var/lib/pgsql/archive/
   ```

   Primeramente es creado el fichero /var/lib/pgsql/backup_in_progress, permitiendo el archivado de todos los ficheros de log que se puedan crear en el tiempo que dure el proceso de creación del

backup. Luego de crearse el backup se borra el archivo interruptor. Por último los archivos de log archivados se añaden al backup.

2.6.5 Recuperación de la Base de Datos

La recuperación de los datos es el proceso de revertir tanto la estructura como los datos de la base de datos, en cualquier momento hacia un punto anterior en el tiempo. Teniendo así la posibilidad de recuperarse frente a los distintos problemas que puedan provocar la caída del servidor.

2.6.5.1 Recuperación utilizando un Archivo Backup

Una vez caído el servidor, se siguen los siguientes pasos para restaurar nuestra base de datos con la última copia de misma:

1. Se detiene el servicio del PostgreSQL, si está en ejecución.
2. Borrar todo los ficheros y subdirectorios existentes bajo el directorio de datos del cluster y de la carpeta raíz de los tablespaces.
3. Restaurar los ficheros de la base de datos desde el último archivo de salva.
4. Crear un fichero de recuperación *recovery.conf* en el directorio de datos del cluster y agregarle unas líneas como las siguientes:

   ```
   restore_command = 'cp /var/lib/pgsql/%f "%p"'
   ```

5. Iniciar el servidor. El servidor arrancará en modo recuperación y procederá a leer los ficheros de log archivados que necesita. Una vez completado el proceso de recuperación, el servidor renombrará el fichero *recovery.conf* a *recovery.done* (para prevenir que accidentalmente vuelva a entrar en modo recuperación si vuelve a caer) y comenzará las operaciones normales sobre la base de datos.
6. Inspeccionar el contenido de la base de datos para asegurarse que se haya restaurado lo que se desea.

Los planes de recuperación esta sujetos a problemas en un entorno de producción real, ya que estos son aplicables cuando ha surgido un problema, mientras tanto se deben realizar pruebas sistemáticas de las

salvas que se hayan realizado para verificar la integridad de las mismas, y prevenir cualquier incidencia sobre estos ficheros, como pueden ser corrupción de los ficheros, deterioro de los dispositivos de almacenamiento entre otros.

Conclusiones del capítulo

En este capítulo se definió la estrategia a seguir para llevar a cabo la migración del diseño y administración del SIGESC 171, logrado a partir de las especificaciones del funcionamiento y administración de la base de datos, de forma tal que permita obtener un sistema estable con una alta disponibilidad y una buena integración.

3 Validación de la solución

Este capítulo abarca:

- Aplicación del Método Delphi para la validación
- Resultados del procesamiento estadístico de las encuestas.

CAPÍTULO III
Validación de la solución

Introducción

La propuesta descrita en el capítulo anterior cubre, adaptándose a las características del proyecto SIGESC 171, todos los aspectos importantes de la migración del diseño y administración de la base de datos.

Sin embargo: ¿Cómo puede asegurarse que la propuesta descrita en el segundo capítulo de esta investigación es realmente la solución al tema de la migración del diseño y la administración del SIGESC 171?, la mejor respuesta a esta interrogante hubiese sido un informe que contenga los resultados de pruebas de funcionalidad, rendimiento y seguridad que le puedan realizar al SIGESC 171 luego de aplicar la propuesta en un entorno similar al que se encuentra desplegado en Venezuela. Este informe contendría una descripción del estado del sistema antes de la aplicación de la propuesta, una tabulación de elementos surgidos de la aplicación misma de la propuesta que incluyera puntos fuertes y puntos problemáticos con el fin de mejoras futuras, y al final una descripción del estado de la aplicación después de aplicada la propuesta.

Aún así la propuesta debe ser validada de algún modo. Por no contar con la infraestructura tecnológica para crear un escenario similar al que se encuentra desplegado el SIGESC 171 en Venezuela, se propone la utilización de un método científico que permita realizar la validación teórica de la propuesta, específicamente el método Delphi, el cual realiza la validación basado en el método de expertos por lo que lo convierte en uno de las vías ampliamente utilizadas en validación de trabajos investigativos.

3.1 El Método Delphi

Método de estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo. El método se basa en la organización de un **diálogo anónimo** entre los expertos consultados de modo individual, a partir de la aplicación de un cuestionario y con el propósito de obtener un consenso general o los motivos discrepantes entre estos. Los expertos, seleccionados previamente, se someten a una serie de interrogantes sucesivas, cuyas respuestas se procesan estadísticamente para conocer la coincidencia o discrepancia que estos tienen en cuanto a lo consultado. (38)

Principales características del método Delphi:

1. **Anonimato**: los expertos contestan las preguntas sin consultarse mutuamente (por lo que es recomendable que dos expertos no conozcan entre sí que están opinando sobre un mismo tema).
2. **Retroalimentación controlada:** después de cada ronda de preguntas, se tabulan las respuestas y se procesan antes de la siguiente ronda, para que los participantes puedan evaluar los resultados de la ronda anterior, así como las razones dadas para cada respuesta y su dispersión del promedio (esto permite que aumente el acuerdo al transcurrir varias rondas del proceso).
3. **Respuesta estadística del grupo**: el procesamiento de cada ronda se realiza con métodos estadísticos. Esto es la característica más importante que diferencia a este método de otros subjetivos.

Para la aplicación del método se siguieron 3 etapas fundamentales:

- Elección de los expertos.
- Elaboración del cuestionario de validación de la propuesta.
- Desarrollo práctico y evaluación de los resultados.

3.1.1 Elección de los expertos

Se ha considerado como experto una persona capaz de ofrecer valoraciones conclusivas sobre Administración de Bases de Datos y de hacer recomendaciones sobre este tema con determinado nivel de competencia.

La selección de los posibles candidatos se realizó bajo los criterios siguientes:

- Graduado de nivel superior.
- Vinculación al desarrollo de proyectos informáticos.
- Mínimo de un año de experiencia.
- Conocimientos sobre Sistemas Gestores de Bases de Datos.

- Experiencia práctica en el uso, y administración de Sistemas Gestores de Bases de Datos.

3.1.2 Elaboración del cuestionario

El cuestionario de validación aplicado a los expertos seleccionados para el panel del presente trabajo de diploma se encuentra en el anexo 1 de esta investigación.

3.2 Resultados del procesamiento estadístico de las encuestas

Se envió el cuestionario a los expertos seleccionados, vía e-mail o se les dio en copia dura (se tuvo en cuenta las no-respuestas y abandonos). Naturalmente se le explicó a cada experto las finalidades, el espíritu del Delphi, así como las condiciones prácticas del desarrollo de la encuesta (plazo de respuesta y garantía de anonimato).

Logrado ya el equipo de expertos, se buscaron sus criterios sobre la selección de la propuesta de migración.

Se confeccionarán tablas para ir recogiendo los resultados aportados por los expertos. Para ello se utilizó el programa Microsoft Excel 2007.

Criterio de expertos: C1: Muy adecuado, C2: Bastante adecuado, C3: Adecuado, C4: Poco adecuado, C5: No adecuado

No	Elementos	C1	C2	C3	C4	C5	Total
1	Pasos para realizar la migración	0	5	2	0	0	7
2	Definición de los Tablespaces	0	6	1	0	0	7
3	Creación de los esquemas y base de datos	0	6	1	0	0	7
4	Selección de los tipos de datos equivalentes	0	7	0	0	0	7
5	Transformación de las Tablas y Secuencias	0	4	3	0	0	7
6	Creación de esquemas por paquetes	0	6	1	0	0	7
7	Definición de roles de usuarios	0	4	3	0	0	7
8	Creación de políticas de salva y recuperación	0	5	2	0	0	7

Tabla 8: Tabla de frecuencias absolutas

Tabulados los datos, se realizan los siguientes pasos para obtener los resultados deseados:

Primer paso: Se construye una tabla de frecuencias acumuladas. Esto es, cada número en la fila, excepto el primero se obtiene sumándole el anterior.

No	Aspectos	C1	C2	C3	C4	C5
1	Pasos para realizar la migración	0	5	7	7	7
2	Definición de los Tablespaces	0	6	7	7	7
3	Creación de los esquemas y base de datos	0	6	7	7	7
4	Selección de los tipos de datos equivalentes	0	7	7	7	7
5	Transformación de las Tablas y Secuencias	0	4	7	7	7
6	Creación de esquemas por paquetes	0	6	7	7	7
7	Definición de roles de usuarios	0	4	7	7	7
8	Creación de políticas de salva y recuperación	0	5	7	7	7

Tabla 9: Tabla de frecuencias absolutas acumuladas.

Observación: En la frecuencia acumulativa desaparece la última columna.

Segundo paso: Se copia la tabla anterior y se borran los resultados numéricos. Ahora, en esta nueva tabla, se construye la tabla de frecuencias relativas acumulativas.

No	Aspectos	C1	C2	C3	C4	C5
1	Pasos para realizar la migración	0.0001	0.7143	0.9999	0.9999	0.9999
2	Definición de los Tablespaces	0.0001	0.8571	0.9999	0.9999	0.9999
3	Creación de los esquemas y base de datos	0.0001	0.8571	0.9999	0.9999	0.9999
4	Selección de los tipos de datos equivalentes	0.0001	0.9999	0.9999	0.9999	0.9999

5	Transformación de las Tablas y Secuencias	0.0001	0.5714	0.9999	0.9999	0.9999
6	Creación de esquemas por paquetes	0.0001	0.8571	0.9999	0.9999	0.9999
7	Definición de roles de usuarios	0.0001	0.5714	0.9999	0.9999	0.9999
8	Creación de políticas de salva y recuperación	0.0001	0.7143	0.9999	0.9999	0.9999

Tabla 10: Tabla de frecuencias relativas acumuladas

Esta tabla se logra dividiendo por el número total de expertos, en este caso 7 cada uno de los números de la tabla anterior.

Tercer paso: Se buscan las imágenes de los elementos de la tabla anterior por medio de la función (Dist. Normal. Standard Inv).

A la misma tabla adicione se le adicionan tres columnas y una fila para colocar los resultados que se mencionan a continuación.

1.- Suma de las columnas.

2.- Suma de filas.

3.- Promedio de las columnas.

4.- Los promedios de las filas se obtienen de forma similar, en este caso también se divide por cuatro porque quedan 4 categorías ya que la última se eliminó.

5. Para hallar N, se divide la suma de las sumas entre el resultado de multiplicar el número de indicadores por el número de preguntas.

6.- El valor N-P da el valor promedio que otorgan los expertos para cada indicador propuesto.

La tabla siguiente resume lo dicho en los puntos anteriores:

No	Aspectos	C1	C2	C3	C4	Suma	P	N-P	
1	Pasos para realizar la migración	-3.72	0.57	3.72	3.72	4.28	1.07	0.46	Bastante Adecuado
2	Definición de los Tablespaces	-3.72	1.07	3.72	3.72	4.79	1.20	0.33	Bastante Adecuado
3	Creación de los esquemas y base de datos	-3.72	1.07	3.72	3.72	4.79	1.20	0.33	Bastante Adecuado
4	Selección de los tipos de datos equivalentes	-3.72	3.72	3.72	3.72	7.44	1.86	-0.33	Bastante Adecuado
5	Transformación de las Tablas y Secuencias	-3.72	0.18	3.72	3.72	3.90	0.97	0.55	Bastante Adecuado
6	Creación de esquemas por paquetes	-3.72	1.07	3.72	3.72	4.79	1.20	0.33	Bastante Adecuado
7	Definición de roles de usuarios	-3.72	0.18	3.72	3.72	3.90	0.97	0.55	Bastante Adecuado
8	Creación de políticas de salva y recuperación	-3.72	0.57	3.72	3.72	4.28	1.07	0.46	Bastante Adecuado
	Suma	-29,75	8.41	29.75	29.75	38.17			
	P. de corte	-3.72	1.05	3.72	3.72				

Tabla 11: Puntos de corte

Las sumas obtenidas en las cuatro primeras columnas dan los puntos de cortes:

Los puntos de corte se utilizan para determinar la categoría o grado de adecuación de cada criterio según la opinión de los expertos consultados. Con ellos se opera del modo siguiente:

Muy adecuado	Bastante adecuado	Adecuado	Poco adecuado	No adecuado
-3.72	1.05	3.72	3.72	

Tabla 12: Categoría o grado de adecuación de cada criterio según la opinión de los expertos.

Conclusiones del capítulo

El presente capítulo es un análisis de los resultados obtenidos luego de aplicar el método de validación teórica Dephi para validar la propuesta de migración expuesta en el capítulo 2, a través del cual se arrojó como resultado la viabilidad de la estrategia de migración del diseño y administración de la base de datos del SIGESC 171 a PostgreSQL.

Conclusiones Generales

Al término de este trabajo se ha diseñado una estrategia para la migración de la Base de Datos del Sistema de Gestión de Emergencias y Seguridad Ciudadana (171) que se encuentra en Oracle hacia el gestor de bases de datos PostgreSQL el cual es un SGBD libre. Para su realización se le dio cumplimiento a las tareas planteadas obteniéndose como resultado una estrategia que permite obtener una base de datos gestionada por PostgreSQL que garantice el almacenamiento, integridad y seguridad de los datos del SIGESC 171.

El desarrollo del trabajo permitió además, adquirir conocimientos acerca de la administración de diferentes gestores de bases de datos y en especial del gestor PostgreSQL como manejador de los datos.

Recomendaciones

En el desarrollo del documento se ha profundizado en un conjunto de aspectos que se consideran estratégicos en el desarrollo del trabajo; a su vez otros temas solo han sido analizados brevemente a pesar de su marcada importancia. A continuación se relacionan un conjunto de ideas que se consideran necesarias para darle continuidad a este trabajo:

- Se recomienda realizar la realización de una aplicación para la migración automática de los procedimientos almacenados de Oracle a PostgreSQL.
- Realizar las pruebas de migración sobre los módulos del SIGESC 171.
- Estudiar las arquitecturas de servidores para garantizar desde el punto de vista del hardware la seguridad y disponibilidad de los datos, un rendimiento óptimo del servidor y flexibilidad a la hora de realizar los backup sin detener el servicio.
- Se propone el estudio Pentaho como herramienta libre para la migración de los datos.

Referencias Bibliográficas

1. Constitución de la República Bolivariana de Venezuela. [En línea] [Citado el: 15 de Diciembre de 2008.] http://www.tsj.gov.ve/legislacion/constitucion1999.htm.

2. **Infomatica, Ministerio del Poder Popular para las Telecomunicaciones y la.** Guía para el plan de migración a Software Libre en la Administración Pública Nacional República Bolivariana de Venezuela. [En línea] 2008. [Citado el: 15 de Diciembre de 2008.] http://www.scribd.com/doc/6442474/Guia-para-el-plan-de-migracion-a-Software-Libre-en-la-Administracion-Publica-Nacional.

3. LicensingFreeSoftware. *Free Software Foundation.* [En línea] 2009. http://www.gnu.org/philosophy/philosophy.es.html#LicensingFreeSoftware.

4. Licencias de Software. *Wikipedia.* [En línea] [Citado el: 13 de Diciembre de 2008.] http://es.wikipedia.org/wiki/Código_libre#Tipos_de_licencias.

5. Licencia GPL. *Fundación GNU.* [En línea] [Citado el: 13 de Diciembre de 2008.] http://www.gnu.org/philosophy/categories.es.html#FreeSoftware.

6. Cyberten » Tipos de Licencias. *Cyberten: Estándares Abiertos y Tecnologías OpenSource.* [En línea] [Citado el: 15 de Enero de 2009.] http://cyberten.com.ar/software-libre/tipos-de-licencias.

7. Wikipedia. *Wikipedia.org.* [En línea] [Citado el: 15 de Diciembre de 2008.] http://es.wikipedia.org/wiki/Base_de_datos.

8. Base de Datos jerárquica. *Wikipedia.* [En línea] [Citado el: 13 de Diciembre de 2008.] http://es.wikipedia.org/wiki/Base_de_datos_jerárquica.

9. Base de Datos en Red. *Wikipedia.* [En línea] [Citado el: 13 de Diciembre de 2008.] http://es.wikipedia.org/wiki/Base_de_datos_de_red.

10. Base de Datos Relacional. *Sitio Web de TE Technology, Inc.* [En línea] [Citado el: 14 de Diciembre de 2008.] http://www.mitecnologico.com/Main/BaseDeDatosRelacional.

Referencias Bibliográficas

11. Base de Datos Orientada a Objetos. *Sitio Web de la Universidad Distrital Fransisco José de Caldas. Bogotá. Colombia.* [En línea] [Citado el: 14 de Diciembre de 2008.] http://atenea.udistrital.edu.co/profesores/jdimate/basedatos1/tema7.htm.

12. Base de Datos. *Wikipedia.* [En línea] [Citado el: 13 de Diciembre de 2008.] http://es.wikipedia.org/wiki/Base_de_datos.

13. Acerca de PostgreSQL. *Comunidad de PostgreSQL en español.* [En línea] [Citado el: 12 de Enero de 2009.] http://www.postgresql-es.org/sobre_postgresql.

14. MVCC. *Sitio Web de PostgreSQL.* [En línea] [Citado el: 1 de Abril de 2009.] http://www.postgresql.org/docs/8.3/interactive/mvcc.html.

15. WAL. *Sitio Web de PostgreSQL.* [En línea] http://www.postgresql.org/docs/8.3/interactive/wal.html.

16. Extendiendo a PostgreSQL. *Sitio Oficial de PostgreSQL.* [En línea] [Citado el: 20 de Diciembre de 2008.] http://www.postgresql.org/docs/8.3/static/extend.html.

17. **Corporation, MySQL.** MySQL. *MySQL.org.* [En línea] [Citado el: 4 de 12 de 2008.] http://www.postgresql.org/docs/8.3/static/ddl-schemas.html.

18. InnoDB. *Sitio Web de InnoBase.* [En línea] [Citado el: 14 de Diciembre de 2008.] http://www.innodb.com/.

19. MyISAM. *Sitio Web de MySQL.* [En línea] http://dev.mysql.com/doc/refman/5.0/es/myisam-storage-engine.html.

20. Valgrind. *Valgrind Home.* [En línea] [Citado el: 23 de Enero de 2009.] http://valgrind.org/.

21. MySQL 5.0 Reference Manual :: Dimensiones máximas de las tablas MySQL. *MySQL :: MySQL 5.0 Reference Manual.* [En línea] http://dev.mysql.com/doc/refman/5.0/es/table-size.html.

22. Las principales características de MySQL. *MySQL 5.0 Reference Manual.* [En línea] [Citado el: 12 de Enero de 2009.] http://dev.mysql.com/doc/refman/5.0/es/features.html.

23. Conoce Firebird en 2 minutos. *Página de Noticias acerca de Firebird.* [En línea] http://www.firebirdnews.org/docs/fb2min_es.html.

24. Apache Derby. *Apache Derby.* [En línea] http://db.apache.org/derby/index.html.

25. Aclantis - Apache Derby Proyect Download Free Apache. *Apache Derby.* [En línea] http://www.aclantis.com/apache-derby-proyect-download-free-apache-art11730.html.

26. Compresión de datos en PostgreSQL. *Sitio Oficial de Postgres.* [En línea] [Citado el: 15 de Enero de 2009.] http://www.postgresql.org/docs/8.3/static/storage-toast.html.

27. Reglas en PostgreSQL. *Sitio Oficial de PostgreSQL.* [En línea] [Citado el: 12 de Enero de 2009.] http://www.postgresql.org/docs/8.3/interactive/rules-update.html.

28. PostgreSQL vs. MySQL vs. Commercial Databases: It's All About What You Need. *DevX.* [En línea] http://www.devx.com/dbzone/Article/20743.

29. Replicación en MySQL. *MySQL 5.0 Reference Manual.* [En línea] http://dev.mysql.com/doc/refman/5.0/es/replication.html.

30. PostgreSQL: Awards. *Sitio Oficial de PostgreSQL.* [En línea] http://www.postgresql.org/about/awards.

31. Tablespaces. *Sitio Oficial de PostgreSQL.* [En línea] http://www.postgresql.org/docs/8.3/static/manage-ag-tablespaces.html.

32. Oracle Corporation. *Oralce Developer Comunity.* [En línea] http://www.oracle.com/global/es/index.html.

33. Secuencias en PostgreSQL. *Sitio Oficial de PostgreSQL.* [En línea] [Citado el: 14 de Enero de 2009.] http://www.postgresql.org/docs/8.3/static/functions-sequence.html.

34. Porting from Oracle PL/SQL. *Sitio Oficial de PostgreSQL.* [En línea] http://www.postgresql.org/docs/8.3/static/plpgsql-porting.html.

35. **Rodíguez, Adonis y Vargas, Daniel.** *Diseño y Administración de Base de Datos.* La Habana : s.n., 2007.

36. Privilegios. *Sitio Oficial de Postgres.* [En línea] http://www.postgresql.org/docs/8.3/static/privileges.html.

37. Client Authentication. *Sitio Oficial de PostgreSQL.* [En línea] [Citado el: 13 de Enero de 2009.] http://www.postgresql.org/docs/8.3/static/client-authentication.html.

38. **Linstone, Harold A. y Turoff, Murray.** *The Delphi method: Techniques and applications.* s.l. : Addison-Wesley Pub. Co., Advanced Book Program (Reading, Mass.) , 1975.

Anexos

Anexo 1: Encuesta Realizada a los expertos

Compañero(a):

En la ejecución del presente trabajo de diploma, deseamos someter a la valoración de un grupo de expertos, la propuesta de Migración del diseño y administración de la base de datos del Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC 171) hacia PostgreSQL.

Para ello necesitamos primeramente conocer el grado de dominio que Ud. posee sobre Administración de Bases de Datos y con este fin deseamos que se sirva de responder la siguiente encuesta:

Nombre y Apellidos: _____

Centro de Trabajo: _____

Departamento: _____

Labor que realiza: _____

Años de Experiencia: _____

Categoría docente: _____ Categoría Científica: _____

1- Indique el grado de conocimiento que posee en materia de Administración de Bases de Datos en una escala de 1 a 10 proporcional el grado de conocimiento al valor numérico seleccionado:

1	2	3	4	5	6	7	8	9	10

2- Marque con una cruz (X) las fuentes que le han servido para argumentar el conocimiento que tiene Ud. de Administración de Bases de Datos. Marque con dos cruces (XX) la que más ha influido.

Fuentes de Argumentación	Grado de Influencia		
	Alto	**Medio**	**Bajo**
Análisis realizado por Ud.			
Experiencia			
Trabajos de autores nacionales.			
Trabajos de autores extranjeros			
Su propio conocimiento del tema.			
Su intuición.			

La propuesta del presente trabajo de diploma para la migración del diseño y administración de la base de datos del SIGESC 171, incluye:

- Definición de los pasos para realizar la migración.
- Definición de los Tablespaces.
- Creación de los esquemas y base de datos.
- Selección de los tipos de datos equivalentes.
- Transformación de las Tablas y Secuencias.
- Creación de esquemas por paquetes.
- Definición de roles de usuarios.
- Creación de políticas de salva y recuperación.

Para responder a las siguientes preguntas Ud. deberá haber leído el segundo capítulo de la presente tesis.

1. Según los detalles de la propuesta valore el grado de factibilidad de cada elemento proporcionado de acuerdo con la siguiente escala: Muy Adecuado (**MA**), Bastante Adecuado (**BA**), Adecuado (**A**), Poco Adecuado (**PA**), No Adecuado (**NA**).

Nº	**Elemento**	**Valoración**
1	Pasos para realizar la migración	
2	Definición de los Tablespaces	
3	Creación de los esquemas y base de datos	
4	Selección de los tipos de datos equivalentes	
5	Transformación de las Tablas y Secuencias	
6	Creación de esquemas por paquetes	
7	Definición de roles de usuarios	
8	Creación de políticas de salva y recuperación	

2. Determine si los elementos descritos son:

Necesarios: Si ____, No ____, No sé ____

Suficientes: Si ____, No ____, No sé ____

Óptimos: Si ____, No ____, No sé ____

Exprese otros criterios, observaciones o recomendaciones que pudieran servir para perfeccionar la propuesta realizada en el trabajo de diploma. (¿Qué modificar? ¿Qué agregar? ¿Qué eliminar? ¿Está correcta completamente? Etc.)

Anexo 2: Competencia entre expertos

	Preg 1	Preg 2									
No	**CON**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**Ka**	**Kc**	**K**	**Competencia**
1	8	0,3	0,4	0,05	0,05	0,05	0,05	0,9	0,8	0,85	ALTO
2	8	0,1	0,5	0,05	0,05	0,05	0,05	0,8	0,8	0,8	ALTO
3	8	0,1	0,4	0,05	0,05	0,05	0,05	0,7	0,8	0,75	MEDIO
4	5	0,2	0,4	0,05	0,05	0,05	0,05	0,8	0,5	0,65	MEDIO
5	6	0,2	0,2	0,05	0,05	0,05	0,05	0,6	0,6	0,6	MEDIO
6	10	0,2	0,5	0,05	0,05	0,05	0,05	0,9	1	0,95	ALTO
7	7	0,2	0,5	0,05	0,05	0,05	0,05	0,9	0,7	0,8	ALTO

Tabla 13: Coeficiente de competencia de los expertos.

Anexo 3: Componentes más importantes en un sistema PostgreSQL

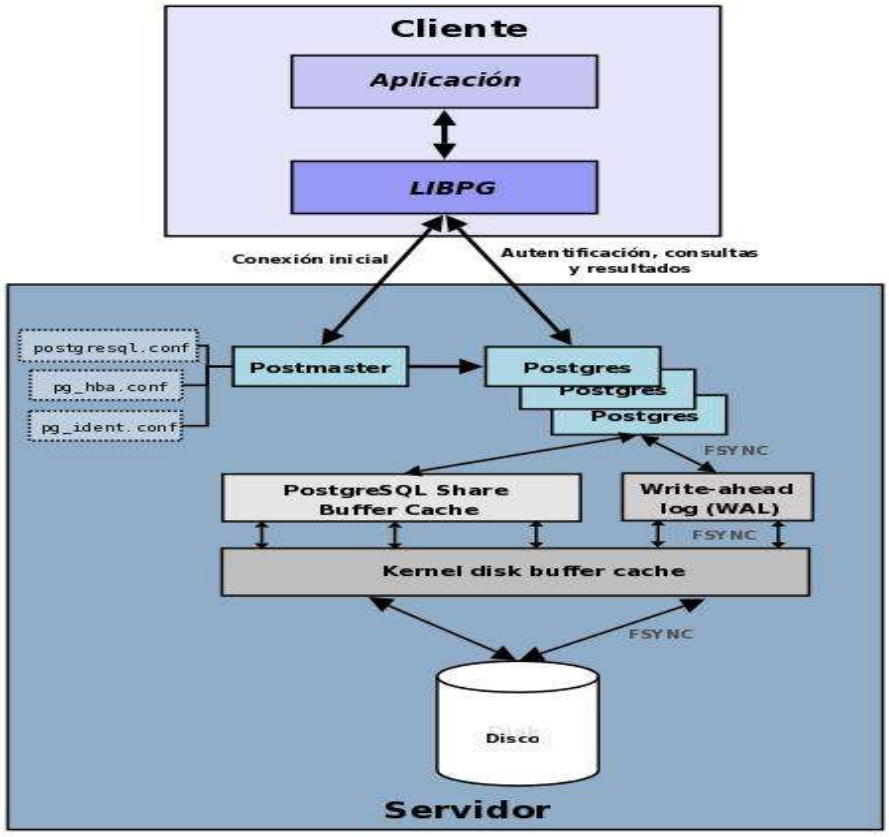

Figura 12: Componentes más importantes en un sistema PostgreSQL.

Glosario de Términos

Tuplas: en la teoría de bases de datos, una tupla se define como una función finita que mapea (asocia unívocamente) los nombres con algunos valores.

Normalización de una base de datos: El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

Argumentos (o parámetros): Un argumento o parámetro es una variable que puede ser recibida por una rutina.

Método (o implementación): un método consiste generalmente de una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regularán dicha acción y, posiblemente, un valor de salida (o valor de retorno) de algún tipo.

SELECT: La sentencia SELECT permite consultar los datos almacenados en una tabla de la base de datos.

CHAR (o VARCHAR): Los datos **char** o **varchar** pueden consistir en un único carácter o una cadena con un máximo recomendado de 8.000 caracteres**.**

BLOB: Los BLOB (**B**inary **L**arge **OB**jects, grandes objetos binarios) son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica. No todos los Sistemas Gestores de Bases de Datos son compatibles con los BLOB.

TEXT: Se utilizan para almacenar cadenas de caracteres superiores a 8000 caracteres o a 4000 en caso de caracteres Unicode.

Glosario de Términos

ACID: Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Así pues, si un sistema de gestión de bases de datos es ACID *compliant* quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID.

Llaves foráneas: Una llave foránea es un atributo (puede ser compuesto) de una relación cuyos valores deben de concordar con los de una llave primaria de alguna relación.

Benchmarks: Es una técnica utilizada para medir el rendimiento de un sistema o componente de un sistema, frecuentemente en comparación con el cual se refiere específicamente a la acción de ejecutar un benchmark. La palabra benchmark es un anglicismo traducible al castellano como *comparativa*. Si bien también puede encontrarse esta palabra haciendo referencia al significado original en la lengua anglosajona, es en el campo informático donde su uso está más ampliamente extendido. Más formalmente puede entenderse que un benchmark es el resultado de la ejecución de un programa informático o un conjunto de programas en una máquina, con el objetivo de estimar el rendimiento de un elemento concreto o la totalidad de la misma, y poder comparar los resultados con máquinas similares. En términos de ordenadores, un benchmark podría ser realizado en cualquiera de sus componentes, ya sea CPU, RAM, tarjeta gráfica, etc. También puede ser dirigido específicamente a una función dentro de un componente, por ejemplo, la unidad de coma flotante de la CPU; o incluso a otros programas.

NDB API: Motor de almacenamiento en memoria que ofrece alta disponibilidad y persistencia de datos. Es altamente configurable ofreciendo un gran número de opciones para manejar el balanceo de cargas y la tolerancia a fallas.

Slony: Sistema de replicación “maestro-esclavo” en cascada con tolerancia a fallos.

ODBC: es un estándar de acceso a Bases de datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué SGBD almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el SGBD, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el SGBD entienda. Para que esto funcione tanto la aplicación como el SGBD deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el SGBD debe ser capaz de responder a ellos.