ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ADMINISTRACIÓN Y PLANIFICACIÓN DE ÍNDICES BASADO EN LA DEMANDA DE OPERACIONES DE CONSULTA Y ACTUALIZACIÓN DE UN SISTEMA DE BASE DE DATOS"

TESIS DE GRADO

PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN COMPUTACIÓN

ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

ESPECIALIZACIÓN SISTEMAS DE INFORMACIÓN

Presentado por

Roberth Darío Chávez Jara

Juan Carlos Viscarra Jaramillo

Guayaquil - Ecuador

2007

AGRADECIMIENTO

Extendemos nuestros más sinceros agradecimientos a todos quienes hicieron posible la culminación de este trabajo.

DEDICATORIA

A nuestras familias, sin su apoyo nada de esto sería posible.

TRIBUNAL DE GRADO

_____

Ing. Holger Cevallos
PRESIDENTE DEL TRIBUNAL

_____

Ing. Fabricio Echeverría
DIRECTOR DE TESIS

_____

Ing. Mónica Villavicencio
VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Graduación de la ESPOL).

_____

Roberth Darío Chávez Jara

_____

Juan Carlos Viscarra Jaramillo

RESUMEN

Las bases de datos probablemente sean el software que más ha revolucionado el mundo de los negocios y la informática. Desde sus inicios cuando solo usaban archivos de texto para almacenar en formato plano los datos en un disco duro hasta los motores de la actualidad que soportan datos tales como XML, imágenes, etc; y utilizan las más sofisticadas soluciones de almacenamiento entre las cuales tenemos NAS (Network- attached Storage), arreglos de disco de fibra óptica, entre otros. Sin embargo, los motores de bases de datos han evolucionado debido a los constantes e insaciables requerimientos del mundo de los negocios en cuanto a información, día tras día surgen nuevos retos que demandan la búsqueda de nuevas formas de resolver los problemas que plantean las empresas y sus necesidades para almacenar y manejar el activo más importante hoy en día, información.

Sin embargo, a pesar de todos los avances en el campo de los motores de bases de datos, no todos pueden alcanzar la perfección y desarrollar un motor independiente en cuanto a mantenimiento y desempeño; los desarrolladores deben hacer un trade-off entre los puntos fuertes y débiles a atacar en el diseño de nuevos motores lo cual resulta en una amplia gama de productos de bases de datos que satisfacen las diferentes necesidades de los negocios. Uno de los puntos a atacar es el manejo de los índices, estos

objetos mejoran el tiempo de respuesta de las aplicaciones que usan las bases de datos proveyendo una herramienta que permite hacer consultas más rápidas usando un campo específico de una tabla. En empresas medianas y pequeñas esto no es un problema debido a que su modelo de negocios se traduce en una base de datos no muy compleja, lo cual es fácil de mantener y monitorear por el administrador de bases de datos; sin embargo en el caso de empresas grandes e inclusive algunas medianas, la forma de llevar el negocio implica sistemas que manejen grandes modelos de bases de datos lo cual significa muchas tablas y el DBA (database administrator o administrador de bases de datos) en ocasiones no se da abasto para monitorear todos los parámetros de desempeño del motor al mismo tiempo, lo cual significa que en ocasiones puede fallar en determinar si se deben crear o no ciertos objetos que pueden mejorar el tiempo de respuesta de la base de datos. El sistema de Monitoreo y Administración de Bases de Datos tiene como objetivo principal determinar el uso de índices en operaciones de consultas y actualizaciones para así predecir su uso a futuro y permitir que el DBA planifique estrategias basadas en planes de control de índices. Se planea utilizar métodos heurísticos para analizar los datos estadísticos que se colecten de la base de datos para conseguir los objetivos antes mencionados. Las implicaciones de este sistema serían maximizar la capacidad de monitoreo de desempeño del motor por parte del DBA y minimizar el impacto de un control inadecuado de los índices sobrecargando

al motor en horas pico lo cual aumentaría el riesgo de el sistema sufra una caída.

Nuestro sistema usará un escenario creado para probar la capacidad del sistema de predecir el uso de los índices en base a la demanda de operaciones de consulta y actualizaciones y así tomar las acciones requeridas para mejorar los tiempos de respuesta, además podría ser desarrollado para otros motores en un futuro, pues inicialmente será implementado para bases de datos en ORACLE.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La tecnología ha revolucionado el mundo actual y su ritmo de crecimiento parece no tener límite. Una de las herramientas que cambió drásticamente la forma de hacer negocios son las bases de datos, con las cuales llegaron nuevos conceptos tales como CRM, ERP, DSS, GIS, entre otros.

En la actualidad estos conceptos dejaron de ser eso y se han convertido en la herramienta con la cual muchas empresas hacen negocios y se mantienen en la competencia. Sin embargo, las bases de datos no pueden hacer todo el truco por si mismas. Es importante un buen diseño inicial y un monitoreo constante para poder obtener el mejor desempeño del motor de bases de datos reflejado en tiempos de respuesta muy cortos; la falta de monitoreo sobre estos objetos puede resultar en la degeneración del desempeño de las aplicaciones vitales para el negocio.

Nuestro sistema, MAEBD (Monitoreo y Análisis Estadístico de Base de Datos), se encarga de monitorear la demanda de uso de tablas basado en estadísticas y utilizando métodos heurísticos podemos predecir los índices que deberían crearse para mejorar el desempeño de la base de datos.

CAPÍTULO 1

1. INFRAESTRUCTURA DE UN SISTEMA DE ADMINISTRACIÓN DE BASE DE DATOS Y TECNOLOGÍAS QUE SE INCORPORAN EN LA MEJORA DEL ALGORITMO

1.1. Sistemas de Administración de Bases de Datos

"Los motores de bases de datos, gestores de bases de datos o sistemas administradores de bases de datos (les llamaremos **DBMS** de ahora en adelante) surgieron en los años sesenta a partir de los sistemas de archivos cuyo desarrollo se llevaba madurando desde los años cuarenta, los cuales utilizaban archivos planos para almacenar datos con funciones básicas de lectura y escritura sobre ellos" [1].

"Para ser precisos, en el año 1964 se conciben los primeros DBMS como un intento para dar un viraje a los sistemas de archivos que se limitaban a la estructuración del almacenamiento físico de los datos. Junto con su concepción, surge el concepto de administración de datos que mediante actividades integradas permitía verlos físicamente en un solo bloque, pero lógicamente podían ser manipulados a través de un esquema compuesto de estructuras donde se establecen vínculos de integridad, métodos de acceso y organización física sobre los datos, permitiendo así obtener valores agregados de utilización tales como: manejo de usuarios, seguridad, atomicidad e independencia física y lógica de los datos" [1].

“Conforme evolucionaban los lenguajes de programación, los DBMS se abrían paso, y es así que en 1 970 se genera una nueva noción que establecía que las tablas en las bases de datos debían representarse mediante una estructura tabular llamada relación o tabla, formada por filas y columnas. De esta manera a finales de los años setenta y bajo este concepto surgen las primeras **DBMS relacionales** (a las cuales llamaremos RDBMS de ahora en adelante). En los años ochenta se siente la necesidad de que los RDBMS manipulen objetos complejos y estructuras como las usadas en CAD (Computer-Aided Design) y CASE (Computer-Aided Software Engineering); a partir de esto surgen dos tendencias que aún siguen siendo estudiadas y desarrolladas en la actualidad.” [2]

La primera tendencia da origen a los **DBMS objeto-relacionales**, a los cuales llamaremos ORDBMS en adelante, los cuales se proyectan como una extensión de los RDBMS orientados hacia el paradigma orientado a objetos.

La segunda de estas tendencias es aún motivo de estudio y aún no ha alcanzado su cúspide y originó los **DBMS orientados a objetos**, a los cuales llamaremos OODBMS en adelante, y que integran herramientas

para almacenamiento y manipulación de clases, los objetos, la asociación entre ellos y sus métodos.

1.1.1. Estructura y Elementos de un Sistema de Administración de Base de Datos

Los DBMS pueden ser descritos como programas que se encargan de manejar las peticiones de creación y acceso a datos de los usuarios (de un sistema o aplicación que utilice una base de datos) a una base de datos, de tal manera que éstos no tengan que entender como están almacenados los datos físicamente en los medios de almacenamiento; puede ser visto como un administrador de archivos que maneja datos en las bases de datos en lugar de archivos en un sistema de archivos. Podemos resumir lo anterior diciendo que sus principales funciones son:

- Manejar datos estructurados
- Manejar las peticiones de los usuarios:
  - Aceptar consultas de los usuarios
  - Responder estas consultas

En lo que respecta a manejar datos estructurados, los DBMS tienen almacenados físicamente los datos de tal manera que si abrimos los archivos de datos, lo único que veremos será una masa de letras, números y símbolos que para nosotros no tiene ningún sentido pero el DBMS es capaz de interpretar estos datos y presentarlos ante nosotros en forma de registros, datos relacionados que tienen algún sentido. Pero esto lo hace con el conocimiento de un archivo especial, el archivo de Metadata, en el cual se encuentra descrita la estructura de los datos almacenados, lo cual permite reconstruir el esquema que inicialmente los usuarios implementaron y mostrar la información necesitada.

En cuanto al manejo de las peticiones de los usuarios, se destacan dos aspectos muy vitales: *la integridad de los datos*, lo cual asegura que estos continuarán disponibles y consistentemente organizados como se espera; y *la seguridad*, la cual garantiza que sólo aquellos con privilegios de acceso puedan manipular los datos.

La estructura y los elementos de un DBMS pueden variar dependiendo del motor, pero la arquitectura básica bajo la cual

se han implementado la mayoría de los DBMS se basa en un modelo de capas:

Figura 1.1 - Arquitectura básica de un DBMS – Modelo de capas [3]

Para definir las funciones de cada una de las capas del modelo de arquitectura básica utilizaremos un DBMS modelo. PostgreSQL, uno de los DBMS de código abierto más utilizados por su robustez y eficiencia en aplicaciones de tamaño medio,

está diseñado utilizando el modelo mencionado anteriormente. Su estructura es la siguiente:

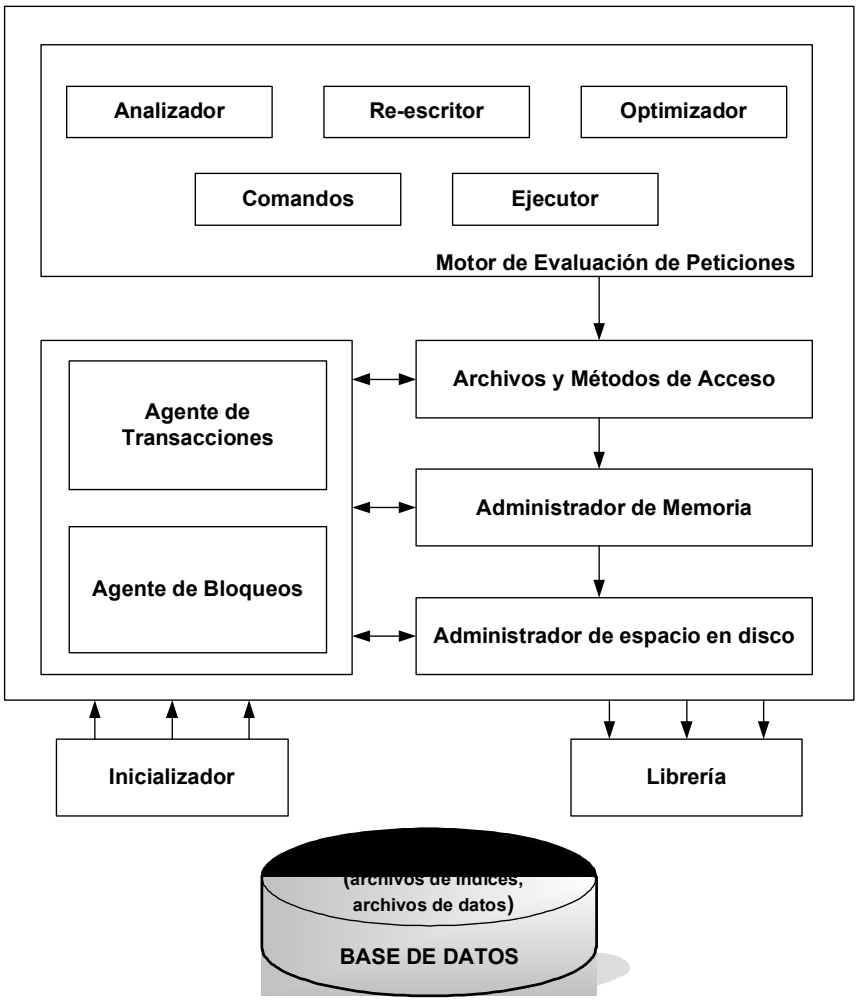


Figura 1.2 - Arquitectura de PostgreSQL – Modelo de capas [4]

1. **Motor de Evaluación de Peticiones:** Esta capa se encarga de tomar las peticiones (queries) de los usuarios ingresados desde una aplicación, generar el plan de ejecución y ejecutar estos planes contra de la base de datos.

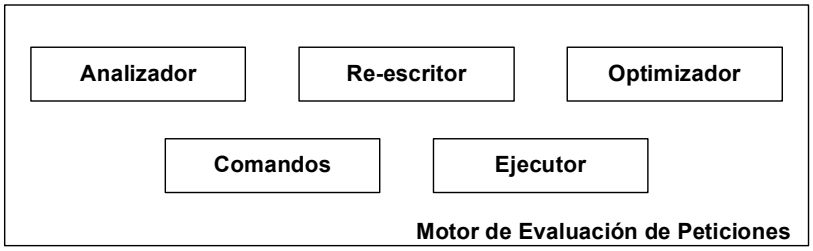


Figura 1.3 - Motor de Evaluación de Peticiones [4]

Está formada por otros subsistemas que se encargan de manejar el motor de optimización y generación de plan de ejecución, lo cual divide las funciones entre los diferentes subsistemas haciendo menos compleja la tarea de esta capa:

a) **El subsistema Analizador**: chequea la petición (cadena de caracteres) ingresado desde la aplicación. Verifica la sintaxis del mismo, y si es

correcta convierte la petición en un árbol (árbol de petición).

b) **El subsistema Re-escritor**: Éste toma como entrada el árbol generado en el subsistema anterior y busca cualquier regla a aplicar en el árbol; además lo transforma en lo que pueda ser usado en futuras optimizaciones y ejecuciones. Este subsistema es vital en la realización de las vistas.

c) **El subsistema Optimizador**: Este subsistema es el corazón del motor de optimización; toma como entrada el árbol re-escrito en el proceso anterior como entrada y crea todos los posibles métodos de ejecutar la petición. Todos estos métodos llevan al mismo resultado, pero evalúa cada uno de ellos para encontrar el mejor plan de ejecución.

d) **El subsistema Ejecutor**: Este subsistema se encarga de ejecutar la petición resultado del proceso de optimización.

e) **El subsistema Comandos**: Este subsistema toma el árbol generado por el subsistema Analizador y procesa los comandos SQL que no necesitan ser

optimizados tales como *copy, alter, create table, create view*, etc.

2. **Archivos y Métodos de Acceso:** Su principal función es dar soporte al concepto de archivo, que en el caso de un DBMS es una colección de registros o páginas. Además soporta los índices.
3. **Administrador de Memoria**: Esta capa controla el uso de memoria, trae las páginas del disco a memoria principal de acuerdo a como se necesitan en base a las peticiones de los usuarios. Actúa de forma similar al buffer de la memoria virtual (del sistema operativo local) pero las políticas de reemplazo se basan en más métricas y criterios.
4. **Administrador de Espacio en Disco**: Esta capa se encarga de la administración de datos en el medio físico. Maneja el concepto de página de manera que las capas superiores pueden asignar, reasignar, leer y escribir páginas a través de esta capa. El tamaño de la página es un asunto crucial, pero puede ser ajustado de manera que una lectura o escritura de una página pueda ser hecho en una operación E/S (Entrada/Salida) de disco.

Existen 4 componentes extra en la estructura de PostgreSQL que no son contemplados en el modelo de capas estándar: el *Agente de Transacciones, el Agente de Bloqueos, el Inicializador y la Librería*.

a) **El subsistema Agente de Transacciones**: Se encarga de la que las peticiones y liberaciones de transacciones se bloqueen de acuerdo un protocolo de bloqueo adecuado, además planifica la ejecución de transacciones.
b) **El subsistema Agente de Bloqueos**: Este subsistema se encarga de llevar un seguimiento de las peticiones de bloqueo y restricciones de bloqueo. Coopera con el subsistema anterior para proveer control de concurrencia, rollback y recuperación por fallas. Ambos constituyen el subsistema de Concurrencia (Concurrency), con el cual interactúan las 3 capas inferiores (*Archivos y Métodos de Acceso, Administrador de Memoria y Administrador de Espacio en Disco*) cuando los datos son accesados.
c) **El subsistema Inicializador**: Es el encargado de administrar y controlar los otros subsistemas.

d) **El subsistema Librería**: De manera similar que el kernel (núcleo) de un SO, este contiene y ejecuta rutinas utilizadas en el DBMS.

Un sistema de datos básicamente es un conjunto de estructuras relacionadas de una manera lógica de acuerdo a un modelo de negocios específico, el cual maneja reglas generales y propias del sistema. Este conjunto de estructuras guarda información valiosa acerca del modelo que se desea manejar.

Estos sistemas abstraen la lógica de un negocio dentro de todo este conjunto de estructuras, como tablas, índices, claves primarias, entre otros objetos. La integridad de toda esta información depende tanto del motor que la maneja y del sistema operativo que lo contiene, en la actualidad existen muchas maneras de salvaguardar los datos que contienen estas estructuras.

Antiguamente una base de datos era simplemente un sistema de ficheros en el que se proponía tratar de informatizar los archivadores manuales a fin de tratar de dar un acceso más eficiente a los datos, sin embargo el primer problema que surgió con estos ficheros era la gran cantidad de datos redundantes que existían lo que provocaba un manejo ineficiente de los recursos del sistema, otro inconveniente que tenían estos sistemas de ficheros era que al no poseer un mecanismo de relacionar estos archivos entre sí, cada fichero funcionaba como un programa independiente que define y maneja sus propios datos. Esto complicaba el poder integrar en una sola aplicación diferentes módulos del sistema. Con el pasar del tiempo estas aplicaciones necesitaban crecer para poder expandir el campo de negocio, lo cual trajo más problemas en la eficiencia de los tiempos de consultas especialmente.

A continuación se detallan los principales problemas de los sistemas de ficheros:

- Datos redundantes.
- Dificultad de crear un sistema integrado.
- Largos tiempos de espera para realizar una consulta.

- Se necesitan crear complejos programas de aplicación para poder responder a cualquier tipo de consulta de datos, incluso para consultas simples.
- La independencia de datos es mínima.
- No existe un fundamento teórico que respalde la eficiencia de estos sistemas.

"A mitad de los sesenta, se desarrolló IDS (Integrated Data Store), de General Electric. Este trabajo fue dirigido por uno de los pioneros en los sistemas de bases de datos, Charles Bachmann. IDS era un nuevo tipo de sistema de bases de datos conocido como *sistema de red*, que produjo un gran efecto sobre los sistemas de información de aquella generación. El sistema de red se desarrolló, en parte, para satisfacer la necesidad de representar relaciones más complejas entre datos que las que se podían modelar con los sistemas jerárquicos, y, en parte, para imponer un estándar de bases de datos. Para ayudar a establecer dicho estándar, CODASYL (Conference on Data Systems Languages), formado por representantes del gobierno de EEUU y representantes del mundo empresarial, formaron un

grupo denominado DBTG (Data Base Task Group), cuyo objetivo era definir las especificaciones estándar que permitieran la creación de bases de datos y el manejo de los datos. El DBTG presentó su informe final en 1 971 y aunque éste no fue formalmente aceptado por ANSI (American National Standards Institute), muchos sistemas se desarrollaron siguiendo la propuesta del DBTG. Estos sistemas son los que se conocen como sistemas de red, o sistemas CODASYL o DBTG.” [5]

“En 1 970 Codd, de los laboratorios de investigación de IBM, escribió un artículo presentando el *modelo relacional*. En este artículo, presentaba también los inconvenientes de los sistemas previos, el jerárquico y el de red. Entonces, se comenzaron a desarrollar muchos sistemas relacionales, apareciendo los primeros a finales de los setenta y principios de los ochenta. Uno de los primeros es System R, de IBM, que se desarrolló para probar la funcionalidad del modelo relacional, proporcionando una implementación de sus estructuras de datos y sus operaciones. Esto condujo a dos grandes desarrollos:

- El desarrollo de un lenguaje de consultas estructurado denominado SQL, que se ha convertido en el lenguaje estándar de los sistemas relacionales.
- La producción de varios SGBD relacionales durante los años ochenta, como DB2 y SLQ/DS de IBM, y ORACLE de ORACLE Corporation.

Hoy en día, existen cientos de SGBD relacionales, tanto para microordenadores como para sistemas multiusuario, aunque muchos no son completamente fieles al modelo relacional.” [5]

Al hablar de un sistema de base de datos, no sólo nos referimos a la aplicación que adquirimos para instalar en nuestra infraestructura de hardware, el concepto va más allá en la actualidad y podemos identificar 4 elementos claves que intervienen directamente con el desempeño de las bases de datos:

- Estructuras (tablas, índices, vistas, etcétera)
- Motor de Base de Datos en sí
- Lenguaje de Consultas

- Modelo Relacional de Negocio

Estructuras de un Sistema de Base de Datos

Dentro de las estructuras que intervienen en un sistema de base de datos se destacan:

- Tablas
- Índices
- Vistas
- Procedimientos Almacenados
- Cursores y Buffers

En ciertas bases de datos existen otras estructuras creadas para algún uso en particular que son propias para la base de datos, un ejemplo de aquello es Oracle, en la cual existe una estructura llamada sinónimo el cual permite básicamente poder ver tablas de otros usuarios o de otra instancia.

Tablas

Las tablas en una base de datos son el elemento principal y además son la razón de existir de la base de datos, ya que las tablas son estructuras que almacenan datos relacionados.

Están formadas por filas y columnas; las columnas se conocen como atributos y representan una porción de información o dato de tipo numérica, alfanumérica u otros, mientras que las filas se denominan registros y están formados por grupos de información que unidos representan un ente cualquiera. (Las tablas tienen como objetivo principal abstraer un modelo del mundo real al mundo de la Computación)

Índices

Los índices son un tipo especial de objeto que utilizan las bases de datos para mejorar su rendimiento global. Los índices se crean usando una o más columnas de la tabla y contienen siempre menos columnas que la tabla, y son implementados con una estructura especial de datos (por lo general árboles B+ y hashing) para optimizar su función: mejorar el tiempo de respuesta de las consultas (acceso a datos).

Se reconocen dos tipos de índices: arracimados y no arracimados. Los índices arracimados son únicos por tabla y por lo general están constituidos por la clave primaria de dicha tabla y guardan en su estructura los datos que están indexando. Mientras que las tablas pueden tener asociados uno o más índices no arracimados, los cuales se componen de columnas

que no forman la clave primaria de la tabla o columnas que forman la clave primaria más otras columnas y guardan en su estructura un identificador de la ubicación física del registro asociado, es decir no guardan los datos que están indexando.

Existen muchos factores que se deben tomar en cuenta al crear índices, primero debemos considerar dichos factores en una etapa de diseño; índices mal diseñados degradan el motor y aumentan el tiempo de respuesta, mientras que un buen diseño mejora la eficiencia del motor al acceder a los datos de manera que entrega un buen rendimiento conjuntamente con las aplicaciones que usan la base de datos.

Posteriormente se hablará de dichos factores y otras cuestiones de diseño que deben ser tomadas muy en cuenta a la hora de crear índices.

Vistas

Las vistas son estructuras que están basadas en una sentencia SQL de consulta, la cual puede incluir una o más tablas. El objetivo final de las vistas es tener una consulta preconstruida para de alguna forma tener un acceso más rápido a los datos. Al igual que lo índices las vistas consumen espacio físico en el

servidor de la base de datos lo cual nos indica que el uso excesivo degrada el rendimiento.

Procedimientos Almacenados

Los procedimientos almacenados surgieron debido a la necesidad de poder ejecutar pequeñas aplicaciones y/o procesos directamente sobre las tablas que contienen los datos, estos procedimientos debido a que se encuentran alojados en el servidor de base de datos y se ejecutan en el motor de base de datos, proveen de mayor eficiencia a la aplicación. El único problema de estos procedimientos almacenados es el hecho de que cada motor diferente tiene su propio lenguaje para estos procedimientos, lo cual nos dice que no es un estándar definido y esto lleva a que los desarrolladores deban aprender diferentes lenguajes en los diferentes motores de base de datos, lo cual cuesta tiempo y dinero.

Motor de Base de Datos

El motor de la base de datos es el elemento fundamental de un sistema de base de datos, ya que aquí se realizan las operaciones de optimización y procesamiento de las sentencias

SQL de la misma, incluso podríamos decir que el motor de una base de datos es el corazón de la misma.

Lenguaje de Consultas

Una base de datos no es más que una aplicación que almacena datos de algún programa específico, pero debe existir alguna forma de poder consultar estos datos almacenados. El lenguaje para todas las bases de datos en la actualidad es el SQL (Structure Query Language), este lenguaje permite a los usuarios tener la capacidad de realizar consultas de los datos, crear más datos, modificarlos o eliminarlos. En la actualidad existe un estándar establecido para este lenguaje, el ANSI/SQL2 que es reconocido oficialmente desde 1 990; a pesar de esto cada motor posee ciertas extensiones de este lenguaje que son diferentes para cada distribución.

Modelo Relacional de Negocio

Son las tablas, relaciones, restricciones y otras consideraciones que se toman en cuenta para crear la base de datos y están

alineados de acuerdo al negocio que la empresa realiza y refleja todos los aspectos del mismo. En este proceso se toman en cuenta las reglas de normalización, desnormalización y otros factores que determinan el desempeño del motor sirviendo a los requerimientos que se hacen a la base de datos.

1.1.2. Factores críticos que influyen en el desempeño de un motor de base de datos

El primer y más importante paso para obtener una base de datos con un buen desempeño es sin duda el diseño, sin importar el tamaño en registros de nuestra base de datos lo más importante siempre será tener un diseño apropiado. La mejor manera de obtener un buen diseño es desde el inicio pensar en la posible demanda que pueda tener nuestra base de datos.

Un punto importante en el diseño de una base de datos es tener claro la información realmente necesaria. Recordemos que cada campo que creamos en nuestro diseño es para satisfacer alguna necesidad de nuestro modelo de negocios, y siempre

teniendo en cuenta que no hay que almacenar ni más ni menos, solamente lo necesario.

Otro dilema para el diseñador de una base de datos es al momento decidir si almacenar o no un valor que puede ser calculado en base a otros campos de la tabla. Es muy conocido en nuestro medio que almacenar un campo que es el resultado de alguna fórmula en la que intervienen otros campos de una tabla no es recomendable, ya que estaríamos creando algo que conocemos como redundancia de datos de una manera indirecta. Pero llevando esto de la práctica a la realidad con el pasar del tiempo nos daremos cuenta que en cierto casos no es bueno ser estrictos con esta regla. Algo que debemos tomar en cuenta al momento de tomar esa decisión es la frecuencia con el que ese dato va a ser necesario calcularlo, porque como debe ser conocimiento para todos el hecho de realizar algún cálculo es una carga más para el procesador, y si por ejemplo ese valor será calculado con una frecuencia tal que tiende a ser muy parecida a la cantidad de registros de una tabla grande es muy probable que estar calculando ese valor por cada registro resulte ineficiente y que muchas veces estemos pensando crear algún mecanismo de hacer más eficiente esa consulta ya sea

usando índices o vistas, cuando el problema de fondo es el estar calculando ese valor con mucha frecuencia.

Otro gran error que suelen cometer al momento de diseñar es que se crean las tablas con muchos campos pensando que pueden hacer falta en un futuro, pero luego en la práctica resulta ser que esos campos nunca se usan lo cual concluiría en graves problemas al momento de retornar una consulta, y esto se debe a que muchas veces los requerimientos de las aplicaciones suelen cambiar y las consultas tienden cada vez mas a ser algo como:

```
SELECT * FROM TABLE
```

Y a medida que las tablas incrementen sus registros los problemas no tardarán en aparecer, en especial los problemas de tiempos de respuestas. Algo importante que se debe tener en cuenta para evitar caer en estas malas prácticas de diseño es tener siempre presente que hacer un cambio en una tabla es algo simple, lo que si deberíamos tener cuidado es en hacer las relaciones correctamente desde el inicio ya que esto si resulta muchas veces tedioso cambiar.

Finalmente, un aspecto importante que debemos tener en cuenta al momento de diseñar una base de datos son los tipos de datos. Para muchos quizás resulte sin importancia tener un campo numérico o un campo de texto, cuando en el fondo para el motor muchas veces resulta una carga tener que realizar conversiones internas de tipos de datos para realizar alguna comparación o cálculo. Una manera para no caer en estos errores es usar campos numéricos en los casos que se requieran hacer cálculos con esos campos o en caso de que esos campos resulten ser de algún PK de un secuencial. Por ejemplo es común ver que en algunos diseños de una base de datos se pueden encontrar campos que representan el teléfono de alguna persona y que ese campo sea de tipo numérico, cuando en la práctica nunca se realice ningún tipo de cálculo con ese campo.

1.2. Proceso de administración de un Sistema de Administración de Base de Datos

En muchos de los caso es común ver que los afinamientos de las bases de datos son hechos en base a un comportamiento general el cual se supone el motor va a estar expuesto, sin embargo, en la realidad este comportamiento resulta un poco variable con el pasar del tiempo. Normalmente este trabajo de afinamiento es hecho por un Administrador de Bases de Datos (DBA), el cuál no siempre puede estar dedicando su tiempo en analizar como puede mejorarse el tiempo de respuesta de las consultas al motor de una base de datos.

Los afinamientos de los motores de bases de datos son hechos en base a un comportamiento inicial esperado, pero con el pasar del tiempo y luego de que el sistema sufra algunos cambios y modificaciones los datos, que surgen a raíz de nuevos requerimientos en las aplicaciones, en casi todos los casos los DBAs se olvidan de realizar un nuevo análisis del desempeño del motor. En otros casos, resulta muy difícil realizar un análisis real de cómo poder mejorar el desempeño, ya sea porque el modelo de la base de datos ha crecido demasiado o porque luego de ciertas modificaciones este modelo resulta muy complejo de analizar.

Una razón muy clara que nos responde a la pregunta de ¿por qué nuestro sistema puede resultar ineficiente en ciertos casos?, es porque las personas que desarrollan generalmente no son DBAs y en tiempos

de desarrollo se olvidan que los datos que tienen de prueba no son reales y nunca se analizan bien las consultas y esto representa un elevado costo en los tiempos de respuesta del motor. Debemos tener claro que para cada problema planteado en el cual necesite de una consulta al motor, puede existir un conjunto finito de consultas que nos retornan el mismo resultado, y puede ser que con los datos de prueba, las consultas puedan parecer iguales, pero cuando son probadas con la cantidad de datos reales las soluciones a estos problemas tienen diferentes tiempos de respuesta.

Otros factores que se deben tener en cuenta es el bloqueo de las tablas en la base de datos y la saturación de los recursos que puedan surgir en tiempos de producción. Por esto, es importante también resaltar que se debe tener muy en cuenta los recursos que dispone nuestro motor de base de datos.

Es muy común ver que cuando se pierde rendimiento en nuestro motor lo primero que se puede pensar es en aumentar o cambiar recursos con los que contamos, cuando lo primero que deberíamos hacer es hacer un análisis detenido para buscar el origen de la pérdida de rendimiento de nuestro motor. En un 80% de estos casos puede ser que nuestro problema de recursos sea resuelto con una consulta bien analizada e implementada con respecto al desempeño, y que quizás nuestro

problema sea resuelto con la simple creación de algún índice para evitar que el motor realice un FULL TABLE SCAN.

Los problemas generados por el bajo desempeño que puede alcanzar nuestro motor no deben ser vistos como algo sin importancia, puesto que este tipo de cosas puede afectar la imagen de la empresa dueña de esa base de datos. Y esto puede desencadenar a que el sistema sea dado de baja.

Un ejemplo práctico para entender mejor un problema de desempeño, es cuando los desarrolladores abusan con el uso de cursores. Este problema se debe a que el motor no está pensado ni hecho para manejar los registros de una tabla fila a fila, sino que el motor los ve como un conjunto de datos, y esto puede desencadenar un deterioro de nuestro motor. Así como se advierte que el exceso en el uso de cursores no es recomendado lo mismo hay que tener en cuenta al momento de crear índices.

Teniendo en cuenta todo esto que interviene en el diseño y afinamiento de una base de datos, vemos la necesidad de poder tener a la mano una herramienta que ayude a los DBAs a realizar un mejor afinamiento a la base de datos, y así poder crear un mejor plan de optimización de una bases de datos en base a la creación o eliminación de índices de acuerdo a si son o no usados. Posteriormente nombraremos los casos

en los cuales los índices a pesar de estar creados no son usados, y esto puede ser ocasionado a consultas mal estructuradas o mal diagramadas.

1.2.1. Proceso de Afinamiento de un Motor de Base de Datos

"La afinación de una bases de datos se lo conoce como Tuning, que no es mas que el proceso de revisar/ajustar el diseño físico de una bases de datos en base al monitoreo de los recursos utilizados.

Las metas que se proponen en un afinamiento son:

- Hacer la aplicación más rápida.
- Disminuir los tiempo de respuesta de las consultas/transacciones
- Mejorar la velocidad de transmisión de los datos en las transacciones." [6]

"La administración correcta del motor de base de datos está determinada por algunos factores importantes. Uno de los factores es el *tamaño* de la base de datos, dentro de esto hay

que tomar en cuenta que el modelo debe estar correctamente normalizado para de esta manera no tener que redundar información.

Un segundo factor que podríamos nombrar es tratar de estimar que tan rápido va a crecer la base de datos, o sea tener en cuenta la *tasa de crecimiento* de nuestra base de datos, ya que eso varía de acuerdo al negocio.

Estimar la *cantidad de usuarios* que se van a conectar a la base de datos, influye también en el desempeño de la misma, ya que no es lo mismo una bases de datos a la que acceden 100 usuarios máximo a una en que accedan mas de 10000 usuarios, para el segundo caso la afinación de las bases de datos no debería estar centralizada solamente en los puntos anteriores, sino también tomar en cuenta que existen muchos usuarios que pueden estar conectados en un instante dado, lo cual sugiere que la posibilidad de que existan bloqueos aumente.” [7]

Finalmente, un punto importante y que interviene directamente con el desempeño de la bases de datos es el tipo de *hardware* sobre el que funciona el motor, para ello debemos tener en cuenta que el servidor sobre el cual debe estar montada la bases de datos no debe ser lo qué es realmente necesario, esto quiere decir que el equipo debe ser lo suficientemente capaz de responder a las necesidades de la aplicación y más. En algunos casos, es necesario contar con un servidor de dos o más procesadores para poder atender adecuadamente a los requerimientos de las aplicaciones.

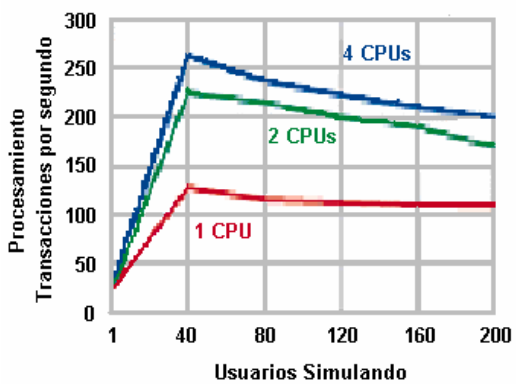


Figura 1.4 – Comparación del rendimiento de un computador con múltiples procesadores. [8]

En la actualidad ya se puede contar con procesadores de doble núcleo (Dual Core), los cuales son microprocesadores en los cuales hay dos procesadores (físicos) independientes en el mismo encapsulado. [8]

1.2.2. Objetivos e importancia de la Administración de un Motor de Base de Datos

Uno de los objetivos principales de una correcta administración de una bases de datos es el poder responder adecuadamente a las exigencias de los usuarios de poder acceder de una manera rápida y eficiente a la información, para lo cual existen algunos mecanismos que nos ayudan en mejorar los tiempo de búsqueda como son los índices, pero sin olvidar que como cualquier cosa usarlos en exceso deterioran el desempeño global de la bases de datos.

1.3. Métodos heurísticos aplicados a la solución de problemas

"Existen muchos problemas de optimización que no pueden ser abordados mediante métodos exactos, ya sea por su alto grado combinatorio o por la dificultad de generar un modelo basado en programación matemática que represente exactamente una situación real. Para hacer frente a este tipo de situaciones, se han venido desarrollando desde los años sesenta una serie de métodos conocidos como **heurísticos**, los cuales han probado ser una de las mejores

soluciones para los problemas de complejidad computacional elevada." [9]

Los métodos heurísticos son métodos que utilizan funciones matemáticas para encontrar la solución de mejor calidad a problemas de optimización con escenarios muy difíciles de representar mediante las técnicas habituales, sin embargo, la solución que se encuentra mediante su aplicación no es la mejor, debido a que se corre el riesgo de no encontrar la solución óptima.

"Inicialmente estos métodos fueron desarrollados para enfrentar problemas de fácil planteamiento y representación, pero de difícil solución tales como: *el problema del agente viajero, el problema de la mochila, el problema de los conjuntos de cobertura*, entre otros. A mediados de los años ochenta, estos métodos evolucionaron de manera que podían ser utilizados en problemas que provenían de diversos sectores y no sólo en aquellos en los que fueron inspirados; dando surgimientos a otro tipo de métodos llamados **meta-heurísticos**." [9]

Los métodos meta-heurísticos tienen como objetivo alcanzar mejores resultados que los heurísticos tradicionales y surgen como una confluencia de distintas áreas del conocimiento como: Matemáticas, Biología, Ingeniería, entre otras. Su origen es el resultado de la fusión de los métodos de optimización con las técnicas de Inteligencia Artificial.

“Los procedimientos meta-heurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los meta-heurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la Inteligencia Artificial, la Evolución Biológica y los mecanismos estadísticos. Los métodos heurísticos además incluyen un gran componente en la solución de problemas: el conocimiento. Algunos de los métodos heurísticos pueden “aprender” y aplicar este conocimiento en la resolución de problemas más complejos disminuyendo el tiempo de búsqueda de la solución.” [10]

En la actualidad, los métodos meta-heurísticos han sobrepasado los heurísticos tradicionales mostrando mayor eficiencia en buscar la mejor

solución a un problema dado; sin embargo, dado que su base radica en los heurísticos tradicionales, los llamaremos en adelante simplemente métodos heurísticos. Entre los métodos más conocidos tenemos los siguientes:

- Optimización por Colonia de Hormigas
- Algoritmos Evolutivos
- Algoritmos Genéticos
- Programación Genética
- Redes Neuronales
- Búsqueda Tabú [9]

Los métodos heurísticos son una amalgama de muchas áreas de las ciencias, es por esto que utilizan funciones para determinar el grado de aceptabilidad que una solución tiene. Esta función se conoce como **función heurística** y permite la evaluación de estados individuales del problema, es decir, es usada como indicativo para saber si el proceso está en su curso correcto en busca de la mejor solución.

La función heurística perfecta guía directamente a la solución, sin embargo el costo de calcularla en ocasiones es más grande que el costo requerido para encontrar la solución, por lo cual se prefiere encontrar una función que equipare ambos costos.

1.3.1. Aplicaciones potenciales

> "En el campo de las bases de datos, los métodos heurísticos son utilizados en la implementación de bases de datos distribuidas. Dentro del diseño de bases de datos distribuidas, el diseño de la distribución de datos, fragmentación y asignación es un aspecto crítico y es lo que marca la diferencia del diseño de una base de datos no distribuida. El problema involucra 2 fases, de las cuales, en una se utilizan métodos heurísticos; la primera es la agrupación de fragmentos de datos y la segunda, es la ubicación eficiente de estos grupos de fragmentos en la red. En la primera fase se utilizan algoritmos heurísticos exitosos en la resolución de problemas de partición tales como el Algoritmo de Particionamiento Vertical Binario o el Algoritmo de Agrupamiento Binario de Fragmentos. La segunda fase necesita el resultado de la aplicación de dichos algoritmos pues

se utilizan como variables de entrada para los algoritmos de la segunda fase.” [11]

Otra de las aplicaciones de estos métodos es la minería de datos, la cual trata sobre la inducción de conocimiento útil a partir de masas ingentes de datos. Su aplicación más utilizada es el estudio de información almacenada por las empresas en repositorios especializados para proyectos de este tipo, en los cuales se almacena datos potencialmente útiles de diversas áreas del negocio para el reconocimiento de patrones que permitan extraer conocimiento y aplicarlo para mejorar la rentabilidad, el nivel de servicio, redefinir estrategias de marketing, etc. Los algoritmos más utilizados en estas aplicaciones son los algoritmos de agrupamiento en todas sus variantes. [12]

1.3.2. Ventajas y desventajas

Existen muchas ventajas que los métodos heurísticos tienen sobre los métodos generales de optimización y búsqueda, sin embargo se considerada a las siguientes como las más importantes:

- Tiempo de búsqueda de la solución
- Modelamiento
- Conocimiento

En lo que respecta al tiempo de búsqueda de la solución, los métodos heurísticos convergen en menor tiempo que los métodos generales de optimización, lo cual los hace muy atractivos. En ocasiones, la diferencia de tiempo puede ser abismal, pero por lo general dicha diferencia se encuentra en un rango aceptable. En muchos negocios, la velocidad a la que se resuelven problemas de planeamiento y optimizaciones es la clave fundamental de su supervivencia por lo cual este tiempo es una ventaja fundamental.

En cuanto al modelamiento, los métodos heurísticos permiten definir representaciones de las variables de entrada de tal forma que se consiga el efecto que estas tendrían en un escenario real, lo cual es de gran utilidad en el caso de simulaciones de aplicaciones en las cuales los errores pueden ser fatales.

En cuanto al conocimiento, como se dijo anteriormente estos métodos pueden adicionar un componente extra a sus criterios de búsqueda, el conocimiento, con el cual la obtención de una solución óptima se garantiza.

Entre las desventajas más notorias de estos métodos tenemos que la mayoría de estos métodos no miden la calidad de la solución una vez que han llegado a ella, lo cual en muchas ocasiones agrega un ingrediente de riesgo que muchas personas no están dispuestas a adicionar en sus negocios.

CAPÍTULO 2

2. ANÁLISIS DE REQUERIMIENTOS, ALCANCES Y DISEÑO DEL PROTOTIPO

2.1. Requerimientos del Prototipo

Nuestro sistema tiene como tarea principal la optimización del uso de índices en un motor de bases de datos mediante el análisis estadístico de operaciones de consulta y utilizando un método heurístico para determinar cuan óptima es la solución encontrada. Los objetos más utilizados por los motores de bases de datos para optimizar el tiempo de respuesta de las peticiones de consultas son los índices, los cuales juegan un papel preponderante en el desempeño del motor de bases de datos, especialmente cuando existen procedimientos almacenados que tienen sus planes de acceso físico enlazados a los índices. Un buen diseño de índices mejora los tiempos de respuesta de las aplicaciones y disminuye los riesgos de que el motor se vaya fuera de línea en horas de alta concurrencia. Mientras que si tenemos índices formados por columnas que no se utilizan frecuentemente en búsquedas como los campos descripción u observaciones o varios índices formados con las mismas columnas (índices redundantes), casos que representan un mal diseño, el motor se degrada, el consumo de recursos aumenta y los tiempos de respuesta se incrementan considerablemente.

No existen normas internacionales que dictaminen que es un buen diseño de índices, sin embargo existen consideraciones que se deben tomar en cuenta:

- crear índices con las columnas usadas en las cláusulas WHERE, GROUP BY y ORDER BY.
- crear al menos un índice no arracimado por tablas (también conocido como índice no único)
- crear los índices al final de la implementación de las aplicaciones que usarán la base de datos

Sin embargo, un buen diseño inicial no basta para mantener al motor funcionando óptimamente, la naturaleza dinámica de todas las empresas implican un cambio constante en los requerimientos de las aplicaciones para adaptarse a las nuevas reglas del negocio ajustando los ciclos de vida de las aplicaciones. Estos cambios implican a su vez cambios en las bases de datos, pero dependiendo de la complejidad del modelo de bases de datos, estos cambios no contemplan la revisión de los índices.

Para los sistemas de bases de datos con grandes cantidades de datos (alrededor de 100.000 registros en promedio por tabla), es una tarea muy difícil monitorear 24 horas al día las operaciones que realizan los

usuarios a través de las aplicaciones; los DBAs muchas veces olvidan monitorear parámetros como el nivel de bloqueos, los tiempos de respuesta de consultas complejas y los recursos que consume el motor para procesarlas, lo cual conlleva efectos no deseados a largo plazo.

Nuestro sistema tiene como objetivo facilitar la tarea de monitoreo de los DBA para determinar el uso de índices mediante la recolección de estadísticas de uso de tablas en la operaciones de consulta y el uso de un método heurístico para determinar cuán óptima es nuestra solución.

2.1.1. Problemas y Limitaciones de los Sistemas Actuales

Los desarrolladores de bases de datos deben negociar entre los puntos fuertes y débiles de las bases de datos para entregar al mercado un producto robusto y escalable, sin embargo muy pocas son las bases de datos especializadas diseñadas para almacenar cierto tipo de datos y optimizar el funcionamiento de aplicaciones dedicadas a resolver un problema muy específico.

La mayoría de los motores de bases de datos comerciales incluyen herramientas de auditoria que permiten monitorean el desempeño del motor, como es el caso del DB2 de IBM con herramienta db2look y db2audit, la cual permite recolectar

estadísticas sobre los espacio de trabajo, columnas e índices de cada una de las tablas de la base de datos. De igual manera Oracle 9i de Oracle, permite recolectar estadísticas sobre el uso de objetos tales como tablas, índices, entre otros; para esto se modifica un parámetro de configuración del motor para iniciar la recolección de estadísticas:

ALTER SYSTEM SET statistics_level=**ALL** [scope=spfile]

El valor de este parámetro puede ser TYPICAL o ALL, para nuestro caso necesitamos ALL para recolectar todas las estadísticas de uso. La opción scope nos permite dejar grabado el parámetro cada vez que reiniciamos el motor, caso contrario tendríamos que alterar el valor del parámetro cuando el motor se ha reiniciado.

Sin embargo, no todo es tan sencillo; estas herramientas de recolección de estadísticas son muy útiles y generan información que puede ser procesada y analizada para rediseñar y modificar algunos aspectos de las bases de datos, pero generan una gran sobrecarga al motor, lo cual

compromete su estabilidad en un ambiente de producción. Basándonos en las características de los diferentes motores comerciales disponibles en el mercado, hemos decidido utilizar Oracle como motor de bases de datos del cual extraeremos las estadísticas debido a que maneja el proceso de recolección de estadísticas mejor que otros motores comerciales, sin agregar mucha sobrecarga de uso de recursos ni aumentar el tiempo de respuesta de las peticiones.

Como punto final, en el mercado nacional no existen aplicaciones conocidas que se enfoquen sólo en el manejo de índices y su impacto en el desempeño de un sistema de bases de datos, razón por la cual nos pareció interesante desarrollar este sistema el cual suple una necesidad básica de todos los DBAs.

2.1.2. Especificación de los Requerimientos del Prototipo

Como ya dijimos nuestro sistema tiene como objetivo facilitar la función de monitoreo del DBA para determinar el uso de índices en una base de datos y así poder sugerir la creación de algunos índices para mejorar los tiempos de respuesta y hacer más

eficiente la ejecución de los planes de optimización definidos por el DBA para afinar el motor; por lo cual el sistema debe cumplir con las siguientes características:

1. *Módulo de Configuración*: en este módulo se incluirán las operaciones de creación, edición y eliminación de las bases de datos a monitorear y tablas. Además aquí se manejan los usuarios que tendrán acceso a nuestro sistema. También en este módulo se manejan procesos de generación de la metadata de las bases de datos monitoreadas hacia nuestro sistema.
2. *Módulo de Análisis de Demanda*: este componente es el principal del sistema e incluye las tareas necesarias para activar el monitoreo de uso de índices sobre una base de datos a la vez. En este módulo se genera el análisis de demanda de las tablas activas para monitoreo de una base de datos y se presentan las sugerencias de índices a crear en base a la recolección de estadísticas. El análisis de demanda debe estar acompañado de las sugerencias de creación de índices por tablas activas, el cual se presentará en base a un día específico. Aquí se aplicarán algoritmos genéticos para generar las combinaciones de columnas de

una tabla para crear el índice que podrían ayudar a mejorar el desempeño del motor.

3. *Módulo de Estadísticas*: Este módulo debe contener las gráficas estadísticas de las demandas de consultas de las tablas activas, lo cual permitirá determinar las fechas en la cuales se ha incrementado la demanda y de esta forma poder con el módulo de análisis de demanda crear el o los índices sugeridos.

Para realizar la selección de índices a crear se debía utilizar un algoritmo heurístico, por lo cual se planteó el uso de: Colonia de Hormigas o Algoritmo Genético.

El algoritmo de Colonia de Hormigas es un método meta-heurístico que utiliza el concepto de hormigas reales para resolver la optimización de un determinado problema, sin embargo para nuestro caso no fue de gran ayuda pues se centra en encontrar la ruta más corta entre dos puntos, lo cual no se ajustaba a nuestro problema de optimizar los tiempos de respuesta mediante el uso de índices. Por esta razón, se descartó su uso.

El algoritmo que se escogió fue el Algoritmo Genético ya que nos permitía solucionar el problema de la creación de índices utilizando los conceptos de demanda basada en estadísticas, lo cual apoya nuestro enfoque de optimización de tiempos de respuestas utilizando índices. A continuación se explicarán las bases teóricas en las que basamos estas afirmaciones.

El algoritmo funciona con una población inicial, la cual está formada por las soluciones conformadas por cromosomas (índices), que a su vez contienen genes (columnas). De esta población inicial, se escoge aquellos cromosomas cuyo grado de aptitud **F** (fitness) sea bajo, y se cruzan entre ellos, es decir un gen de un cromosoma pasa a formar parte de otro cromosoma. Una vez terminado este proceso, los nuevos cromosomas cruzados se incorporan como parte de población inicial, excluyendo a los cromosomas padres. De esta población inicial escogemos a los de mejor rendimiento hasta encontrar una solución óptima.

Los elementos o genes que conforman un cromosoma en nuestro sistema son las columnas de la tabla analizada. Un cromosoma representaría un posible índice de dicha tabla. De

esta manera, nuestro sistema presentará al usuario una serie de soluciones de posibles índices para una tabla y permitirá escoger la que mejor se acople a los objetivos requeridos.

2.2. Alcance del Prototipo

El tema de índices es muy amplio en cuanto a consideraciones y dado que nuestro sistema es un prototipo que ofrecerá las funciones básicas necesarias para monitorear el uso de índices en una base de datos, tendrá los siguientes alcances:

- El sistema permitirá al usuario el ingreso de los siguientes datos para el monitoreo de índices en una base de datos:
    - Base de datos a monitorear (servidor y usuario con credenciales administrativas).
    - Tablas a monitorear de la base de datos, el usuario no las debes ingresar manualmente, sin embargo, debe ejecutar el proceso que ingresa la información a la base de nuestro sistema (metadata).
    - Fechas en las que se realizarán los monitoreos.
    - Usuarios que podrán ejecutar las acciones anteriores.

- Una vez ingresados estos datos el sistema permitirá al usuario escoger la base de datos y las tablas de esa base a monitorear, ver las estadísticas de demanda de las tablas en un rango de fechas escogido por el usuario (rango mayor a una semana) y por día, ver las sugerencias de índices a crear, crear los índices sugeridos, generar reportes de demanda, ver los historiales de replicación de datos de demanda entre la base de producción y la de nuestro sistema.
- Las operaciones sobre la base de producción serán ejecutadas por un agente (aplicación) que hemos creado para simular las consultas que se harían sobre una base de datos por los usuarios de la solución desplegada en el negocio.
- Un agente (aplicación) manejará el proceso de replicación de datos estadísticos obtenidos en producción e ingresados a nuestro sistema cada cierto tiempo, el cual será manejado con una aplicación desarrollada por terceros que planifique tareas, como recomendación sugerimos el uso de VisualCron versión 3.2.2 para Windows.
- Se asume que la base de producción ya existe y está operativa y las aplicaciones que usan la base de datos están desplegadas y funcionando.

- Nuestro sistema estará implementado sobre una plataforma Windows:
    - La capa de base de datos estará en Postgres versión 8.x sobre Windows.
    - La capa de la lógica del negocio estará en JSF (JavaServer Faces) / con Tomcat versión 5.x sobre Windows.
    - Finalmente la capa de presentación puede estar en cualquier plataforma pues al ser una aplicación web, un navegador bastará para usar la herramienta.

2.3. Representación del modelo de optimización

Como se había mencionado anteriormente, un índice puede estar formado por una o más columnas de una tabla, pero no de todas las columnas. Sin embargo, existen columnas en base a las cuales no se podría crear un índice útil por la naturaleza de las mismas, por ejemplo una columna cuyo nombre sea OBSERVACIONES y cuyo tipo de dato sea VARCHAR con longitud 250; sería poco práctico crear un índice con este tipo de columnas debido a que los datos que contendrían no serían muy utilizados en consultas en las aplicaciones y de crear un índice con esta columna, la estructura necesaria para almacenarlo sería compleja y comprometería los recursos del motor.

Por esta razón, como paso previo a la generación de la población inicial de cromosomas, se hace un proceso de selección de aquellas columnas que pueden ser los genes. Este proceso asocia cada columna con un patrón genético, el cual es un número binario. Al transformar dicho patrón genético a decimal, podemos clasificar el grado de aptitud (fitness) de una columna. El grado de aptitud de los cromosomas es la suma del grado de aptitud de todos sus genes.

El patrón genético está formado por ocho dígitos, los cuales pueden ser 0 ó 1. Su valor depende de si cumple (1) o no (0) las reglas que se han definido para medir la aptitud de los genes. La estructura del patrón genético se define a continuación:

Figura 2.1 – Estructura del patrón genético

Dicho patrón cuya representación obedece a un número binario [13] es transformado a un número decimal, el cual es la medida o valor de aptitud utilizado para obtener la aptitud de las soluciones. Así un patrón como 00100111 resulta en el número 39, el cual es el valor de aptitud.

Las reglas que se han definido para generar el patrón genético obedecen a la aplicación de recomendaciones generales en el diseño de bases de datos, tanto a nivel teórico como práctico, durante la investigación previa a la realización de este proyecto. Se han definido 2 clases de reglas: reglas estándar y reglas de conocimiento.

Las reglas estándar obedecen a recomendaciones estándar aplicadas al diseño de índices para bases de datos, entre ellas figuran:

- No utilizar campos que almacenan grandes cantidades de texto como observaciones o comentarios.
- No utilizar los mismos campos repetidamente en los índices de una misma tabla.
- No crear demasiados índices para tablas con grandes volúmenes de datos.
- Las tablas con muy pequeños volúmenes de datos no necesitan índices.
- Seleccionar campos que son frecuentemente parte de SELECTs. Considerar el orden de las columnas, un

índices compuesto por las columnas a,b,c no es lo mismo que un índice formado por las columnas c,b,a.

- No crear índices en base a columnas que reflejen estados o almacenen abreviaturas usadas frecuentemente en toda la tabla, o que contengan campos nulos o que contengan datos repetidos (distribución de datos). Los datos únicos ayudan a mejorar la eficiencia del

Tomando en cuenta las recomendaciones anteriores hemos definido las siguientes reglas estándar, las cuales conforman los dígitos del 1 al 6, las mismas que se detallan a continuación:

Regla 1 – Factor de Nulabilidad: Se determina obteniendo la cantidad de valores nulos contenidos en la columna para la cantidad de registros de la tabla. Ejemplo: una tabla PERSONA, contiene 100.000 registros. Dicha tabla tiene una columna DIRECCION que contiene 30.000 (los otros 70.000 tiene algún valor diferente de nulo) valores nulos.

Nulabilidad = 30.000 / 100.000

Las columnas con un factor de nulabilidad menor o igual al 30% se califican con 1.

Regla 2 – Clave foránea: Aquellas columnas que son claves foráneas (FK) se califican con 1.

Regla 3 – Demanda de clave foránea: Aquellas columnas que son claves foráneas y cuya demanda total (física + lógica) sea mayor a 1500 se califican con 1.

Regla 4 – Clave primaria: Aquellas columnas que no son clave primaria se califican con 1.

Regla 5 – Factor de Selectividad: Se determina obteniendo la cantidad de valores diferentes en una columna para la cantidad de registros de la tabla. Ejemplo: una tabla PERSONA, contiene 100.000 registros. Dicha tabla tiene una columna NOMBRES que contiene 95.670 valores diferentes.

Selectividad = 95.670 / 100.000

Las columnas con un factor de selectividad mayor o igual al 80% se califican con 1.

Regla 6 - El tipo de dato: Las columnas se califican con 1, si y solo si están dentro de la siguiente clasificación:

Tipo de Dato	Longitud Mínima	Longitud Máxima
VARCHAR2	4	30
NUMBER	4	10
DATE	--	--

Tabla 2.1 – Rangos de validez del tamaño del campo por tipo de dato.

Las reglas de conocimiento, por el contrario, están ligadas a la naturaleza del algoritmo utilizado y son muy específicas dependiendo del problema que se quiera enfocar. Éstas permiten cuantificar el conocimiento adquirido por aquellos cromosomas que ya constituyeron soluciones. ¿Pero como conseguir esto? Para poder cuantificar dicho conocimiento utilizaremos la demanda de estas soluciones para integrarla al cálculo de la aptitud.

Además hemos considerado que estas reglas tiene gran relevancia para el cálculo de la aptitud, de manera que constituyen los dígitos más significativos del patrón genético (dígito A y B). A continuación detallamos las reglas:

Regla A – Regularidad: Es el número de veces que el índice (por ende las columnas) ha sido leído lógica o físicamente en un período de 30 días. Esta regla se expresa en forma de porcentaje y basta con que el índice haya sido una sola vez en el día para calificar. Esto es que si el índice se uso al menos 1 vez por 20 días, la regularidad será (20/30)*100 = 66,7%

Regularidad = (# veces al menos 1 vez x día) / 30

Las columnas con una regularidad mayor o igual al 60% se califican con 1.

Regla B – Nivel de regularidad: El nivel de regularidad es el número de días que el índice (por ende las columnas) ha tenido una demanda superior al promedio calculado en base a un período de 30 días. Veamos esto con el siguiente gráfico:

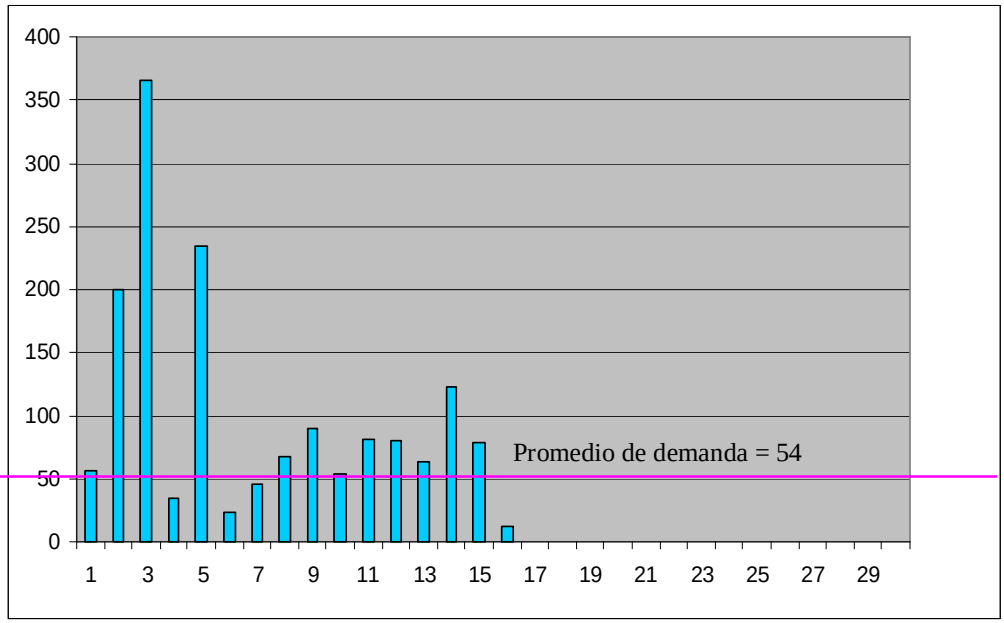


Figura 2.2 – Nivel de regularidad de un índice

La gráfica representa la demanda de un índice en un período de 30 días, como podemos apreciar el promedio de demanda es 54. De esto podemos notar que en los días 4, 6, 7 y 16 a 30 (18 días en total) la demanda diaria ha estado por debajo del promedio mensual, lo cual nos dice que en los restantes 12 días el índice ha estado sobre el promedio mensual, lo cual nos permite determinar el nivel de regularidad del índice:

Nivel de regularidad = (# días demanda excede promedio mensual) / 30

= 12 / 30 = 0.4 = 40%

Las columnas con un nivel de regularidad mayor o igual al 70% se califican con 1.

En base a estas reglas podemos determinar la aptitud de los genes y de los cromosomas que constituirán posibles índices.

De esta manera, tenemos que la aptitud F_i para cada una de nuestras soluciones es:

$$F_n = \sum_{i=1}^{k} F_i$$

donde *n* representa las soluciones (cromosomas) al problema y *k* representa el número de genes (columnas) que contiene el cromosoma$_i$.

La probabilidad de reproducción de cada una de las soluciones se obtiene de la siguiente manera:

$$R = \frac{F_i}{\sum_{j=1}^{n} F_j}$$

donde F_i representa la aptitud del cromosoma$_i$, y *j* representa el universo total de las soluciones, es decir la sumatoria de todas las aptitudes.

Finalmente, podemos representar el problema de optimización, el cual tiene como objetivo mejorar los tiempos de respuesta de la base de datos mediante la utilización de índices, para lo cual analizamos la demanda de las tablas basados en las estadísticas de operaciones de consulta:

$$f_o = \frac{D_T}{D_I}$$

donde D_T representa la demanda total, lógica y física, de la tabla en un día determinado y D_I representa la demanda total, lógica y física, de sus índices.

Pero, ¿cómo estos procesos se ejecutan en el sistema? El primer paso es asegurarse que la tabla sobre la cual vamos a crear índices esté activa (Figura 2.3) para monitoreo:

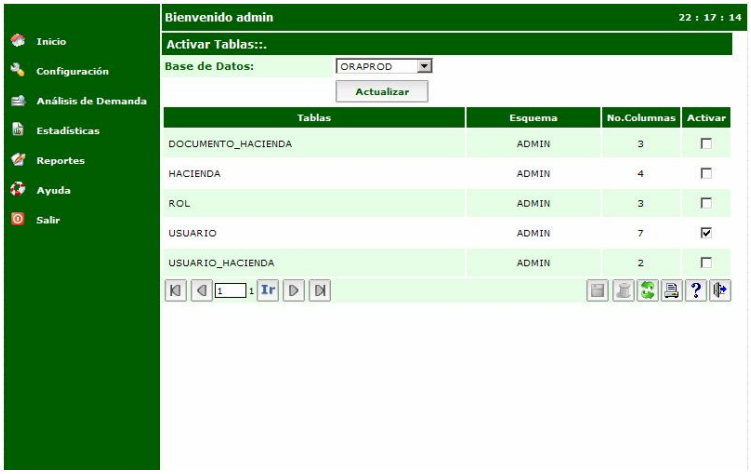


Figura 2.3 – Activar monitoreo de tablas de una base de datos

Utilizaremos la tabla USUARIO como referencia. Existen varias herramientas que nos permiten monitorear la demanda de uso de las tablas, lo cual nos ayuda a determinar si la tabla necesita revisión para posteriormente crear índices:

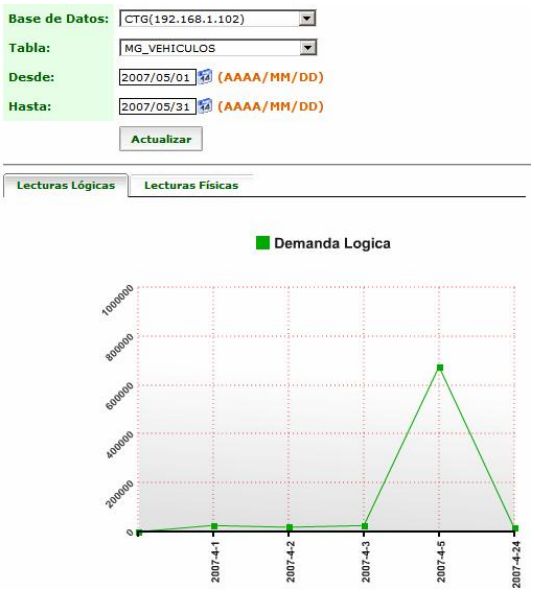


Figura 2.4 – Estadística de demanda de uso de tablas (lecturas lógicas)

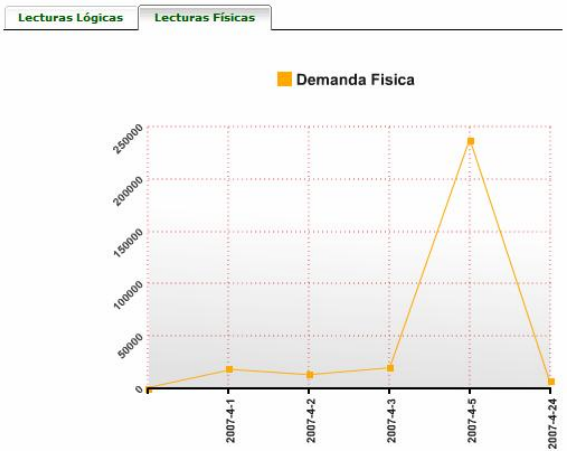


Figura 2.5 – Estadística de demanda de uso de tablas (lecturas físicas)

El sistema cuenta con herramientas que nos permiten determinar el comportamiento de la demanda de uso de una tabla específica. En este

caso, podemos obtener la cantidad de lecturas lógicas (Figura 2.4), con lo cual podemos determinar las fechas (Figura 2.4, encerrado en recuadro rojo) en la cual el motor ha realizado más lecturas a caché en busca de datos almacenados en dicha tabla, lo cual nos servirá para poder utilizar otras herramientas de análisis y determinar a nivel de hora la demanda de uso de la tabla. También podemos obtener la cantidad de lecturas físicas por día (Figura 2.5, encerrado en recuadro rojo), esto nos ayuda a determinar cuando el motor ha realizado lecturas a disco en busca de datos de esa tabla.

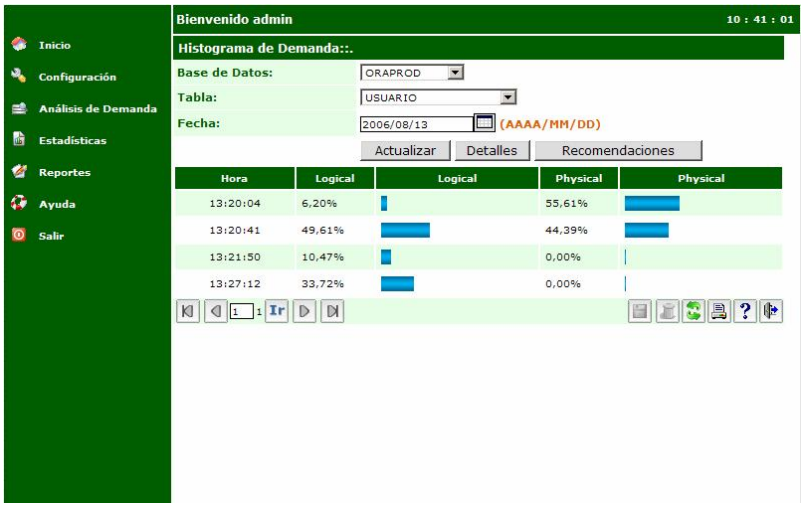


Figura 2.6 – Histograma de demanda diaria de la tabla USUARIO

El DBA debe decidir si la información presentada por el histograma exige el análisis de demanda de la tabla USUARIO. Para el análisis debemos escoger la tabla a analizar y generar el mismo:

Figura 2.7 – Análisis de demanda de la tabla USUARIO

El análisis de demanda de una tabla genera información basada en sus columnas, calificando las mismas según cumplan las reglas definidas para generar el patrón genético y generando su valor de aptitud o fitness.

En la siguiente tabla resumimos el método de calificación de las columnas en base a las reglas definidas en este capítulo:

Nº Regla	Dígito en patrón genético	Medida	Nivel de cumplimiento	Calificación SI cumple	Calificación si NO cumple
1	1 (menos significativo)	Factor de nulabilidad	<= 30%	1	0
2	2	Clave foránea	¿Es clave	1	0

			foránea?		
3	3	Demanda clave foránea	> 1500	1	0
4	4	Clave primaria	¿Es clave primaria?	1	0
5	5	Factor de selectividad	>= 80%	1	0
6	6 (más significativo)	Tipo de datos	Varchar2(4-30) Number(4-10) Date	1	0

Tabla 2.2 – Resumen del método de calificación de columnas para obtener la aptitud

La aplicación de estas reglas en las columnas de la tabla USUARIO genera los siguientes resultados:

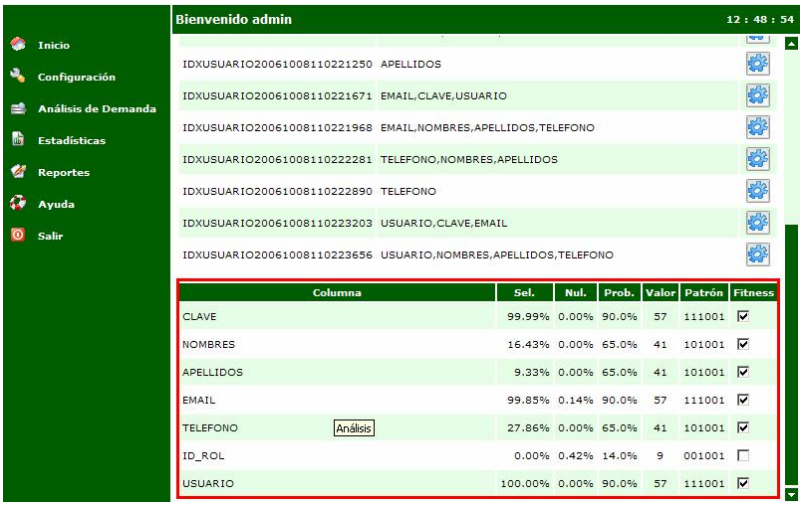


Figura 2.8 – Generación de resultados del análisis de demanda de la tabla USUARIO

Como se puede apreciar en la Figura 2.8, las columnas han sido evaluadas en base a las reglas programadas y los resultados son mostrados. Además, se genera el patrón genético, que a su vez permite obtener la aptitud de las columnas. Este proceso también produce las combinaciones de posibles índices que el DBA puede crear (Figura 2.9, encerrado en recuadro rojo); aquellas combinaciones de índices existentes no pueden volver a crearse (Figura 2.9, encerrado en recuadro azul)

Figura 2.9 – Combinación de posibles índices

En la Figura 2.9, encerrado en recuadro morado, podemos observar un botón que nos provee información sobre las combinaciones cuando éstas han sido usadas anteriormente para crear índices con el sistema. Esta información ayudará al DBA a determinar cuan efectivo fue el índice antes de ser eliminado y decidir si volver a crearlo o no.

Una vez creados los índices, el DBA debe monitorear su demanda y si lo desea puede eliminarlos, pero el sistema almacena las columnas que formaron el índices y la próxima vez que se genere el análisis de demanda, el número de veces que dicha combinación ha sido usada influenciará en el orden de sugerencia de posibles combinaciones de índices a crear y mostrará algunas estadísticas referentes a esas columnas.

La aptitud de la solución puede ser obtenida sumando las aptitudes de sus columnas y es un indicador de cuán óptima es la solución obtenida, sin embargo el monitoreo de las estadísticas es una tarea fundamental que permitirá tomar mejores decisiones en cuanto a los índices que deben ser creados para mejorar los tiempos de respuesta de las aplicaciones, pero este indicador está solo disponible para aquellos índices que fueron creados, eliminados y que nuevamente posan como posibles nuevos índices a crear.

2.4. Especificación de casos de uso y escenarios

CASOS DE USO

<table>
<tr><td colspan="2">Caso de Uso 1: Agregar Usuario</td></tr>
<tr><td colspan="2">Actor: Usuario del Sistema</td></tr>
<tr><td colspan="2">Descripción: Este caso de uso describe el escenario en el cual el Administrador del Sistema ingresa un nuevo usuario a la base de datos</td></tr>
<tr><td colspan="2">Secuencia típica de Eventos:</td></tr>
<tr><td>Actor</td><td>Sistema</td></tr>
<tr><td>El usuario ingresa su nombre de usuario y contraseña.</td><td>El sistema valida las credenciales ingresadas. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.</td></tr>
<tr><td>El usuario ingresa al menú CONFIGURACIÓN.</td><td>El sistema muestra la ventana de CONFIGURACIÓN.</td></tr>
<tr><td>El usuario ingresa al menú SEGURIDAD.</td><td>El sistema muestra la ventana de SEGURIDAD.</td></tr>
<tr><td>El usuario escoge la opción de AGREGAR USUARIO.</td><td>El sistema muestra la pantalla inicial del asistente para ingresar un nuevo usuario.</td></tr>
<tr><td>El usuario hace clic en el botón SIGUIENTE para iniciar el asistente.</td><td>El sistema muestra el formulario con los datos a ingresar del nuevo usuario.</td></tr>
<tr><td>El usuario ingresa los datos del nuevo usuario verificando los campos obligatorios.</td><td>El sistema valida los datos ingresados y una vez acepados, se pasa a una pantalla de revisión.</td></tr>
<tr><td>El usuario revisa los datos y puede enviar el nombre de usuario y contraseña al correo del nuevo usuario.</td><td>El sistema envía el nombre de usuario y contraseña al mail del nuevo usuario si es necesario.</td></tr>
<tr><td>El usuario acepta la revisión.</td><td>El sistema muestra la ventana de SEGURIDAD.</td></tr>
</table>

Tabla 2.3 – Caso de Uso 1: Agregar Usuario

Caso de Uso 2: Editar Usuario	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual el Administrador del sistema modifica los datos de un usuario existente.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y contraseña.	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario ingresa al menú CONFIGURACIÓN.	El sistema muestra la ventana de CONFIGURACIÓN.
El usuario ingresa al menú SEGURIDAD.	El sistema muestra la ventana de SEGURIDAD.
El usuario escoge la opción de EDITAR USUARIO.	El sistema muestra una lista paginada de los usuarios existentes en el sistema y su información.
El usuario escoge la opción EDITAR (ícono EDITAR) o puede hacer clic sobre el nombre de usuario del usuario a editar para modificar sus datos.	El sistema muestra una pantalla con un formulario editable con la información del usuario escogido.
El usuario modifica la información que necesita y hace clic en GUARDAR.	El sistema verifica que los datos modificados sean válidos y guardas los datos modificados en la base de datos y regresa a la pantalla mostrando todos los usuarios.

Tabla 2.4 – Caso de Uso 2: Editar Usuario

Caso de Uso 3: Eliminar Usuario	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual el Administrador del sistema elimina los datos de un usuario existente.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y contraseña.	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario ingresa al menú CONFIGURACIÓN.	El sistema muestra la ventana de CONFIGURACIÓN.
El usuario ingresa al menú SEGURIDAD.	El sistema muestra la ventana de SEGURIDAD.
El usuario escoge la opción de EDITAR USUARIO.	El sistema muestra una lista paginada de los usuarios existentes en el sistema y su información.
El usuario marca la opción ELIMINAR (casilla ELIMINAR) del usuario que va a eliminar.	El sistema activa el icono de Eliminar de la barra de herramientas (parte inferior).

Tabla 2.5 – Caso de Uso 3: Eliminar Usuario

Caso de Uso 4: Agregar Base de Datos	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual un Administrador agrega una nueva base de datos a ser monitoreada por el sistema.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de cuenta y clave	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir
El usuario escoge la opción de Configuración.	El sistema muestra en pantalla la configuración del sistema.
El usuario ingresa al menú de Sistema.	El sistema muestra en pantalla las opciones disponibles dentro de sistema.
El usuario escoge la opción de Agregar Base de Datos.	El sistema muestra la pantalla inicial del asistente para ingresar una nueva base de datos.
El usuario da clic en el botón de siguiente.	El sistema muestra en pantalla el formulario donde se deben ingresar los datos de la nueva base de datos.
El usuario da clic en el botón de siguiente.	El sistema muestra una pantalla donde se puede verificar el estado de la conexión a la base de datos.
El usuario da clic en finalizar.	El sistema valida que los datos sean correctos y se proceden a almacenar estos datos al sistema.

Tabla 2.6 – Caso de Uso 4: Agregar Base de Datos.

Caso de Uso 5: Editar Base de Datos	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual un Administrador va a editar una base de datos ingresada al sistema.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir
El usuario escoge la opción de configuración.	El sistema muestra la pantalla de configuración del sistema.
El usuario ingresa al menú Sistema.	El sistema muestra en pantalla las opciones del sistema.
El usuario escoge la opción de Editar una base de datos.	El sistema muestra en pantalla todas las bases de datos ingresadas al sistema.
El usuario selecciona la base de datos que desea modificar.	El sistema muestra un formulario con los datos de la base de datos seleccionada.
El usuario guarda los cambios.	El sistema valida que los datos cambiados sean válidos y luego se procede a actualizar los datos.

Tabla 2.7 – Caso de Uso 5: Editar Base de Datos.

Caso de Uso 6: Poblar datos de una base de datos al sistema	
Actor: Usuario del Sistema	
Descripción: En este caso de uso se describe el escenario en el cual un Administrador va a realizar la respectiva población de datos de la estructura de una base de datos.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir
El usuario escoge la opción de Configuración.	El sistema muestra la pantalla de configuración del sistema.
El usuario ingresa al menú Sistema.	El sistema muestra en pantalla las opciones del sistema.
El usuario escoge la opción de Poblar datos de una base de datos al sistema.	El sistema muestra en pantalla una lista de las bases de datos ingresadas.
El usuario escoge una base de datos y procede a dar clic en el botón de poblar.	El sistema empieza a ejecutar el proceso de poblar los datos y al finalizar el proceso muestra un resumen de cualquier novedad presentada en el proceso.

Tabla 2.8 – Caso de Uso 6: Poblar datos de una base de datos al sistema.

Caso de Uso 7: Eliminar datos de una base de datos del sistema	
Actor: Usuario del Sistema	
Descripción: En este caso de uso se describe el escenario en el que el Administrador procede a eliminar los datos de la estructura de una base de datos.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	El sistema valida las credenciales. Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir
El usuario escoge la opción de Configuración.	El sistema muestra la pantalla de configuración del sistema.
El usuario ingresa al menú Sistema.	El sistema muestra en pantalla las opciones del sistema.
El usuario escoge la opción de Eliminar datos de una base de datos al sistema.	El sistema muestra en pantalla una lista de las bases de datos ingresadas.
El usuario escoge una base de datos y procede a dar clic en el botón de eliminar datos.	El sistema empieza a ejecutar el proceso de poblar los datos y al finalizar el proceso muestra un resumen de cualquier novedad presentada en el proceso.

Tabla 2.9 – Caso de Uso 7: Eliminar datos de una base de datos del sistema.

Caso de Uso 8: Activar Monitoreo de Bases de Datos	
Actor: Usuario del Sistema	
Descripción: En este caso de uso se describe el escenario en el cual el usuario activa el monitoreo de una base de datos.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla las opciones de Análisis de Demanda.
El usuario escoge la opción de Activar Monitoreo de Base de Datos.	El sistema muestra en pantalla la pagina donde se puede seleccionar la base de datos que desea activar.
El usuario escoge la base de datos que desea activar.	El sistema inmediatamente activa la base seleccionada y desactiva la que estaba activa anteriormente.
El usuario guarda los cambios realizados para que se hagan efectivos.	El sistema realiza los cambios internamente.

Tabla 2.10 – Caso de Uso 8: Activar Monitoreo de Bases de Datos.

Caso de Uso 9: Activar Monitoreo de Tablas	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual el usuario del sistema activa el monitoreo de una o mas tablas.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla las opciones de Análisis de Demanda.
El usuario escoge la opción de Activar Monitoreo de Tablas.	El sistema muestra en pantalla la página donde se puede seleccionar la base de datos activa.
El usuario selecciona la base de datos activa y presiona actualizar.	El sistema muestra una lista de todas tablas registradas en el sistema de la base de datos activa.
El usuario activa las tablas que desea habilitar su monitoreo.	El sistema le cambia el estado de monitoreo a la(s) tabla(s) seleccionada(s).
El usuario guarda los cambios realizados.	El sistema realiza las modificaciones solicitadas por el usuario.

Tabla 2.11 – Caso de Uso 9: Activar Monitoreo de Tablas.

Caso de Uso 10: Demanda por Base de Datos	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual el usuario del sistema va a revisar las demandas que existen por bases de datos.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla las opciones de Análisis de Demanda.
El usuario escoge la base de datos activa y da clic en actualizar.	El sistema muestra en pantalla una lista de las demandas sumarizadas obtenidas por cada tabla monitoreada.

Tabla 2.12 – Caso de Uso 10: Demanda por Base de Datos.

Caso de Uso 11: Demanda por Tabla	
Actor: Usuario del Sistema	
Descripción: Este caso de uso detalla el escenario donde el usuario del sistema revisa las demandas que ha tenido cada tabla en un día específico.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla las opciones de Análisis de Demanda.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla la página donde se puede seleccionar la base de datos activa para revisar la demanda que ha tenido cada tabla.
El usuario escoge la opción de Demanda por Tablas.	El sistema muestra en pantalla la página donde se puede seleccionar la base de datos activa para revisar la demanda que ha tenido cada tabla.
El usuario selecciona la base de datos que se encuentra activa.	El sistema muestra una lista desplegable de las tablas de la base de datos activa.
El usuario selecciona una tabla de la base de datos activa y selecciona la fecha y da clic en actualizar.	El sistema muestra una lista con las lecturas de todo el día.

Tabla 2.13 – Caso de Uso 11: Demanda por Tabla.

Caso de Uso 12: Ver Log de Replicación de Datos	
Actor: Usuario del Sistema	
Descripción: Este caso de uso detalla el escenario donde el usuario del sistema revisa las demandas que ha tenido cada tabla en un día específico.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Análisis de Demanda del panel izquierdo.	El sistema muestra en pantalla las opciones de Análisis de Demanda.
El usuario escoge la opción de Historial&Logs.	El sistema muestra en pantalla una página donde se muestra el log por fecha.
El usuario escoge una fecha para mostrar el log de ese día únicamente y presiona en Actualizar.	El sistema muestra en pantalla el log generado por el agente replicador del día seleccionado por el usuario.

Tabla 2.14 – Caso de Uso 12: Ver Log de Replicación de Datos.

Caso de Uso 13: Estadísticas por Base de Datos	
Actor: Usuario del Sistema	
Descripción: Este caso de uso describe el escenario en el cual el usuario del sistema consulta las estadísticas de la base de datos a manera de gráficas estadísticas.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Estadísticas del panel izquierdo.	El sistema muestra en pantalla las opciones de Estadísticas.
El usuario escoge la opción de Estadísticas por Base de Datos.	El sistema muestra en pantalla una página donde el usuario debe seleccionar la base de datos activa.
El usuario selecciona la base de datos y un rango de fechas.	El sistema muestra dos gráficas, una de lecturas lógicas y otra de lecturas físicas del rango de fechas seleccionado.

Tabla 2.15 – Caso de Uso 13: Ver Estadísticas por Base de Datos.

Caso de Uso 14: Estadísticas por Tablas

Actor: Usuario del Sistema

Descripción: Este caso de uso detalla el escenario en el cual el usuario del sistema

Secuencia típica de Eventos:

Actor	Sistema
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Estadísticas del panel izquierdo.	El sistema muestra en pantalla las opciones de Estadísticas.
El usuario escoge la opción de Estadísticas por Tablas.	El sistema muestra en pantalla una página donde el usuario debe seleccionar la base de datos activa.
El usuario selecciona la base de datos, la tabla y el rango de fechas que desea consultar.	El sistema muestra dos gráficas, una de lecturas lógicas y otra de lecturas físicas del rango de fechas seleccionado.

Tabla 2.16 – Caso de Uso 14: Estadísticas por Tablas.

Caso de Uso 15: Estadísticas por Índices	
Actor: Usuario del Sistema	
Descripción: Este caso de uso detalla el escenario en el cual el usuario revisa las estadísticas de las lecturas lógicas y físicas del índice seleccionado.	
Secuencia típica de Eventos:	
Actor	**Sistema**
El usuario ingresa su nombre de usuario y su contraseña.	Las opciones no permitidas no se desactivan, sin embargo las páginas no autorizadas muestran un mensaje notificando la falta de privilegios en caso de existir.
El usuario escoge la opción de Estadísticas del panel izquierdo.	El sistema muestra en pantalla las opciones de Estadísticas.
El usuario escoge la opción de Estadísticas por Índices.	El sistema muestra en pantalla una página donde el usuario debe seleccionar la base de datos activa.
El usuario selecciona la base de datos, el índice y el rango de fechas que desea consultar.	El sistema muestra dos gráficas, una de lecturas lógicas y otra de lecturas físicas del rango de fechas seleccionado.

Tabla 2.17 – Caso de Uso 15: Estadísticas por Índices.

ESCENARIOS

Caso de Uso 1: Agregar Usuario

Escenario 1.1:

El usuario ingresa un nuevo usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de AGREGAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.
- El usuario ingresa correctamente todos los datos requeridos.
- El nuevo nombre de usuario del nuevo usuario no existe en el sistema.

Salidas:

- El sistema ingresa el nuevo usuario.
- El sistema retorna a la pantalla de Seguridad.

Escenario 1.2:

El usuario no ingresa un nuevo usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de AGREGAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.

- El usuario no ingresa correctamente todos los datos requeridos.
- El nuevo nombre de usuario del nuevo usuario ya existe en el sistema.

Salidas:

- El sistema no ingresa el nuevo usuario.
- El sistema muestra un mensaje de error.

Caso de Uso 2: Editar Usuario

Escenario 2.1:

El usuario modifica la información de un usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de EDITAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.
- El usuario escoge el usuario que desea modificar.
- El usuario ingresa correctamente todos los datos requeridos.

Salidas:

- El sistema guarda los datos modificados del usuario.
- El sistema retorna a la pantalla de usuarios del sistema.

Escenario 2.2:

El usuario no modifica la información de un usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de EDITAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.
- El usuario escoge el usuario que desea modificar.
- El usuario no ingresa correctamente todos los datos requeridos.

Salidas:

- El sistema no guarda los datos modificados del usuario.
- El sistema muestra un mensaje de error.

Caso de Uso 3: Eliminar Usuario

Escenario 3.1:

El usuario elimina un usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de EDITAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.
- El usuario escoge el usuario que desea eliminar.
- El usuario escoge la opción ELIMINAR de la barra de herramientas inferior para confirmar la eliminación.

Salidas:

- El sistema elimina el usuario.
- El sistema refresca la pantalla de usuarios del sistema.

Escenario 3.2:

El usuario no elimina un usuario exitosamente.

Supuestos:

- El usuario decide escoger la opción de EDITAR USUARIOS dentro del menú CONFIGURACIÓN del menú principal.
- El usuario no escoge el usuario que desea eliminar.
- El usuario no escoge la opción ELIMINAR de la barra de herramientas inferior para confirmar la eliminación.

Salidas:

- El sistema no elimina el usuario.

Caso de Uso 4: Agregar Base de Datos

Escenario 4.1:

El usuario ingresa una nueva base de datos al sistema exitosamente.

Supuestos:

- El usuario decide que base de datos va a ingresar
- El usuario llena perfectamente los campos necesarios para el ingreso de la nueva base de datos.
- El nombre de la base de datos no existe dentro del sistema.

Salidas:

- El sistema agrega la base de datos al sistema.
- El sistema muestra una pantalla donde confirma que la base de datos ha sido agregada con éxito.

Escenario 4.2:

El usuario no puede ingresar una nueva base de datos al sistema.

Supuestos:

- El usuario decide que base de datos va a ingresar
- El usuario no llena perfectamente los campos necesarios para el ingreso de nuevo proveedor.
- El nuevo proveedor no existe dentro del sistema.

Salidas:

- El sistema no puede agregar la base de datos al sistema.
- El sistema muestra un mensaje del error.

Caso de Uso 5: Editar Base de Datos

Escenario 5.1:

El usuario modifica una base de datos del sistema de manera exitosa.

Supuestos:

- El usuario decide modificar datos de una base ya ingresada.
- El usuario cambia perfectamente los campos deseados.

Salidas:

- El sistema guarda los datos modificados.
- El sistema muestra la pantalla de las opciones del sistema.

Escenario 5.2:

El usuario no puede modificar una base de datos del sistema.

Supuestos:

- El usuario decide que base de datos va a modificar.
- El usuario no llena perfectamente los campos que desea modificar.

Salidas:

- El sistema no puede modificar la base de datos del sistema.
- El sistema muestra en pantalla el error generado.

Caso de Uso 6: Poblar datos de una base de datos al sistema

Escenario 6.1:

El usuario decide poblar los datos necesarios de una base de datos ya ingresada al sistema.

Supuestos:

- El usuario decide poblar los datos de una base de datos ingresada al sistema.
- El canal de comunicación para conectarse a la base de datos se encuentra disponible.
- La instancia de la base de datos de Oracle se encuentra habilitada.
- El usuario asociado con la base de datos a poblar tiene privilegios de DBA.

Salidas:

- El sistema realiza un scan para obtener los datos de la estructura de la base de datos seleccionada.
- El sistema muestra en pantalla que se realizó correctamente la generación de datos.

Escenario 6.2:

El usuario no pudo generar correctamente los datos de la base de datos.

Supuestos:

- El usuario decide poblar los datos de una base de datos ingresada al sistema.
- El canal de comunicaciones a la instancia de la base en Oracle no está disponible.
- La base de datos en Oracle no se encuentra habilitada.

Salidas:

- El sistema no puede modificar la base de datos del sistema.
- El sistema muestra en pantalla el error generado.

Caso de Uso 7: Eliminar datos de una base de datos al sistema

Escenario 7.1:

El usuario decide eliminar los datos necesarios de una base de datos ya ingresada al sistema.

Supuestos:

- El usuario decide eliminar los datos de una base de datos ingresada al sistema.

- El canal de comunicación para conectarse a la base de datos se encuentra disponible.
- La instancia de la base de datos de Oracle se encuentra habilitada.
- El usuario asociado con la base de datos a poblar tiene privilegios de DBA.

Salidas:

- El sistema realiza un scan para obtener los datos de la estructura de la base de datos seleccionada.
- El sistema elimina todos los datos de la base de datos seleccionada del sistema.

Escenario 7.2:

El usuario no pudo eliminar correctamente los datos de la base de datos.

Supuestos:

- El usuario decide poblar los datos de una base de datos ingresada al sistema.
- El canal de comunicaciones a la instancia de la base en Oracle no se encuentra disponible.
- La base de datos en Oracle no se encuentra habilitada.

Salidas:

- El sistema no puede modificar la base de datos del sistema.
- El sistema muestra en pantalla el error generado.

Caso de Uso 8: Activar Monitoreo de Bases de Datos

Escenario 8.1:

El administrador decide eliminar los datos necesarios de una base de datos ya ingresada al sistema.

Supuestos:

- El administrador decide activar el monitoreo de una base de datos.
- El canal de comunicación para conectarse a la base de datos se encuentra disponible.
- La instancia de la base de datos de oracle se encuentra habilitada.
- El usuario asociado con la base de datos a poblar tiene privilegios de DBA.

Salidas:

- El sistema realiza una búsqueda en las tablas de catálogo para obtener los datos de la estructura de la base de datos seleccionada.

- El sistema elimina todos los datos de la base de datos seleccionada del sistema.

Escenario 8.2:

El administrador no pudo eliminar correctamente los datos de la base de datos.

Supuestos:

- El administrador decide poblar los datos de una base de datos ingresada al sistema.
- El canal de comunicaciones a la instancia de la base en Oracle no se encuentra disponible.
- La base de datos en Oracle no se encuentra habilitada.

Salidas:

- El sistema no puede modificar la base de datos del sistema.
- El sistema muestra en pantalla el error generado.

Caso de Uso 9: Activar Monitoreo de Tablas

Escenario 9.1:

El usuario del sistema decide activar el monitoreo de una o mas tablas de una base de datos.

Supuestos:

- El administrador decide activar el monitoreo de una o mas tablas.
- Las tablas que desea activar el usuario del sistema deben estar generadas en la aplicación.

Salidas:

- El sistema almacena todos los cambios.

Caso de Uso 10: Demanda por Base de Datos

Escenario 10.1:

El usuario del sistema decide activar el monitoreo de una Base de Datos.

Supuestos:

- El administrador decide revisar la demanda global de todas las tablas activas de una base de datos.

Salidas:

- El sistema muestra en pantalla de manera gráfica los porcentajes de demanda de cada tabla de la base de datos seleccionada.

Caso de Uso 11: Demanda por Tabla

Escenario 11.1:

El usuario del sistema decide activar el monitoreo de una Base de Datos.

Supuestos:

- El administrador decide revisar la demanda que existe en cada tabla tablas activa de una base de datos.

Salidas:

- El sistema muestra en pantalla de manera gráfica los porcentajes de demanda de la tabla seleccionada.

Caso de Uso 12: Ver Log de Replicación de Datos

Escenario 12.1:

- El usuario del sistema decide activar el monitoreo de una Base de Datos.

Supuestos:

- El administrador decide revisar el log generado por el agente replicador.

Salidas:

- El sistema muestra en pantalla los datos almacenados en el log del agente replicador.

Caso de Uso 13: Estadísticas por Base de Datos

Escenario 13.1:

El usuario del sistema decide revisar las estadísticas de una base de datos.

Supuestos:

- El usuario del sistema decide revisar las estadísticas de una base de datos activa.
- El usuario selecciona una tabla activa de la base de datos seleccionada.

Salidas:

- El sistema muestra en pantalla las gráficas de lecturas lógicas y físicas de la base de datos seleccionada.

Caso de Uso 14: Estadísticas por Tablas

Escenario 14.1:

El usuario del sistema decide revisar las estadísticas de una tabla activa.

Supuestos:

- El usuario del sistema decide revisar las estadísticas de una base de datos activa.
- El usuario selecciona una tabla activa de la base de datos seleccionada.

Salidas:

- El sistema muestra en pantalla las gráficas de lecturas lógicas y físicas de la tabla seleccionada.

Caso de Uso 15: Estadísticas por Índices

Escenario 15.1:

El usuario del sistema decide revisar las estadísticas de un índice.

Supuestos:

- El usuario del sistema decide revisar las estadísticas de una base de datos activa.
- El usuario selecciona un índice de una tabla activa de la base de datos seleccionada.

Salidas:

- El sistema muestra en pantalla las gráficas de lecturas lógicas y físicas de la tabla seleccionada.

2.5. Flujo de Ventanas y Layouts

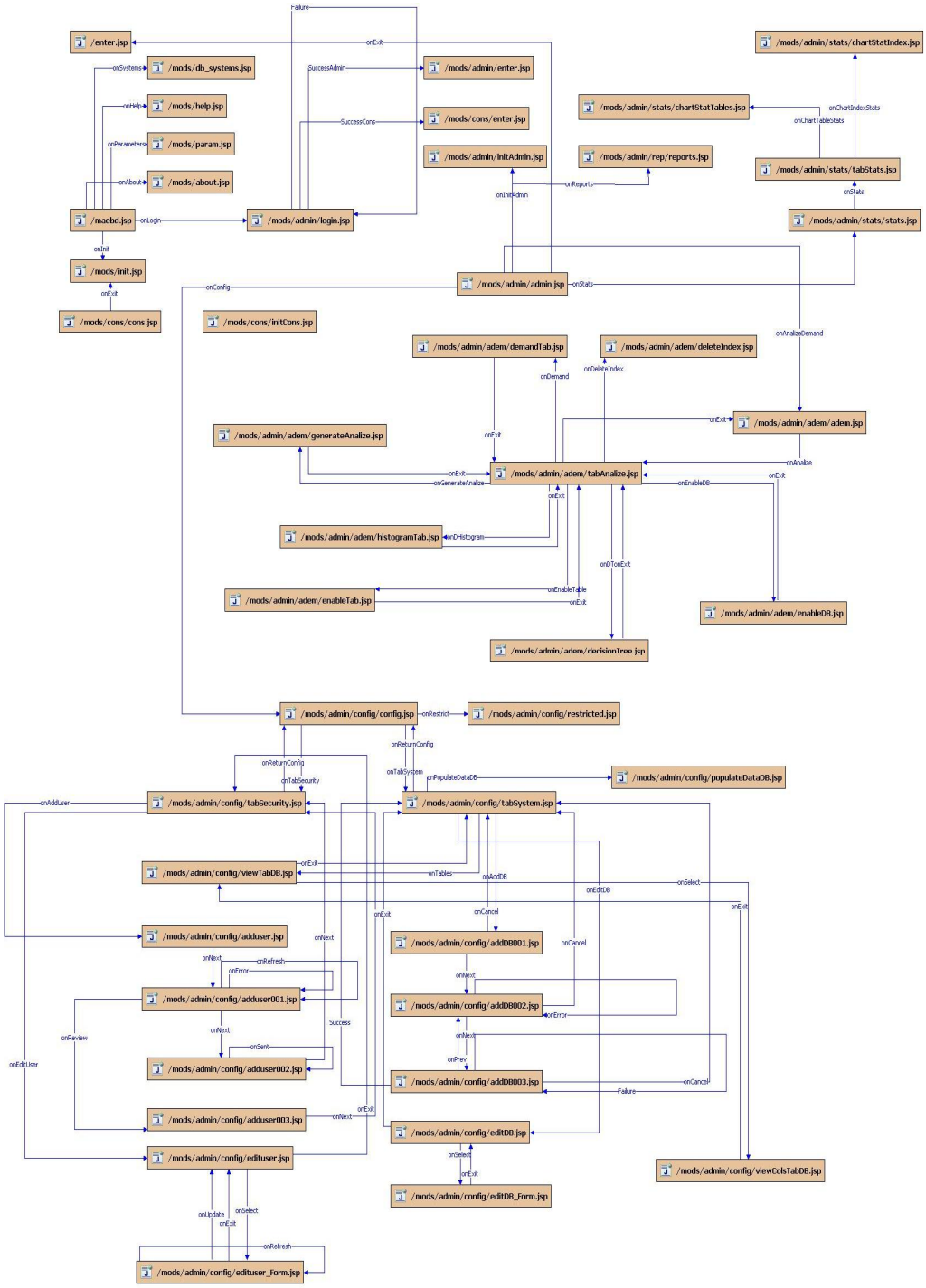


Figura 2.10 - Diagrama del flujo de ventanas de la aplicación

2.6. Diagrama de Objetos

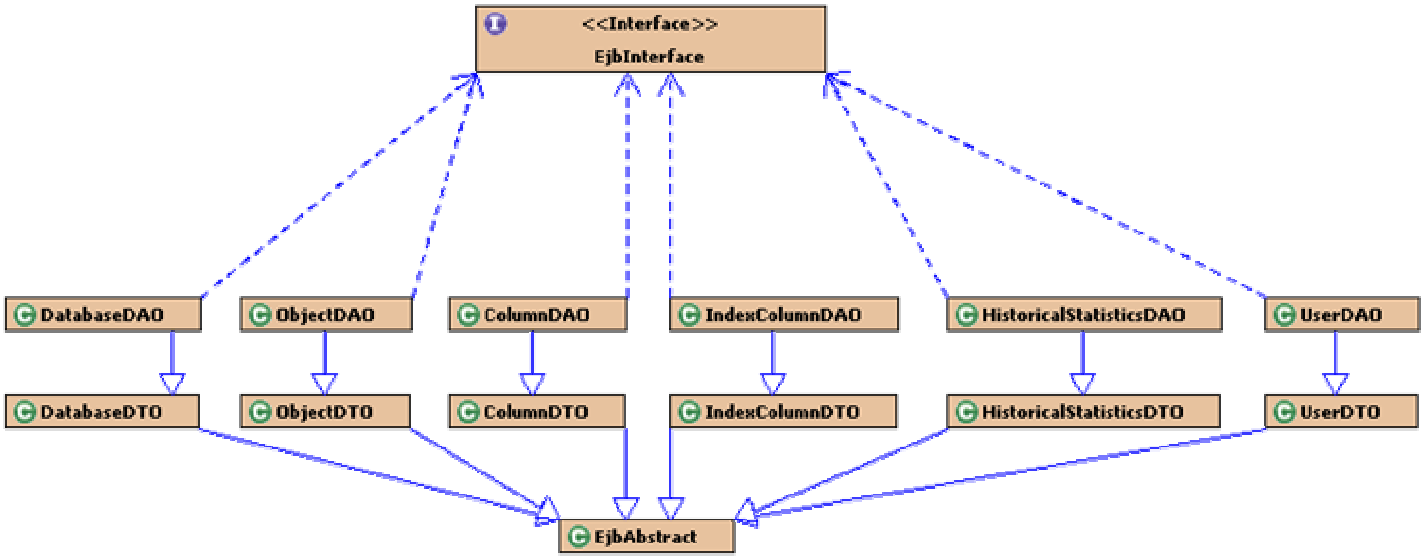


Figura 2.11 - Diagrama de objetos del paquete com.maebd.db

Figura 2.12 - Diagrama de objetos del paquete com.maebd.app.opt.obj

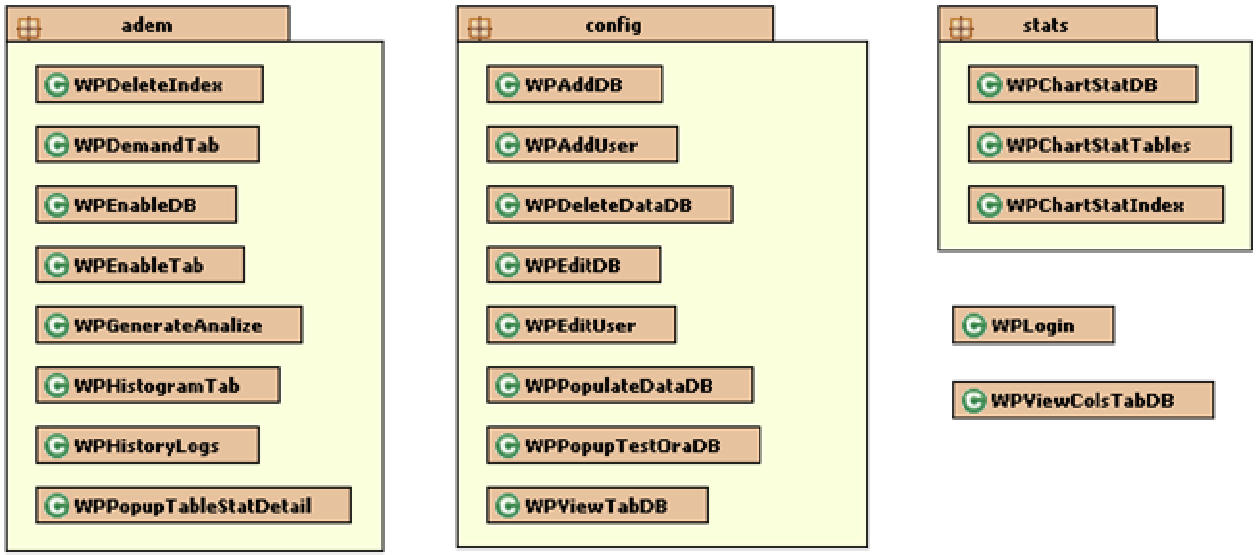


Figura 2.13 - Diagrama de objetos del paquete com.maebd.web.admin

2.7. Diseño del Prototipo

2.7.1. Diseño Arquitectónico

Para la implementación de nuestro sistema hemos utilizado el esquema de 3 capas: Base de Datos (Capa 1), Lógica del Negocio u Objetos (Capa 2) y Presentación o Interfaz de Usuario (Capa 3).

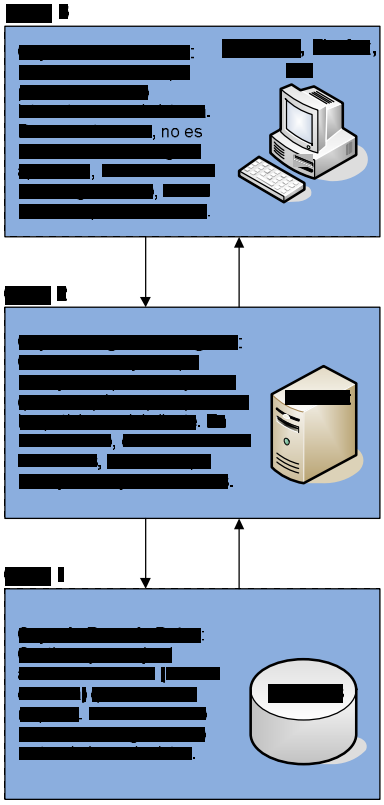


Figura 2.14 – Modelo arquitectónico del sistema

Para facilitar la implementación y mejorar el factor escalabilidad se decidió utilizar el enfoque de 3 capas para desarrollar nuestro sistema, tal como se muestra en la figura 2.14.

La capa 1 o de base de datos, maneja el acceso a los datos atendiendo las peticiones de lectura y escritura del cliente (capa 2 y capa 3). El sistema utilizará Postgres como base de datos.

Mientras que la capa 2 o de lógica del negocio, contiene todos los objetos que manejan la lógica de la aplicación, procesos y datos que se requieren para procesar las peticiones de los clientes (capa 3). El sistema utilizará un contenedor de servlets, Tomcat para almacenar estos objetos.

Finalmente, la capa 3 o de presentación es la interfaz de usuario, con la cual éste interactuará con el sistema, que en el caso del nuestro será cualquier navegador web (Internet Explorer, Firefox, Opera, etc.)

Así mismo nuestro aplicación es un sistema cliente/servidor, en el cual las capas 1 y 2 actúan como servidor y la capa 3 actúa como cliente. La capa 2 también puede actuar como cliente cuando hace peticiones a la base de datos o capa 1, sin embargo globalmente veremos al sistema dividido de la forma indicada en primera instancia y tal como lo detalla la figura 2.15:

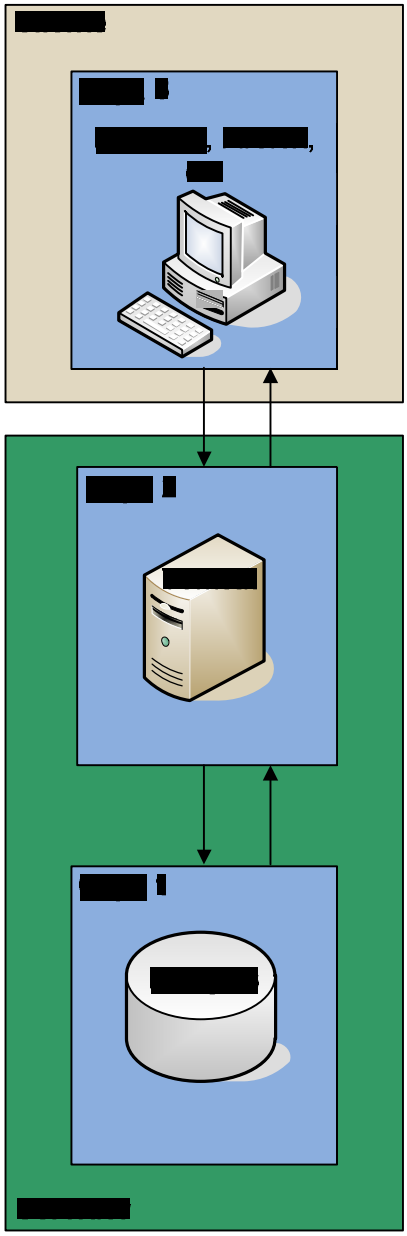


Figura 2.15 – Esquema cliente/servidor

2.7.2. Diseño de interacción con el usuario

En esta sección describiremos ciertos aspectos importantes que se tomaron en cuenta al momento de hacer el diseño de la aplicación.

El diseño del sistema MAEBD (Monitor y Analizador Estadístico de Bases de Datos) fue pensado para ser sencillo y fácil para los usuarios. Hay que resaltar que en muchas de las opciones están enfocadas a ser usadas por DBAs o personas con conocimientos de sistemas.

Figura 2.16 - Esquema estructural de la aplicación

En la Figura 2.16 se muestra un esquema general de cómo se encuentra dividido el diseño de la aplicación. Del lado izquierdo tenemos un panel donde se muestran todas las opciones que puede escoger el usuario sin necesidad de autentificación. En el panel central se encuentra un frame en el cual se mostrarán las páginas de las opciones que vaya seleccionando el usuario.

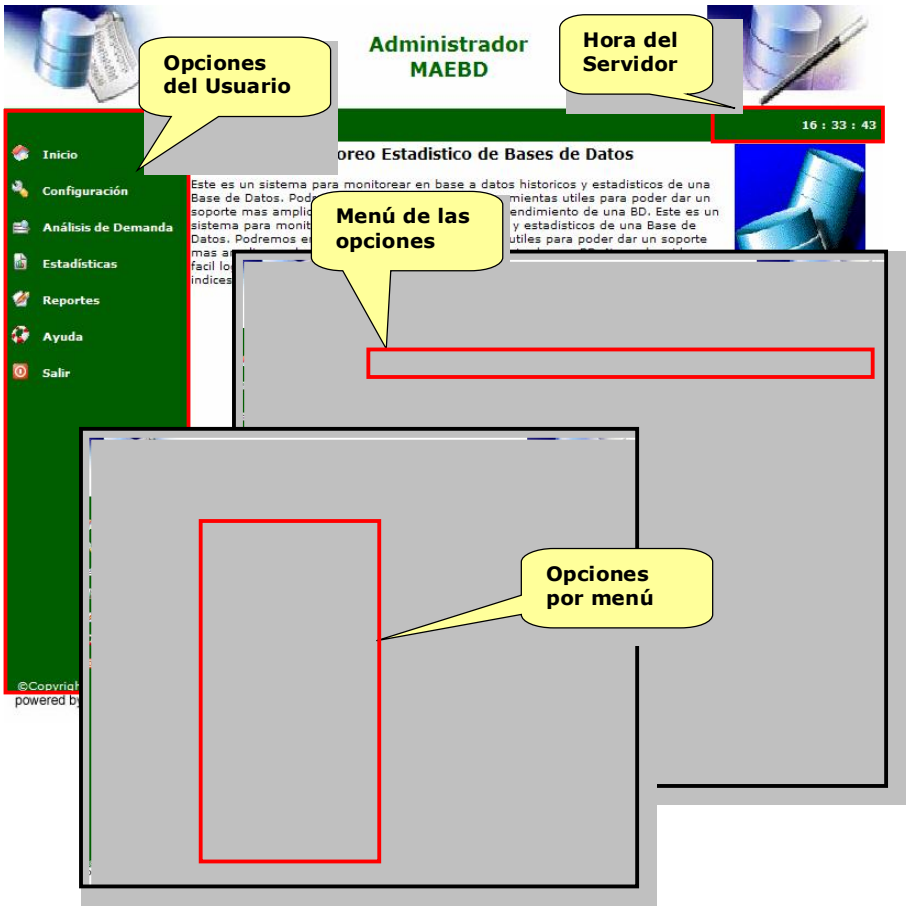
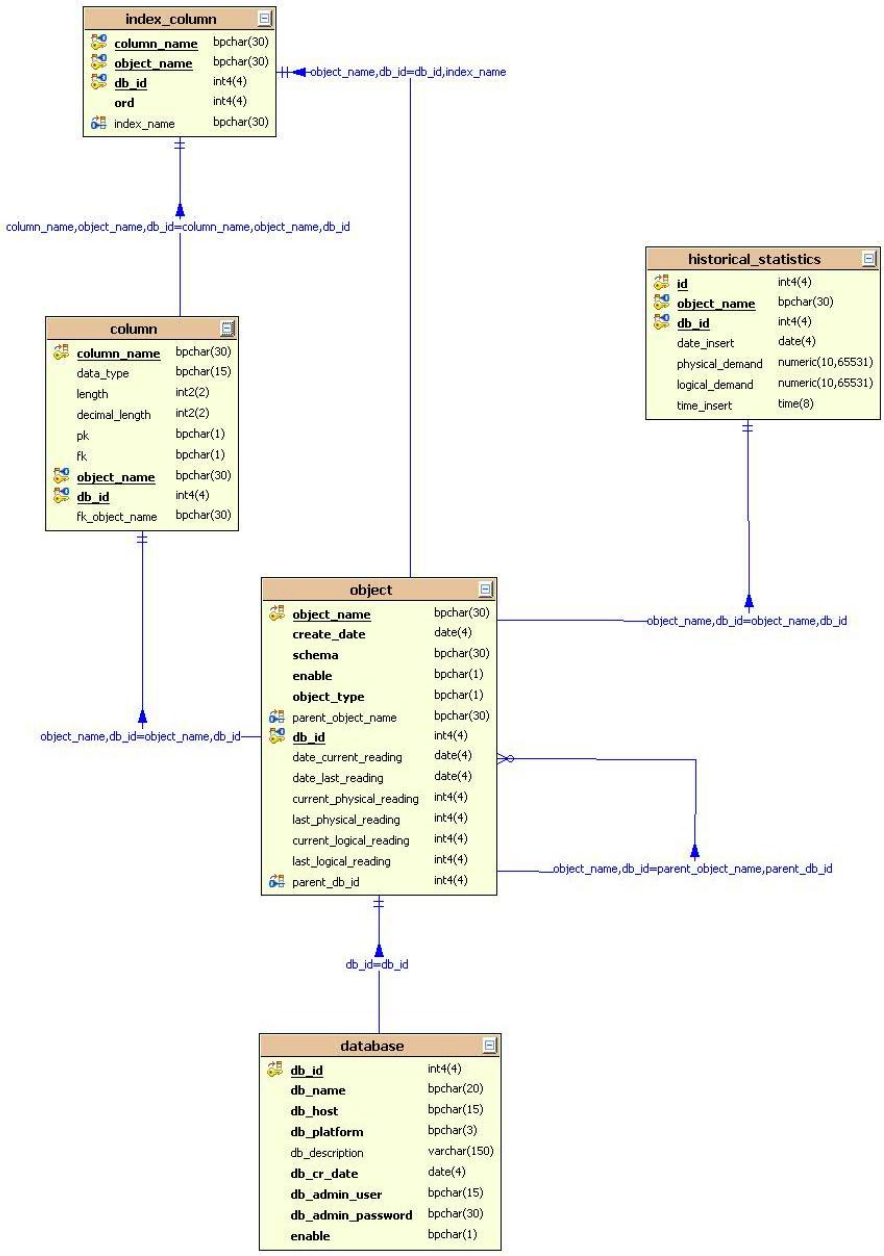


Figura 2.17 - Esquema estructural del administrador

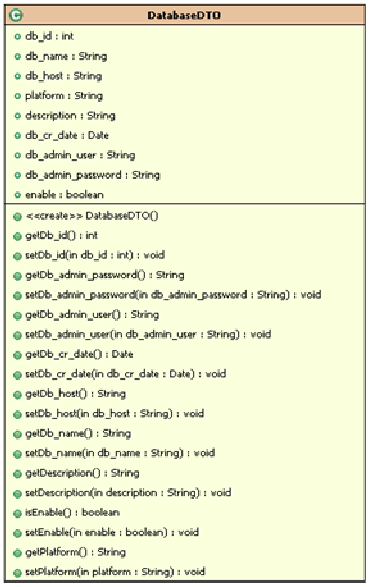
En la figura 2.17 se muestra el esquema general del diseño que tiene el usuario luego de ser autentificado por el sistema. Como se puede apreciar se mantiene un diseño similar al de la figura anterior, con la pequeña diferencia que varía el color.

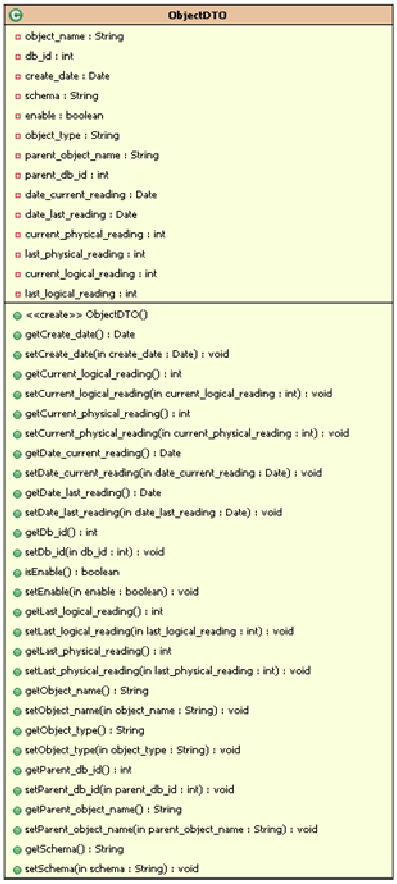
Los errores que se produzcan en el sistema serán fácilmente apreciados en color rojo. Éste diseño fue pensado para que el sistema sea fácil de usar e intuitivo.

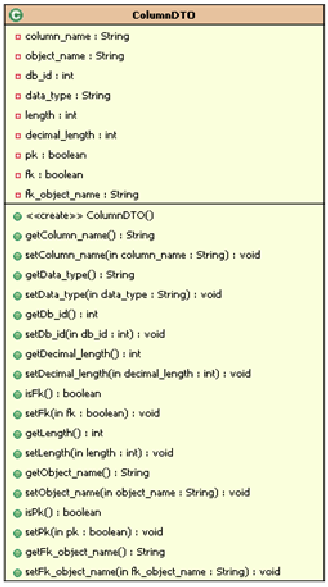
2.7.3. Diseño de la Base de Datos

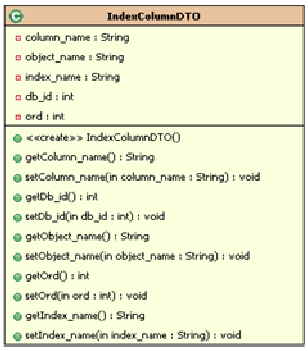


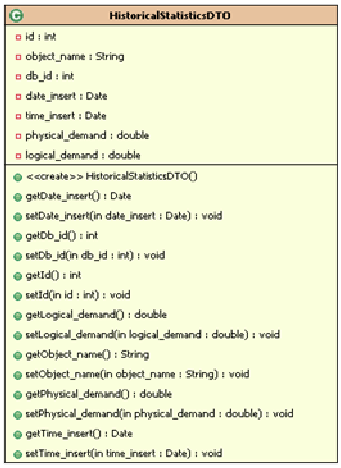
2.7.4. Diseño de Clases

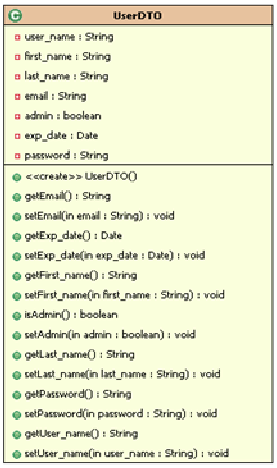

DatabaseDTO
db_id : int
db_name : String
db_host : String
platform : String
description : String
db_cr_date : Date
db_admin_user : String
db_admin_password : String
enable : boolean
<<create>> DatabaseDTO()
getDb_id() : int
setDb_id(in db_id : int) : void
getDb_admin_password() : String
setDb_admin_password(in db_admin_password : String) : void
getDb_admin_user() : String
setDb_admin_user(in db_admin_user : String) : void
getDb_cr_date() : Date
setDb_cr_date(in db_cr_date : Date) : void
getDb_host() : String
setDb_host(in db_host : String) : void
getDb_name() : String
setDb_name(in db_name : String) : void
getDescription() : String
setDescription(in description : String) : void
isEnable() : boolean
setEnable(in enable : boolean) : void
getPlatform() : String
setPlatform(in platform : String) : void

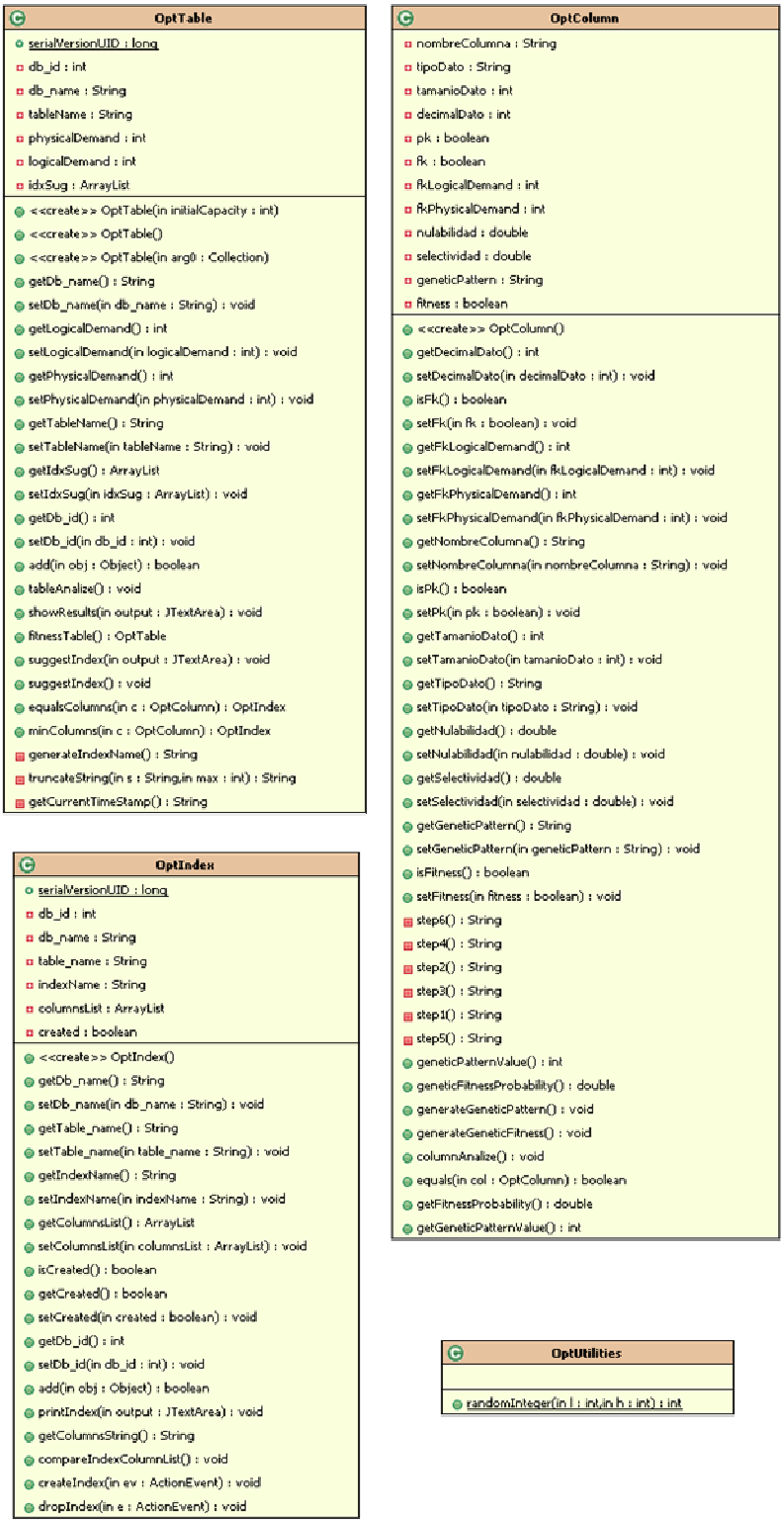

ObjectDTO
object_name : String
db_id : int
create_date : Date
schema : String
enable : boolean
object_type : String
parent_object_name : String
parent_db_id : int
date_current_reading : Date
date_last_reading : Date
current_physical_reading : int
last_physical_reading : int
current_logical_reading : int
last_logical_reading : int
<<create>> ObjectDTO()
getCreate_date() : Date
setCreate_date(in create_date : Date) : void
getCurrent_logical_reading() : int
setCurrent_logical_reading(in current_logical_reading : int) : void
getCurrent_physical_reading() : int
setCurrent_physical_reading(in current_physical_reading : int) : void
getDate_current_reading() : Date
setDate_current_reading(in date_current_reading : Date) : void
getDate_last_reading() : Date
setDate_last_reading(in date_last_reading : Date) : void
getDb_id() : int
setDb_id(in db_id : int) : void
isEnable() : boolean
setEnable(in enable : boolean) : void
getLast_logical_reading() : int
setLast_logical_reading(in last_logical_reading : int) : void
getLast_physical_reading() : int
setLast_physical_reading(in last_physical_reading : int) : void
getObject_name() : String
setObject_name(in object_name : String) : void
getObject_type() : String
setObject_type(in object_type : String) : void
getParent_db_id() : int
setParent_db_id(in parent_db_id : int) : void
getParent_object_name() : String
setParent_object_name(in parent_object_name : String) : void
getSchema() : String
setSchema(in schema : String) : void


ColumnDTO
column_name : String
object_name : String
db_id : int
data_type : String
length : int
decimal_length : int
pk : boolean
fk : boolean
fk_object_name : String
<<create>> ColumnDTO()
getColumn_name() : String
setColumn_name(in column_name : String) : void
getData_type() : String
setData_type(in data_type : String) : void
getDb_id() : int
setDb_id(in db_id : int) : void
getDecimal_length() : int
setDecimal_length(in decimal_length : int) : void
isFk() : boolean
setFk(in fk : boolean) : void
getLength() : int
setLength(in length : int) : void
getObject_name() : String
setObject_name(in object_name : String) : void
isPk() : boolean
setPk(in pk : boolean) : void
getFk_object_name() : String
setFk_object_name(in fk_object_name : String) : void

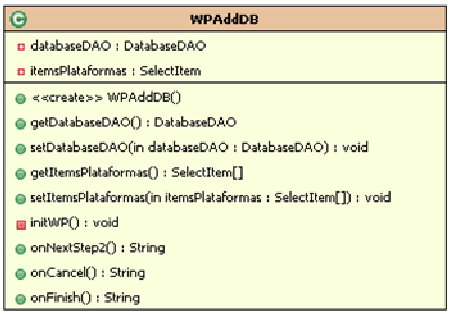

IndexColumnDTO
column_name : String
object_name : String
index_name : String
db_id : int
ord : int
<<create>> IndexColumnDTO()
getColumn_name() : String
setColumn_name(in column_name : String) : void
getDb_id() : int
setDb_id(in db_id : int) : void
getObject_name() : String
setObject_name(in object_name : String) : void
getOrd() : int
setOrd(in ord : int) : void
getIndex_name() : String
setIndex_name(in index_name : String) : void

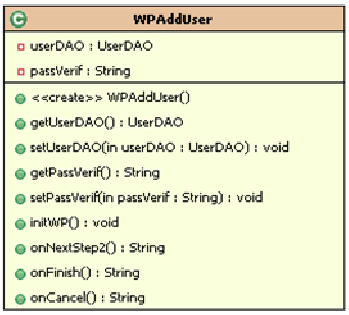

HistoricalStatisticsDTO
id : int
object_name : String
db_id : int
date_insert : Date
time_insert : Date
physical_demand : double
logical_demand : double
<<create>> HistoricalStatisticsDTO()
getDate_insert() : Date
setDate_insert(in date_insert : Date) : void
getDb_id() : int
setDb_id(in db_id : int) : void
getId() : int
setId(in id : int) : void
getLogical_demand() : double
setLogical_demand(in logical_demand : double) : void
getObject_name() : String
setObject_name(in object_name : String) : void
getPhysical_demand() : double
setPhysical_demand(in physical_demand : double) : void
getTime_insert() : Date
setTime_insert(in time_insert : Date) : void

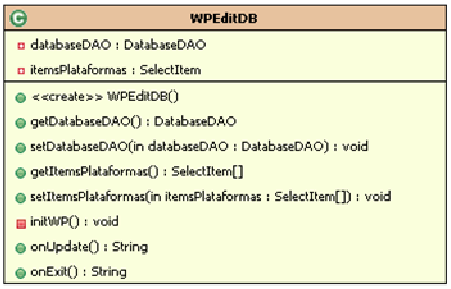

UserDTO
user_name : String
first_name : String
last_name : String
email : String
admin : boolean
exp_date : Date
password : String
<<create>> UserDTO()
getEmail() : String
setEmail(in email : String) : void
getExp_date() : Date
setExp_date(in exp_date : Date) : void
getFirst_name() : String
setFirst_name(in first_name : String) : void
isAdmin() : boolean
setAdmin(in admin : boolean) : void
getLast_name() : String
setLast_name(in last_name : String) : void
getPassword() : String
setPassword(in password : String) : void
getUser_name() : String
setUser_name(in user_name : String) : void

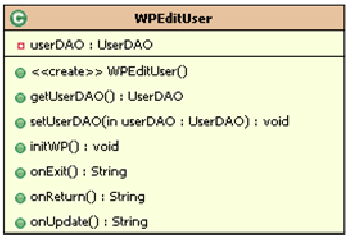

OptTable
serialVersionUID : long
db_id : int
db_name : String
tableName : String
physicalDemand : int
logicalDemand : int
idxSug : ArrayList
<<create>> OptTable(in initialCapacity : int)
<<create>> OptTable()
<<create>> OptTable(in arg0 : Collection)
getDb_name() : String
setDb_name(in db_name : String) : void
getLogicalDemand() : int
setLogicalDemand(in logicalDemand : int) : void
getPhysicalDemand() : int
setPhysicalDemand(in physicalDemand : int) : void
getTableName() : String
setTableName(in tableName : String) : void
getIdxSug() : ArrayList
setIdxSug(in idxSug : ArrayList) : void
getDb_id() : int
setDb_id(in db_id : int) : void
add(in obj : Object) : boolean
tableAnalize() : void
showResults(in output : JTextArea) : void
fitnessTable() : OptTable
suggestIndex(in output : JTextArea) : void
suggestIndex() : void
equalsColumns(in c : OptColumn) : OptIndex
minColumns(in c : OptColumn) : OptIndex
generateIndexName() : String
truncateString(in s : String,in max : int) : String
getCurrentTimeStamp() : String
OptColumn
nombreColumna : String
tipoDato : String
tamanioDato : int
decimalDato : int
pk : boolean
fk : boolean
fkLogicalDemand : int
fkPhysicalDemand : int
nulabilidad : double
selectividad : double
geneticPattern : String
fitness : boolean
<<create>> OptColumn()
getDecimalDato() : int
setDecimalDato(in decimalDato : int) : void
isFk() : boolean
setFk(in fk : boolean) : void
getFkLogicalDemand() : int
setFkLogicalDemand(in fkLogicalDemand : int) : void
getFkPhysicalDemand() : int
setFkPhysicalDemand(in fkPhysicalDemand : int) : void
getNombreColumna() : String
setNombreColumna(in nombreColumna : String) : void
isPk() : boolean
setPk(in pk : boolean) : void
getTamanioDato() : int
setTamanioDato(in tamanioDato : int) : void
getTipoDato() : String
setTipoDato(in tipoDato : String) : void
getNulabilidad() : double
setNulabilidad(in nulabilidad : double) : void
getSelectividad() : double
setSelectividad(in selectividad : double) : void
getGeneticPattern() : String
setGeneticPattern(in geneticPattern : String) : void
isFitness() : boolean
setFitness(in fitness : boolean) : void
step6() : String
step4() : String
step2() : String
step3() : String
step1() : String
step5() : String
geneticPatternValue() : int
geneticFitnessProbability() : double
generateGeneticPattern() : void
generateGeneticFitness() : void
columnAnalize() : void
equals(in col : OptColumn) : boolean
getFitnessProbability() : double
getGeneticPatternValue() : int
OptIndex
serialVersionUID : long
db_id : int
db_name : String
table_name : String
indexName : String
columnsList : ArrayList
created : boolean
<<create>> OptIndex()
getDb_name() : String
setDb_name(in db_name : String) : void
getTable_name() : String
setTable_name(in table_name : String) : void
getIndexName() : String
setIndexName(in indexName : String) : void
getColumnsList() : ArrayList
setColumnsList(in columnsList : ArrayList) : void
isCreated() : boolean
getCreated() : boolean
setCreated(in created : boolean) : void
getDb_id() : int
setDb_id(in db_id : int) : void
add(in obj : Object) : boolean
printIndex(in output : JTextArea) : void
getColumnsString() : String
compareIndexColumnList() : void
createIndex(in ev : ActionEvent) : void
dropIndex(in e : ActionEvent) : void
OptUtilities
randomInteger(in l : int,in h : int) : int

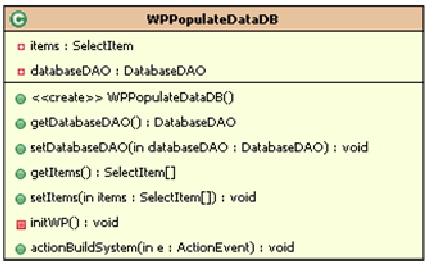

ChartDBDemand
historicDAO : HistoricalStatisticsDAO
minDate : Date
maxDate : Date
<<create>> ChartDBDemand()
getHistoricDAO() : HistoricalStatisticsDAO
setHistoricDAO(in historicDAO : HistoricalStatisticsDAO) : void
getMaxDate() : Date
setMaxDate(in maxDate : Date) : void
getMinDate() : Date
setMinDate(in minDate : Date) : void
addChartDataLogical(in chart : jspChart) : void
addChartDataPhysical(in chart : jspChart) : void
ConectionNode
<<create>> ConectionNode()
getPgConexion() : Connection
getPgConexion(in commit : boolean) : Connection
getOraConexion() : Connection
getOraConexion(in user : String,in pass : String,in host : String,in sid : String) : Connection
tryToCommit(in dbNode : Connection) : void
tryToClose(in dbNode : Connection) : void
EjbAbstract
ejbInterface : EjbInterface
LB_FIRSTPAGE : String
LB_LASTPAGE : String
LB_NEXTPAGE : String
LB_PREVPAGE : String
array : ArrayList
scrollPageSize : int
scrollCurrentPage : int
change : boolean
refresh : boolean
edit : boolean
delete : boolean
error : String
warning : String
errors : ArrayList
getArray() : ArrayList
setArray(in array : ArrayList) : void
getScrollMaxPage() : String
getScrollCurrentPage() : int
setScrollCurrentPage(in scrollCurrentPage : int) : void
getEjbInterface() : EjbInterface
setEjbInterface(in ejbInterface : EjbInterface) : void
getScrollPageSize() : int
setScrollPageSize(in scrollPageSize : int) : void
setScrollPageSize(in scrollPageSize : String) : void
getError() : String
setError(in error : String) : void
getErrors() : ArrayList
setErrors(in errors : ArrayList) : void
getWarning() : String
setWarning(in warning : String) : void
isChange() : boolean
setChange(in change : boolean) : void
isRefresh() : boolean
setRefresh(in refresh : boolean) : void
isDelete() : boolean
setDelete(in delete : boolean) : void
isEdit() : boolean
setEdit(in edit : boolean) : void
getErrorsStr() : String
getLB_FIRSTPAGE() : String
getLB_LASTPAGE() : String
getLB_NEXTPAGE() : String
getLB_PREVPAGE() : String
getHaveToSaveAll() : boolean
getHaveToDeleteAnything() : boolean
arrayPageFilter() : ArrayList
getArrayPage() : ArrayList
getShowLastPage() : boolean
getShowFirstPage() : boolean
getPageSizeList() : SelectItem[]
scrollNextPageAction(in e : ActionEvent) : void
scrollPrevPageAction(in e : ActionEvent) : void
scrollFirstPageAction(in e : ActionEvent) : void
scrollLastPageAction(in e : ActionEvent) : void
deleteAllAction(in e : ActionEvent) : void
updateAllAction(in e : ActionEvent) : void
refreshAllAction(in e : ActionEvent) : void
clearAction(in e : ActionEvent) : void
forceToRefresh(in e : ActionEvent) : void
getPgConection() : Connection
getPgConection(in commit : boolean) : Connection
getOraConection(in user : String,in pass : String,in host : String,in sid : String) : Connection
tryToCloseConection(in c : Connection) : void
tryToClosePreparedStatement(in ps : PreparedStatement) : void
tryToCloseResultSet(in rs : ResultSet) : void
getFormatDate(in ts_date : Timestamp) : String
convertToTimeStampFecha(in fecha : String) : Timestamp
booleanYesNoMapping(in b : boolean) : String
stringYesNoMapping(in s : String) : boolean
nextval(in seq : String) : String
existError() : boolean
getStyleToolbar() : ObjStyleClassesToolbarNavigation
ChartTableDemand
historicDAO : HistoricalStatisticsDAO
minDate : Date
maxDate : Date
<<create>> ChartTableDemand()
getHistoricDAO() : HistoricalStatisticsDAO
setHistoricDAO(in historicDAO : HistoricalStatisticsDAO) : void
getMaxDate() : Date
setMaxDate(in maxDate : Date) : void
getMinDate() : Date
setMinDate(in minDate : Date) : void
addChartDataLogical(in chart : jspChart) : void
addChartDataPhysical(in chart : jspChart) : void
<<Interface>>
EjbInterface
queryId() : void
query() : void
insert() : void
delete() : void
update() : void
clear() : void
select() : String
selectRow(in o : Object) : void
OraStatsLabels
OBJ_TABLE : String
STA_LOGICS_READS : String
STA_PHYSICS_READS : String
STA_PHYSICS_WRITES : String
STA_PHYSICS_READS_DIR : String
STA_PHYSICS_WRITES_DIR : String
ROW_LOCK : String
getStatsTypeList() : ArrayList

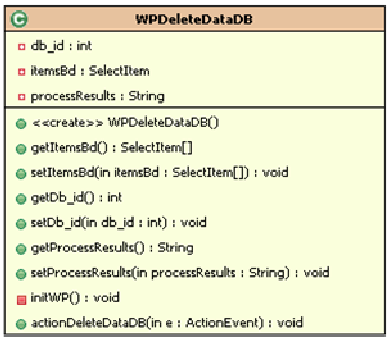

WPAddDB
databaseDAO : DatabaseDAO
itemsPlataformas : SelectItem
<<create>> WPAddDB()
getDatabaseDAO() : DatabaseDAO
setDatabaseDAO(in databaseDAO : DatabaseDAO) : void
getItemsPlataformas() : SelectItem[]
setItemsPlataformas(in itemsPlataformas : SelectItem[]) : void
initWP() : void
onNextStep2() : String
onCancel() : String
onFinish() : String

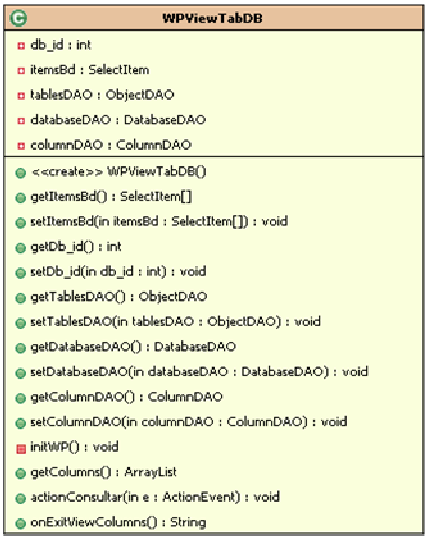

WPAddUser
userDAO : UserDAO
passVerif : String
<<create>> WPAddUser()
getUserDAO() : UserDAO
setUserDAO(in userDAO : UserDAO) : void
getPassVerif() : String
setPassVerif(in passVerif : String) : void
initWP() : void
onNextStep2() : String
onFinish() : String
onCancel() : String

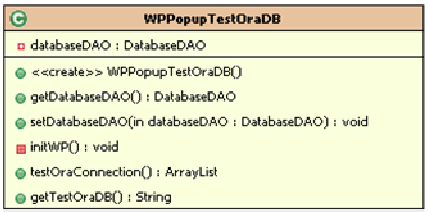

WPEditDB
databaseDAO : DatabaseDAO
itemsPlataformas : SelectItem
<<create>> WPEditDB()
getDatabaseDAO() : DatabaseDAO
setDatabaseDAO(in databaseDAO : DatabaseDAO) : void
getItemsPlataformas() : SelectItem[]
setItemsPlataformas(in itemsPlataformas : SelectItem[]) : void
initWP() : void
onUpdate() : String
onExit() : String


WPEditUser
userDAO : UserDAO
<<create>> WPEditUser()
getUserDAO() : UserDAO
setUserDAO(in userDAO : UserDAO) : void
initWP() : void
onExit() : String
onReturn() : String
onUpdate() : String


WPPopulateDataDB
items : SelectItem
databaseDAO : DatabaseDAO
<<create>> WPPopulateDataDB()
getDatabaseDAO() : DatabaseDAO
setDatabaseDAO(in databaseDAO : DatabaseDAO) : void
getItems() : SelectItem[]
setItems(in items : SelectItem[]) : void
initWP() : void
actionBuildSystem(in e : ActionEvent) : void


WPDeleteDataDB
db_id : int
itemsBd : SelectItem
processResults : String
<<create>> WPDeleteDataDB()
getItemsBd() : SelectItem[]
setItemsBd(in itemsBd : SelectItem[]) : void
getDb_id() : int
setDb_id(in db_id : int) : void
getProcessResults() : String
setProcessResults(in processResults : String) : void
initWP() : void
actionDeleteDataDB(in e : ActionEvent) : void


WPViewTabDB
db_id : int
itemsBd : SelectItem
tablesDAO : ObjectDAO
databaseDAO : DatabaseDAO
columnDAO : ColumnDAO
<<create>> WPViewTabDB()
getItemsBd() : SelectItem[]
setItemsBd(in itemsBd : SelectItem[]) : void
getDb_id() : int
setDb_id(in db_id : int) : void
getTablesDAO() : ObjectDAO
setTablesDAO(in tablesDAO : ObjectDAO) : void
getDatabaseDAO() : DatabaseDAO
setDatabaseDAO(in databaseDAO : DatabaseDAO) : void
getColumnDAO() : ColumnDAO
setColumnDAO(in columnDAO : ColumnDAO) : void
initWP() : void
getColumns() : ArrayList
actionConsultar(in e : ActionEvent) : void
onExitViewColumns() : String


WPPopupTestOraDB
databaseDAO : DatabaseDAO
<<create>> WPPopupTestOraDB()
getDatabaseDAO() : DatabaseDAO
setDatabaseDAO(in databaseDAO : DatabaseDAO) : void
initWP() : void
testOraConnection() : ArrayList
getTestOraDB() : String

2.7.5. Diseño del Esquema de Seguridad

Debido a la naturaleza del sistema, no podemos permitir que cualquier usuario tenga acceso a las operaciones de creación índices, una operación que por ser en línea puede ser muy delicada. Por esta razón, para acceder al sistema el usuario debe poseer una cuenta (nombre de usuario y contraseña), conocida solo por el propio usuario.

Además como medida adicional de seguridad se encripta la clave con el algoritmo criptográfico SHA-1 [14], y de esta forma se almacena en la base de datos.

Debido a que se trata de un prototipo, no se han implementado otras seguridades tales como uso de un contenedor de servlets seguro (https), encriptación de datos, encriptación de variables de sesión, certificados digitales, entre otros. Para una versión final, se recomienda el uso de este tipo de seguridades para agregar un mayor nivel de control y evitar ataques.

2.7.6. Integración de componentes

Dada la naturaleza de nuestro sistema, una aplicación web, no podíamos recrear el escenario modelo para realizar las pruebas que validen nuestras conclusiones. No todos los componentes del sistema son desarrollados utilizando las mismas herramientas de programación debido a que las funciones de la una diferían mucho la una de la otra y la herramienta que se utilizó no brindaba completo soporte para implementar la funcionalidad requerida. Los principales componentes identificados que forman el sistema son:

- La aplicación web (el sistema en sí)
- El motor de replicación
- El agente simulador de consultas
- La base de datos de producción

De los cuatro componentes identificados, la aplicación web y el motor son internos, mientras que los restantes, el agente y la base de datos de producción, son externos.

La aplicación web es la que brinda al usuario toda la funcionalidad implementada en este proyecto. El motor de replicación, una aplicación Java, es la encargada de leer las estadísticas de Oracle e insertar los registros en la base de datos del sistema. Además, este motor interactúa con el sistema pues solo se replican datos de tablas activas y de una base de datos activa.

El agente simulador, una aplicación Java, emula las consultas que las aplicaciones de la empresa harían sobre la base de datos monitoreada. Finalmente, la base de datos de producción, Oracle, es una base de pruebas que sirve como base ejemplo de un sistema real sobre el cual existen aplicaciones realizando consultas.

A continuación detallamos gráficamente los componentes del sistema y su interacción:

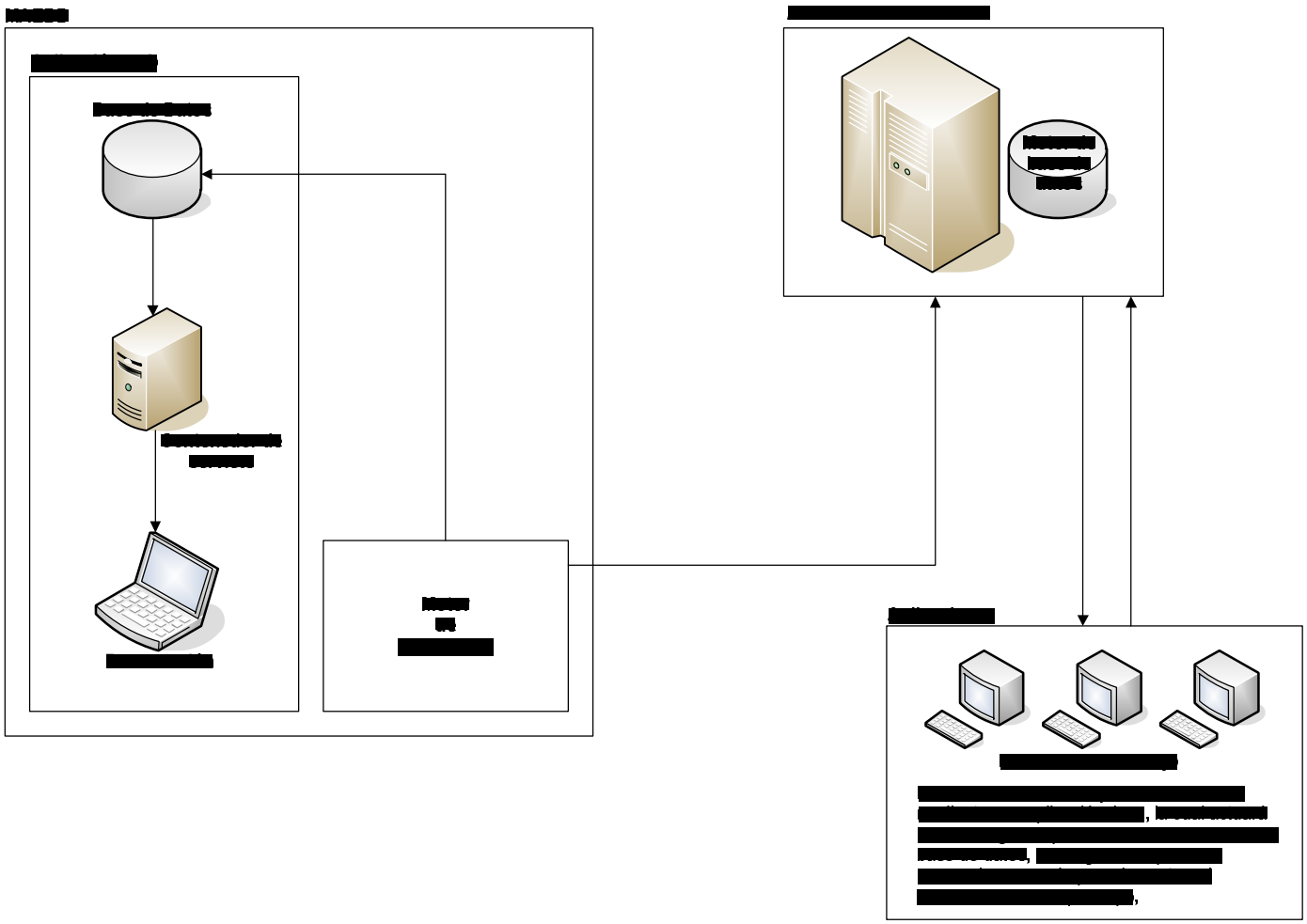


Figura 2.18 – Integración de componentes de la aplicación

CAPÍTULO 3

3. SIMULACIÓN Y REPRESENTACIÓN DE ESCENARIOS

3.1. Esquema de simulación de escenarios

La importancia de explicar clara y detalladamente los fundamentos usados en las pruebas del MAEBD, radica en poder cumplir el principal objeto que tiene la aplicación, que es ayudar a administrar correctamente los índices de una Base de Datos, en especial si esta base de datos es muy grande. Administrar los índices de un esquema de base de datos puede resultar muy complicado para un DBA, y en mucho de los casos debido a esto nunca se realizan ajustes en sus índices y estos ajustes sólo se realizan después que el sistema de base de datos el cual se desea mejorar su rendimiento, haya presentados problemas en los tiempos de consulta, lo cual puede resultar muy costoso para el negocio que ésta representa.

Para este análisis es necesario especificar los actores que intervienen en el mismo:

- Un sistema de base de datos (el cual llamaremos sistema de base de datos externo)
- Aplicación (la cual va a ser la interfaz mediante el cual los usuarios de la aplicación van a acceder al sistema de base de datos externo)
- El DBA (que es la persona que realiza los ajustes del motor para un mejor rendimiento)

A continuación detallaremos cada uno de los componentes que intervienen en la simulación, así como la función que cumplen cada uno de los mismos.

Nombre del Objeto	Objeto que Representa	Descripción	
Sistema Externo	Modelo de base de datos con datos de ejemplo	Este objeto representa el sistema al cual se le desea mejorar su rendimiento en tiempo de respuesta de las consultas que realiza una aplicación X sobre la misma.	
Aplicación Externa	Interfaz del Sistema Externo	Este objeto representa la aplicación que hace de interfaz con la que las personas acceden a la aplicación que maneja los datos del Sistema Externo.	
DBA	La persona que tiene privilegios para realizar ajustes en el sistema externo	Dentro del esquema de simulación el DBA cumple un papel importante, ya que ésta persona es la que va a manejar el MAEBD.	
Monitor	El sistema MAEBD	Este objeto es la aplicación la cual actúa como una herramienta para el Sistema Externo	

Tabla 3.1 – Componentes de la simulación de escenarios

3.2. Especificación de requerimientos y limitantes para un escenario válido

Es necesario poder especificar los requerimientos y limitantes importantes que se tomaron en cuenta dentro del esquema de simulaciones para poder tener un fundamento claro y sólido de validez del escenario usado para la simulación.

Como primer requerimiento, debemos tener una base de datos con la suficiente cantidad de datos para que pueda simular los grandes sistemas para la cuál está dirigida esta herramienta. Es importante este requerimiento en el escenario de simulación ya que para poder ver movimientos considerables de las estadísticas de tablas e índices se deben tener gran cantidad de registros y sólo en estos casos es notoria la mejora en los tiempos de respuesta.

Otro punto importante que se debe tener en cuenta es que el usuario el cual se configura para la base de datos que se ingresa al MAEBD debe tener los privilegios suficientes para poder acceder a las tablas de catálogo de Oracle, que es donde se almacena toda la estructura de las tablas de un esquema específico. Este usuario también debe tener los suficientes privilegios para poder realizar consultas sobre cualquier esquema.

3.3. Simulación sobre un sistema ejemplo

Para representar un escenario válido que pueda generar las suficientes estadísticas como para poder mover la cantidad de datos necesarias para generar datos estadísticos y datos históricos ya que esos datos son la médula de la aplicación, fue indispensable crear una pequeña aplicación que simule la aplicación externa. Esto se logró conseguir almacenando en una pila un conjunto de consultas que se realizan comúnmente sobre el Sistema Externo de base de datos.

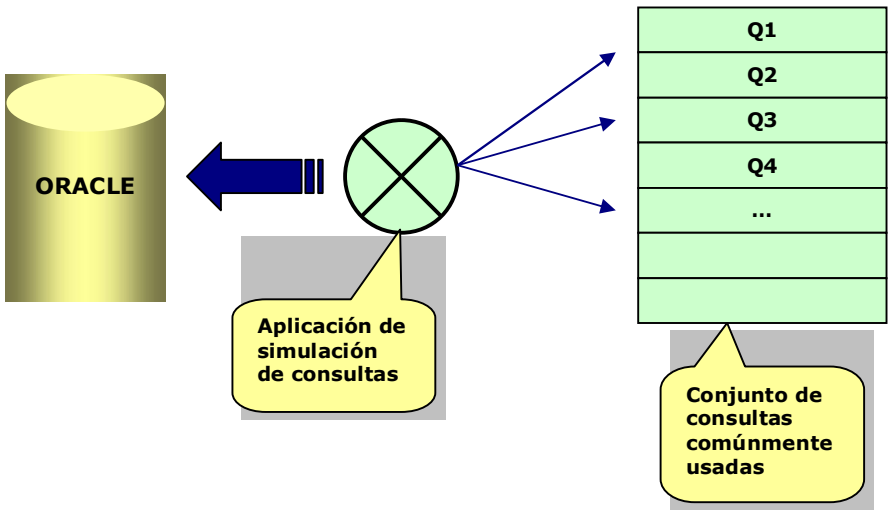


Figura 3.1 – Simulación de aplicaciones consultando sobre la base

Con este modelo de simulación se logró conseguir un escenario simulado muy apegado a la realidad. Ahora es importante describir el

otro lado del escenario de simulación usado, lo cual implica tener una aplicación que periódicamente realice consultas sobre la base de datos de Oracle de las demandas y las inserte en el MAEBD cada resultado de estas consultas. A este componente lo llamaremos el Replicador, el cual se encarga de hacer un barrido de las tablas activas de la base de datos que se encuentra activa (el sistema permite sólo una base de datos activa a la vez), y luego realizar la consulta de su demanda en la base en Oracle para luego retornar esos resultados e insertarlos en el historial de estadísticas del MAEBD. Finalmente el sistema utiliza estos datos para poder procesar y tomar decisiones sobre qué índices se pueden sugerir.

CAPÍTULO 4

4. IMPLEMENTACIÓN, DESPLIEGUE Y PRUEBAS

4.1. Implementación del prototipo

Como primer paso a la implementación del sistema se establecieron estándares propios para facilitar la implementación y mantenimiento del mismo. Entre los estándares que se acordaron figuran:

- Mantener una convención de nombres para atributos y métodos de las clases y páginas.
- Documentar las porciones de código complejas.
- Mantener una línea gráfica (estandarización de interfaces) para todo el sistema.

Posteriormente, se procedió a escoger el lenguaje de programación del sistema, el cual debía ser apropiado para una aplicación web; inicialmente se escogió JSP por nuestra experiencia en el área, sin embargo existían ciertas funcionalidades que agregaban gran complejidad a la implementación. Debido a esto, se decidió utilizar JSF (JavaServer Faces) una tecnología poco explorada en el medio y que ofrece mayor facilidad para obtener cierto nivel de funcionalidad requerido.

La implementación del prototipo incluye el desarrollo de 3 componentes:

- La aplicación web
- El motor de replicación
- El agente simulador de consultas

La aplicación web fue desarrollada utilizando JSF, una API (Advanced Programming Interface) para desarrollo de aplicaciones web, la cual es una evolución natural del desarrollo web utilizando J2EE y recientemente un estándar especificado: JSR 127 y parte de J2EE 1.5.

La estructura de una aplicación web utilizando la tecnología JSF está compuesta de lo siguiente:

- Una aplicación web estándar utilizando J2EE 1.3
- Archivos JAR: Commons y JSF, en el directorio WEB-INF/lib
- Servlet controlador de FACES configurado en el archivo Web.xml
- Archivo de configuración: faces-config.xml

De la misma manera está estructurada nuestra aplicación web. A continuación se presenta un diagrama básico mostrando la configuración de nuestro sistema:

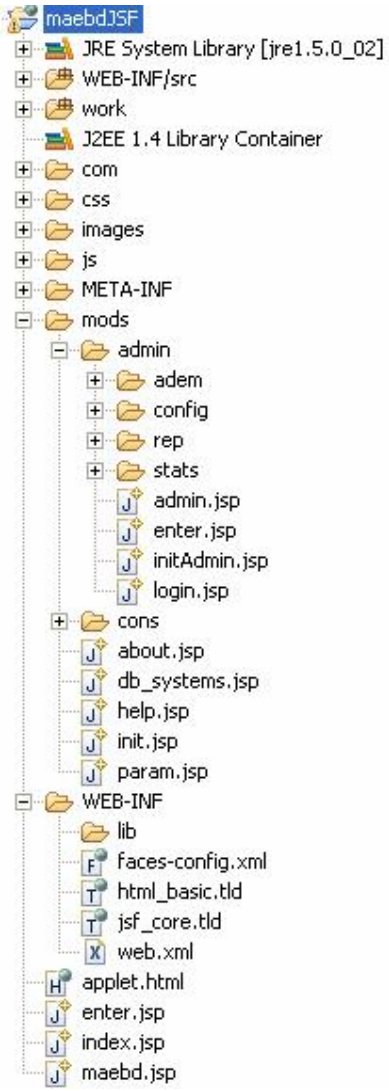


Figura 4.1 – Estructura de la aplicación

Una aplicación JSF tradicional consiste básicamente de:

- Páginas JSP con componentes UI (User Interface) encapsulados en librerías de tags JSP tales como CORE, HTML, etc.
- Un Modelo de Navegación (el modelo del sistema es presentado en la sección 2.5) especificado en el archivo faces-config.xml
- Un set de beans manejados que facilitan la lógica UI de la aplicación.

“Las páginas JSF son representadas como un árbol de componentes UI llamado Vista. El ciclo de vida de las páginas JSF comienza cuando el cliente solicita una página.” [15]

No ahondaremos en el ciclo de vida de las páginas JSF, debido a que el presente trabajo no está enfocado al estudio de esta tecnología, sin embargo consideramos necesario mostrar el ciclo de vida debido a que JSF está recién despegando y no está muy difundido en nuestro medio.

En la Figura 4.1, se muestran los 6 pasos que existen durante el ciclo y algunas generalidades:

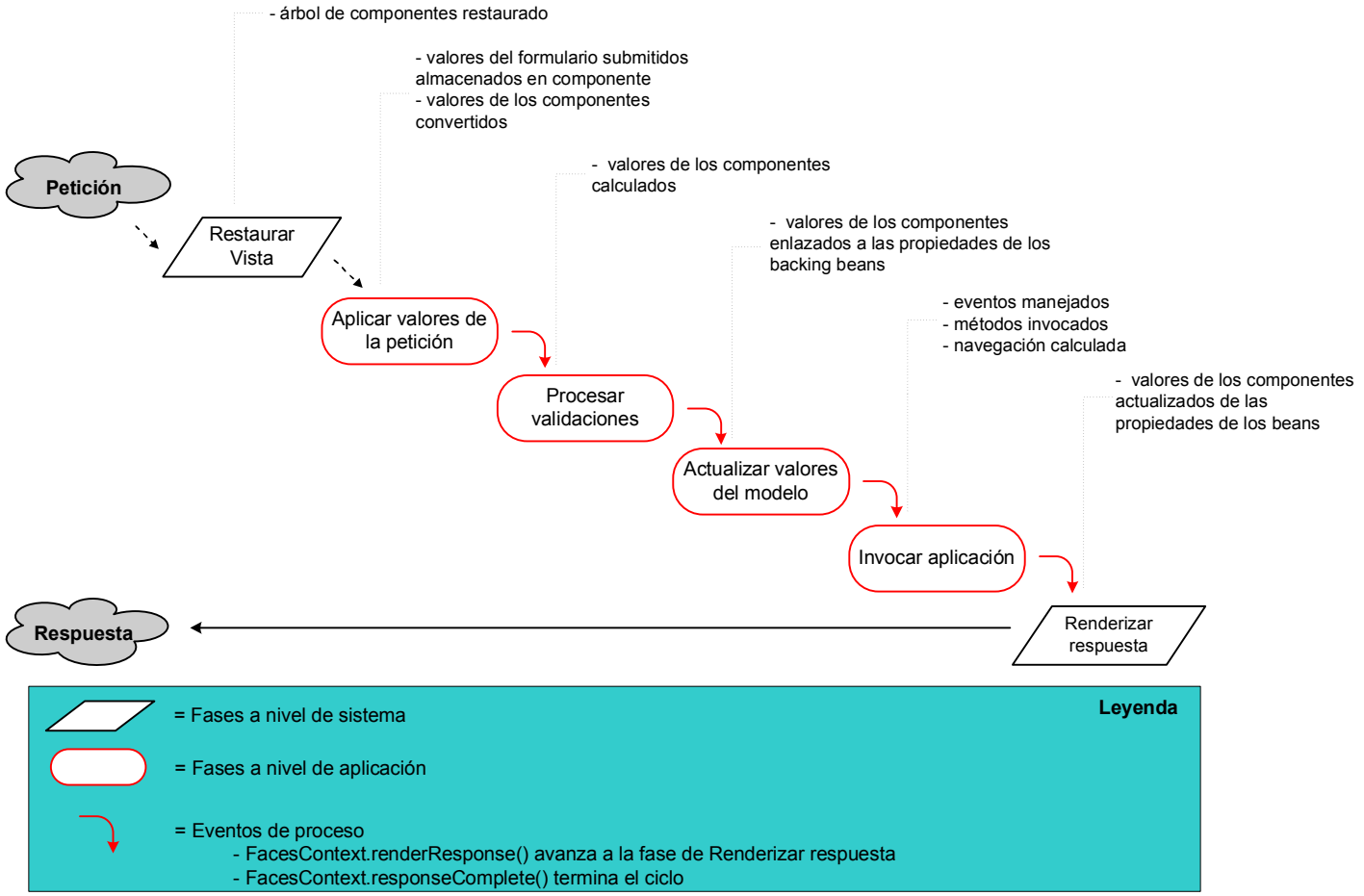


Figura 4.2 – Ciclo de vida de las páginas JSF [16]

4.2. Despliegue del prototipo

La instalación del sistema requiere de un nivel medio de conocimientos de sistemas operativos, bases de datos y aplicaciones web. Para instalar el sistema se necesita 3 componentes en el host:

- J2SDK 1.5 o superior
- Tomcat 5.x para Windows
- Postgres 8.x para Windows

Adicionalmente, se debe considerar el planificador de tareas con el cual automatizaremos el motor de replicación, un componente importante del sistema.

El host debe tener acceso a la red local para poder agregar sistemas para su posterior monitoreo. Debe tener las seguridades respectivas en cuanto a antivirus, firewall y otros.

4.3. Pruebas realizadas

Como primer paso para cuantificar los resultados, se creó una base de datos ejemplo propia a partir de un modelo de negocio sencillo, se crearon las estructuras necesarias (principalmente las tablas) y se generaron datos para poblar a dichas tablas. La información fue generada utilizando una aplicación Java desarrollada como parte de este proyecto.

Primeramente, se revisaron todos los módulos del sistema para probar su funcionalidad y determinar si existían errores por corregir. Luego, se probó el esquema de navegación implementado para verificar que todas las opciones de los menús lleven al usuario a donde indicaban. Adicionalmente se probó el esquema de seguridad implementado en lo referente a autenticación, expiración de sesión y encriptación de claves de usuario en el host. Finalmente, se verificó el correcto funcionamiento del motor de replicación para determinar si no existían problemas de comunicación o de funcionalidad.

Posteriormente, se realizaron las pruebas del módulo de análisis de demanda, el cual es la parte clave de este proyecto. Se simularon los

escenarios especificados en el capítulo 3 y se verificó que no exista ningún error de comunicación, funcionalidad y seguridad.

Luego de realizar todas las pruebas, se encontró un nivel aceptable de errores, los cuales eran causados por fallas en la lógica de la aplicación pero posteriormente fueron detectados y corregidos con éxito. Como paso siguiente se probó el sistema con algunos usuarios para detectar errores de interacción y se detectó un nivel bajo y aceptable de errores. En cuanto a Seguridad, el sistema manejó los esquemas de Seguridad implementados de manera satisfactoria, por lo cual no se presentaron errores.

En general, se encontró un nivel aceptable de errores en el sistema, los cuales han sido detectados y corregidos. Si existen nuevos errores que corregir, la facilidad de mantenimiento del mismo hará que dichos cambios se realicen sin mayores contratiempos exceptuando aquellos relacionados con nuevas funcionalidades.

CONCLUSIONES Y RECOMENDACIONES

Una vez culminado el proyecto, obtuvimos resultados que corroboraron nuestro planteamiento inicial que identificaba a los índices como objetos críticos en la optimización de bases de datos y el impacto que tiene un mal diseño y el escaso monitoreo de los mismos, el cual puede reflejarse como incremento en el tiempo de respuesta de las aplicaciones y el mayor consumo de recursos; a partir de esto concluimos lo siguiente:

- Los tiempos de respuesta de nuestra base de datos modelo disminuyeron al crear los índices utilizando las sugerencias del sistema de monitoreo, los cuales pudieron ser monitoreados para determinar su eficiencia y eficacia para el problema planteado.
- El monitoreo de índices mostró grandes resultados, pues antes esta tarea demandaba de un monitoreo constante y comprometía recursos si se utilizaban las herramientas nativas del motor de base de datos; ahora con el sistema de monitoreo no se comprometen los recursos y las herramientas de estadística permiten visualizar rápidamente la demanda por tablas y crear índices ágilmente.
- Los métodos heurísticos aplicados a problemas cuya solución no sea única y el resultado final dependa de la evaluación de muchas variables, pueden servir para resolver dichos problemas evaluando

sus ventajas y desventajas y aplicando el más adecuado para nuestro problema a resolver.

- Los recursos del servidor de base de datos no se vieron comprometidos pues el sistema de monitoreo de índices puede correr en otro servidor y solo realiza consultas específicas a tablas del sistema (motor de base de datos), lo cual evita que se produzcan bloqueos en las tablas de las aplicaciones o se incrementen los tiempos de respuesta.
- No caer en el error de que un sistema con más índices funciona más rápida y eficientemente, ya que existen otros factores que determinan el verdadero rendimiento de un sistema de base de datos, como por ejemplo el uso de recursos, la demanda de uso de tablas, entre otros. Esto plantea un problema cuya solución depende de múltiples variables de entrada lo cual no puede ser resuelto por métodos tradicionales.
- Integramos exitosamente la demanda de uso de un índice como una variable de entrada para determinar cuan óptima la creación del mismo puede resultar, esto gracias a la flexibilidad de algoritmo genético.

De la misma manera planteamos las siguientes recomendaciones:

- Hacer un análisis concienzudo del método heurístico a utilizar. No todos son apropiados para obtener una óptima solución y debemos tratar de simular los escenarios más frecuentes de manera que observemos el comportamiento del método y la rapidez para proveer una respuesta en base a las variables de entrada que debamos tomar en cuenta.
- Utilizar el sistema de monitoreo de índices de manera frecuente en las bases de datos con tiempos de respuesta de las aplicaciones altos. En muchos casos, la administración de sistemas, conforme va rotando, no detecta fácilmente problemas de diseño de índices; el sistema permite monitorear las estadísticas de demanda de uso y asistir al DBA en la toma de decisiones relacionadas con creación y eliminación de índices.
- Hay que recordar que éste sistema no es totalmente autosuficiente, esto es, que no siempre en todos los casos que los índices sean recomendados quiere decir que ese índice sea indispensable, ya que cabe recordar que el sistema con el pasar del tiempo podrá ir adquiriendo el conocimiento sobre la necesidad de crear ése índice en base a datos históricos.

Mejorar el diseño de conexión a bases de datos para disminuir el riesgo de brechas en la seguridad.

ANEXOS

Manual de Usuario

En este apéndice se detallarán los pasos para el uso del Usuario del Sistema de Monitoreo y Análisis de Bases de Datos (MAEBD). Para ingresar al sistema, se debe utilizar un navegador web y acceder a la siguiente dirección:

http://<ip_servidor_aplicaciones:puerto>/maebd

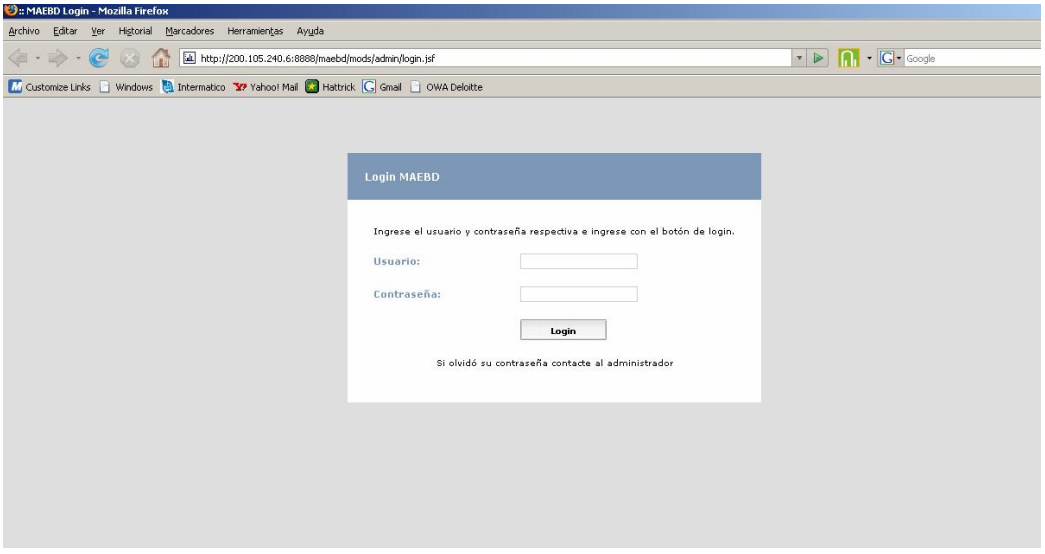


Figura A.1 – Ventana de inicio para acceso al sistema

La figura A.1 muestra la pantalla inicial del sistema, en la cual se ingresa al menú de la aplicación:

Para ingresar, tipeamos las credenciales respectivas y hacemos clic en **Login**:

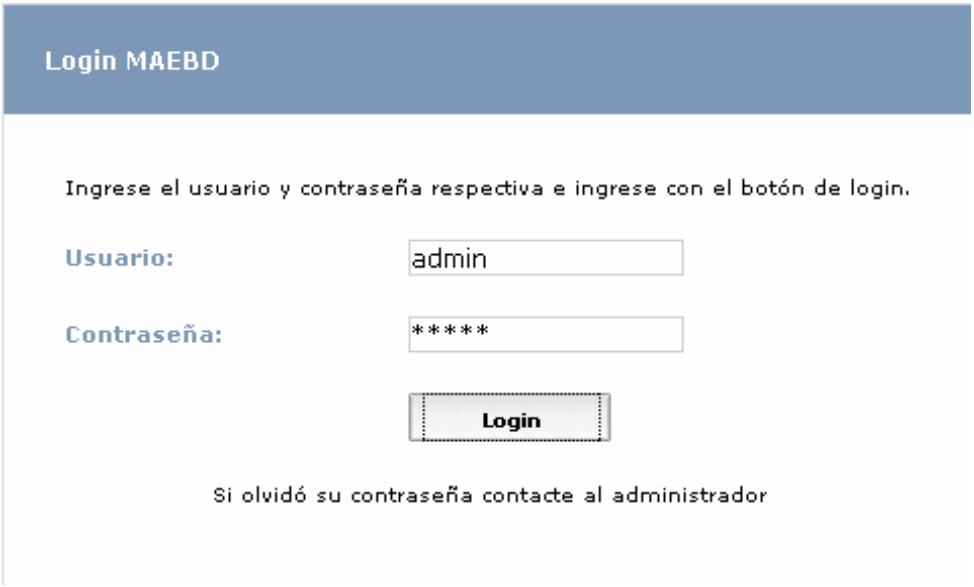


Figura A.2 – Ingreso a la aplicación: ingreso de credenciales

Si las credenciales proporcionadas no son válidas, el sistema nos mostrará un mensaje de error:

Figura A.3 – Mensaje de error al ingreso al sistema

Si se proporcionan credenciales válidas, el sistema redirecciona al usuario a la página de inicio de la aplicación:

Figura A.4 – Página de inicio del sistema

En la figura A.4, se muestra la página de inicio del módulo de administración; aquí se detallan todas las opciones que el DBA cuenta para realizar el monitoreo de índices y la administración del sistema en si. Los menús disponibles son:

1. Inicio
2. Configuración
3. Análisis de demanda
4. Estadísticas
5. Salir

Las funciones o procesos contenidos en cada una de estos menús se detallan a continuación:

1. **Inicio:** utilizar esta opción para ubicarse en la página de inicio del módulo de administración.

Figura A.5 – Opción Inicio

2. **Configuración:** en este menú se encuentran opciones de administración y configuración del sistema:
    - Seguridad
    - Sistema

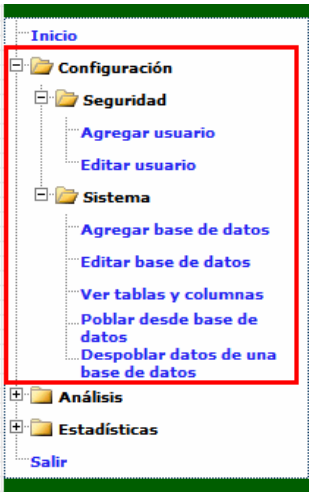


Figura A.6 – Opciones del menú Configuración

Seguridad: esta opción tiene funciones que sirven para administración de usuarios.

Sistema: aquí encontraremos funciones para administrar las bases de datos a monitorear.

3. **Análisis de Demanda:** en este menú se concentran las opciones de análisis y monitoreo de índices.

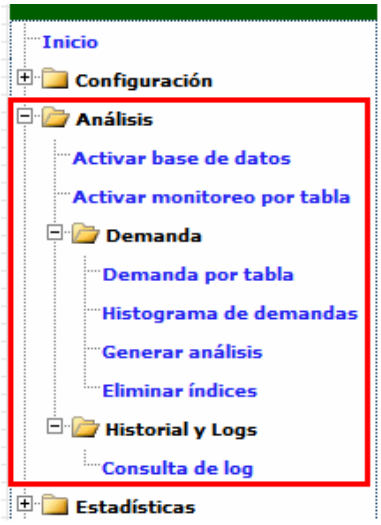


Figura A.7 – Opciones del menú Análisis de Demanda

4. **Estadísticas:** este menú contiene funciones que permiten visualizar la demanda de bases de datos, tablas e índices.

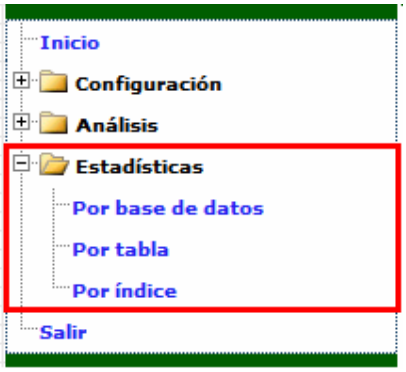


Figura A.8 – Opciones del menú Estadísticas

5. **Salir:** esta opción permite salir del módulo de administración.

Funciones más comunes

Esta sección detallará los pasos a seguir para ejecutar las funciones más comunes del sistema.

- **MONITOREO DE ÍNDICES**

Para monitorear los índices de una base de datos o monitorear si el uso de estos es requerido, debemos realizar las siguientes tareas:

- Activación de base de datos a monitorear
- Activación de tablas a monitorear
- Monitorear demanda de tablas
- Realizar análisis de demanda
- Crear índices
- Monitorear demanda de índices

Se detallarán los pasos necesarios para llevar a cabo cada uno de las tareas mencionadas anteriormente.

- **ACTIVACIÓN DE BASE DE DATOS PARA MONITOREO**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Análisis**.

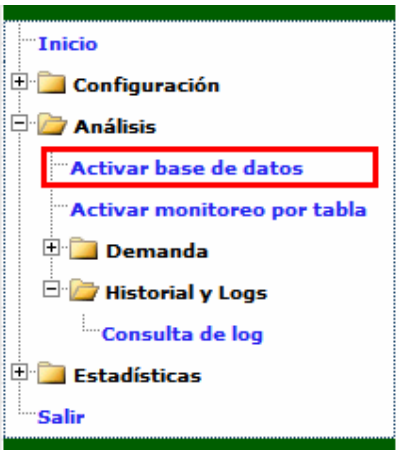


Figura A.9 – Opciones del menú Análisis

3. Escoger la opción **Activar Bases de Datos**.

Figura A.10 – Activación de una base de datos para monitoreo

4. Seleccionar la base de datos a monitorear y hacer clic en el casillero **Activar**. Solo se permite monitorear una base de datos a la vez.

5. Si todo se procesa correctamente, el sistema redireccionará al usuario a la página del menú Análisis de Demanda.

- **ACTIVACIÓN DE TABLAS PARA MONITOREO**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Análisis**.
3. Seleccionar la opción **Activar Monitoreo de Tablas**. Seleccione la base de datos donde están las tablas a monitorear y haga clic en **Actualizar**.

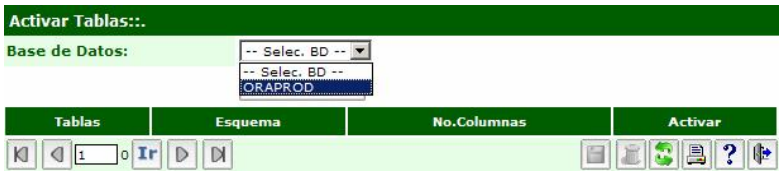


Figura A.11 – Activación de tablas para monitoreo

Solo aparecerá como opción la base de datos activada para monitoreo.

4. Seleccione las tablas que va a monitorear. Puede escoger más de una tabla para su monitoreo.

Figura A.12 – Activando tablas para monitorear

5. Si todo se procesa correctamente, el sistema redireccionará al usuario a la página del menú Análisis de Demanda.

- **MONITOREAR DEMANDA DE TABLAS**

1. Ingresar al módulo de administración con credenciales administrativas.

Esta tarea se puede realizar con varias herramientas del sistema, dependiendo de lo que el usuario necesite para tomar una decisión con relación a la creación de índices.

Las opciones disponibles para el monitoreo de tablas son:

- **Demanda por tablas**: información general de demanda de tablas, lecturas físicas y lógicas. No se debe especificar fechas.

- **Histograma de Demandas**: información muy detallada de demanda de tablas, lecturas físicas y lógicas. Se especifica una fecha determinada ingresado por el usuario y la información de demanda es mostrada a nivel de horas.
- **Estadísticas por Tablas**: información detallada de demanda de tablas. Muestra lecturas físicas y lógicas en un rango de fechas especificado por el usuario.

2. Dependiendo del enfoque aplicado en el monitoreo, el usuario escogerá la opción que se requiera:

Mediante Demanda de Tablas:

1. Ir al menú **Análisis**.
2. Seleccionar la opción **Demanda por Tablas**. Escoger la base de datos donde residen las tablas y hacer clic en **Actualizar**.

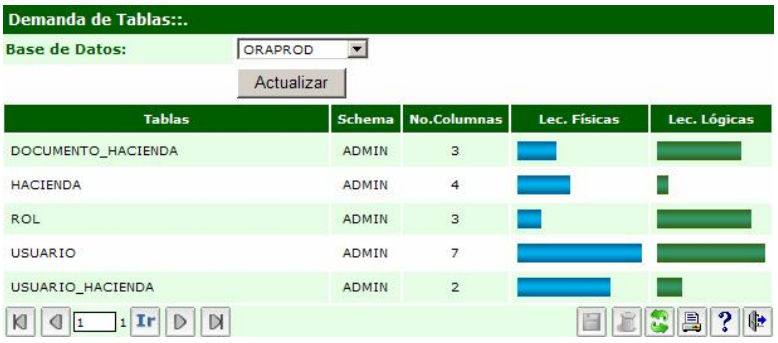


Figura A.13 – Demanda de tablas, monitoreo

Mediante Histograma de Demandas:

1. Ir al menú **Análisis**.

2. Seleccionar la opción **Histograma de Demandas**. Escoger la base de datos donde residen la tabla, escoger la tabla y la fecha de la cual se requiera la consulta. Luego, hacer clic en **Actualizar**.

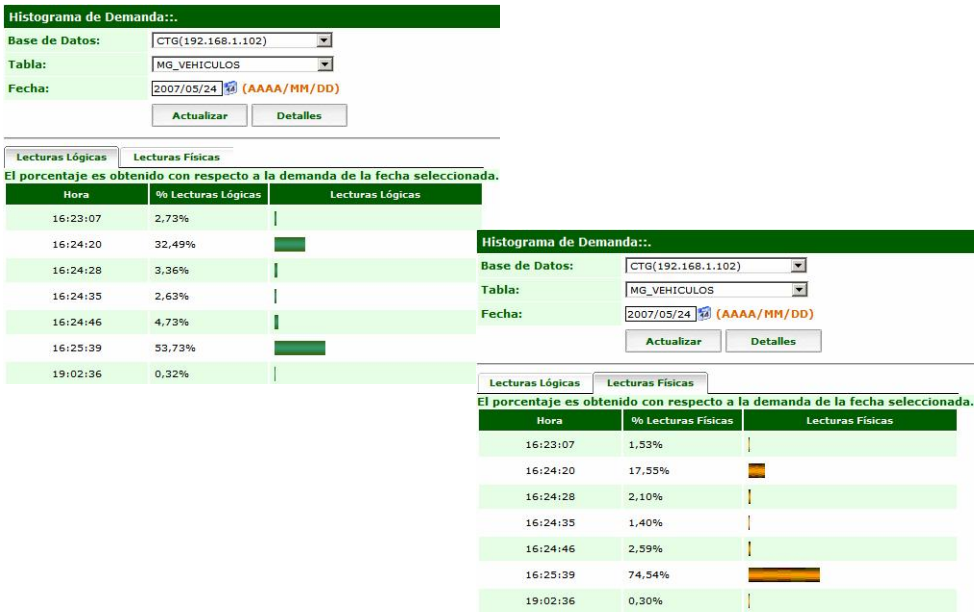


Figura A.14 – Histograma de demanda, monitoreo

Mediante Estadística por Tablas:

1. Ir al menú **Estadísticas**.
2. Escoger la opción **Estadísticas por Tablas**. Esta versátil herramienta permite visualizar gráficamente, mediante un gráfico de líneas, la demanda de una tabla específica en una base de datos específica en una fecha determinada, todos estos elementos ingresados por el usuario. Se muestran 2 gráficos, el verde representa las lecturas lógicas y el naranja las lecturas físicas.

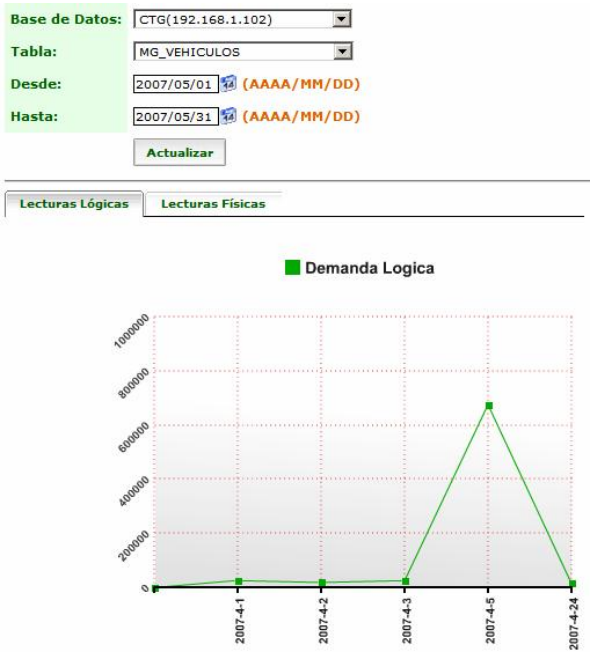


Figura A.15 – Estadísticas por Tablas (lecturas lógicas), monitoreo

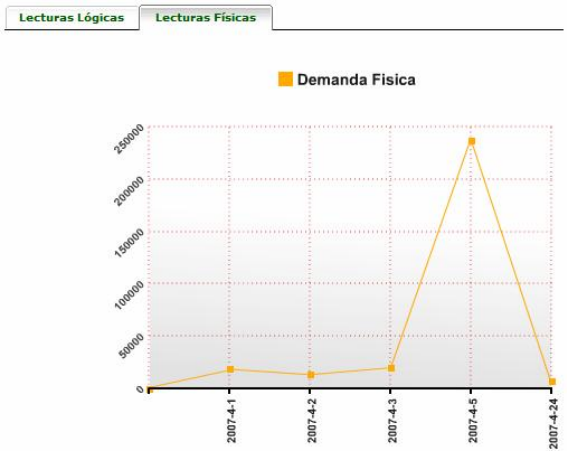


Figura A.16 - Estadísticas por Tablas (lecturas físicas), monitoreo

- **REALIZAR ANÁLISIS DE DEMANDA**

  Es la operación más importante del sistema puesto que es la que realiza el análisis de demanda de las tablas y genera los posibles índices a crear. Además toma más tiempo en ejecutar que el resto de funciones.

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Análisis**.
3. Escoger el submenú **Demanda**.
4. Seleccionar la opción **Generar Análisis**.

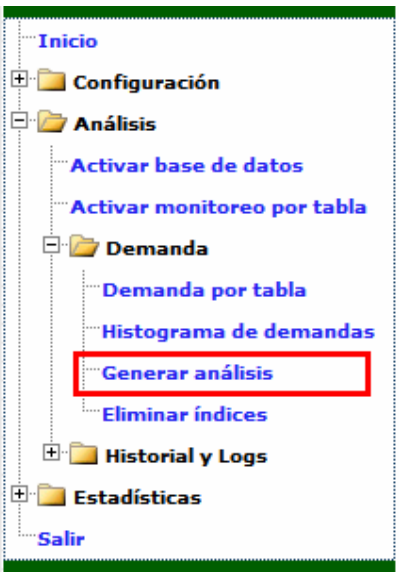


Figura A.17 – Análisis de Demanda

5. Escoger la base de datos activa y la tabla de la cual se va a generar el análisis de demanda y hacer clic en **Analizar**.

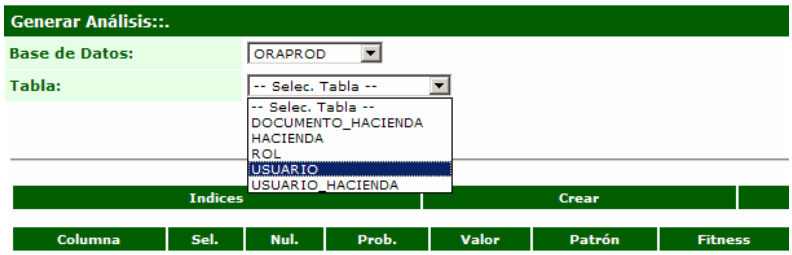


Figura A.18 – Generando análisis de demanda

6. El proceso tomará cierto tiempo y luego presentará la información procesada en base a la demanda de la tabla seleccionada.

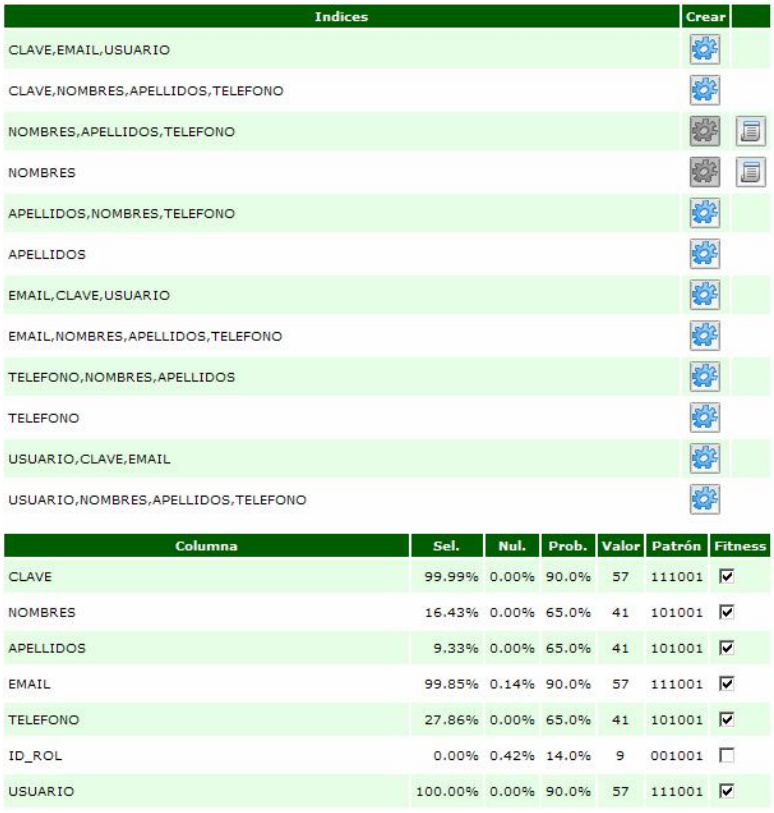


Figura A.19 – Análisis de demanda de una tabla, resultado

7. La información que se muestra corresponde las columnas de la tabla, su aptitud y otros datos importantes.

- **CREAR ÍNDICES**

1. Seguir los pasos 1, 2, 3, 4 y 5 de la tarea **REALIZAR ANÁLISIS DE DEMANDA.**
2. Parte de la información mostrada son los posibles índices a crear. Escoger el índice que se requiera crear y hacer clic en la columna **Crear**.

Figura A.20 – Creación de índices

3. Junto al icono de crear, está el icono de historia de índices para aquellas columnas que han formado índices anteriormente (creados con el sistema), hacer clic para ver información histórica de la combinación de columnas.

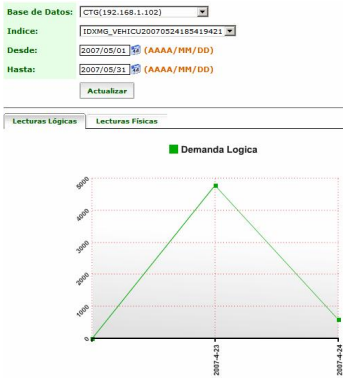


Figura A.21 – Información histórica de índices

- **MONITOREAR DEMANDA DE ÍNDICES**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Estadísticas**.
3. Seleccionar la opción **Estadísticas de Índices**. El usuario debe seleccionar la base de datos, el índice y el rango de fechas en el cual se va a consultar la demanda.

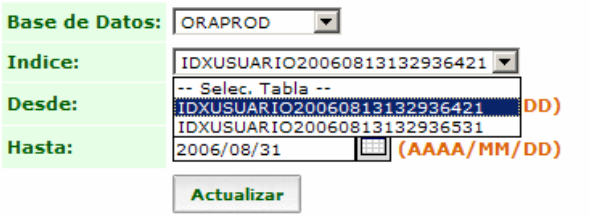


Figura A.22 – Estadísticas de índices

4. Las estadísticas se presentan gráficamente y están separadas lecturas lógicas (verde) y lecturas físicas (naranja). Es muy parecido a las demás estadísticas generadas por el sistema.

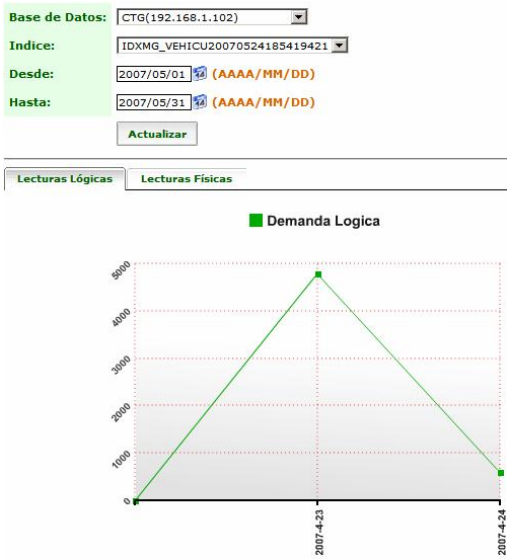


Figura A.23 – Estadísticas de índices: lecturas lógicas, resultado

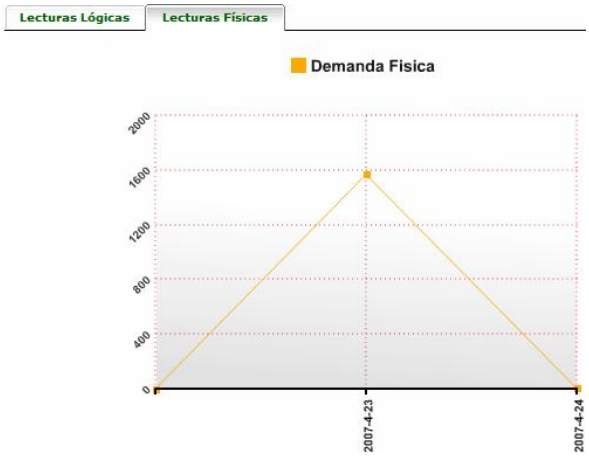


Figura A.24 – Estadísticas de índices: lecturas físicas, resultado

5. Si todo se procesa correctamente, debe presentarse las gráficas de demanda.

- **ELIMINAR ÍNDICES**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Análisis**.
3. Escoger el submenú **Demanda**.
4. Seleccionar la opción **Eliminar Índices**. Seleccione la base de datos donde están los índices a eliminar y haga clic en **Consultar**.

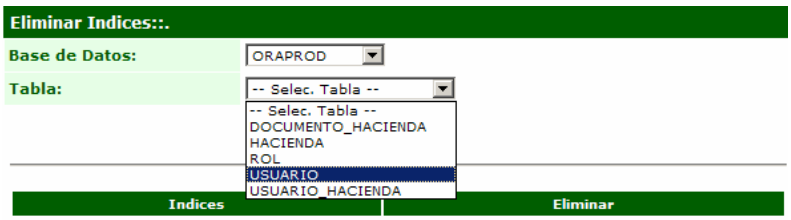


Figura A.25 – Eliminación de índices

5. Escoja el índice a eliminar y haga clic en el icono de Eliminar  .

Figura A.26 – Eliminando índices

Manual del Administrador

En este apéndice se detallarán los pasos para el uso del Administrador del Sistema de Monitoreo y Análisis de Bases de Datos. Para ingresar al sistema, se debe utilizar un navegador web y acceder a la siguiente dirección:

http://<ip_servidor_aplicaciones:puerto>/maebd

En caso de dudas acerca de cómo ingresar al módulo de administración, referirse al Apéndice A – Manual de Usuario.

Funciones más comunes

Esta sección detallará los pasos a seguir para ejecutar las funciones más comunes del administrador del sistema.

- **AGREGAR USUARIOS**
    1. Ingresar al módulo de administración con credenciales administrativas.
    2. Ir al menú **Configuración**.
    3. Escoger el submenú **Seguridad**.

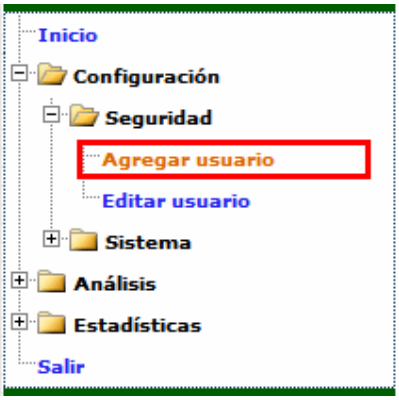


Figura B.1 – Opciones submenú Seguridad

4. Hacer clic en la opción **Agregar Usuarios**.
5. La primera pantalla le dará bienvenida, hacer clic en **Continuar**.
6. Llenar los datos del formulario. Aquellos datos marcados con asterisco (*) son obligatorios.

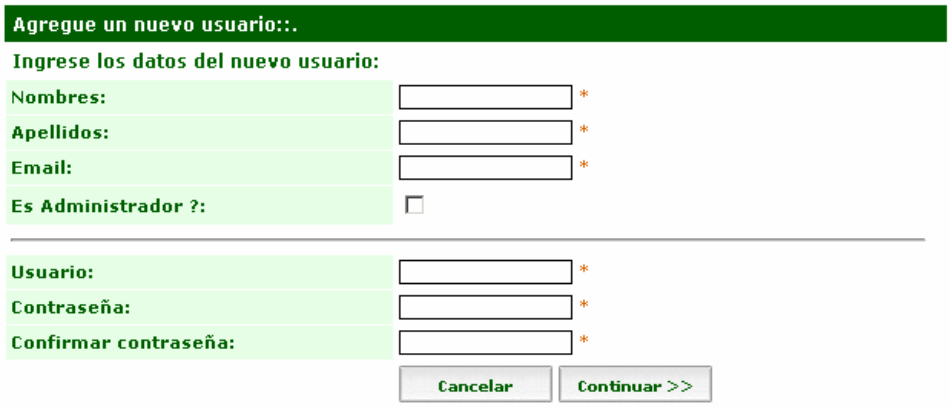


Figura B.2 – Formulario para agregar un nuevo usuario

7. Posibles errores:

a. No se han completado todos los campos obligatorios.

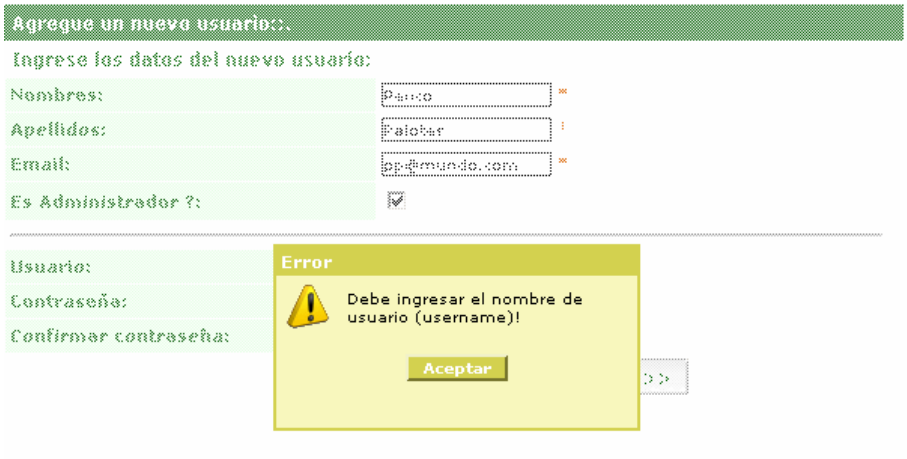


Figura B.3 – Error por campos obligatorios faltantes

b. Las contraseñas no coinciden.

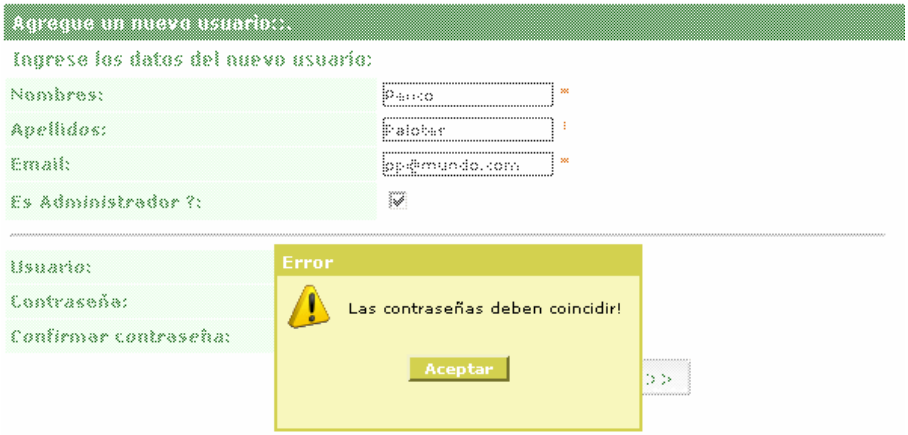


Figura B.4 – Error por contraseñas diferentes

c. El usuario ya existe.

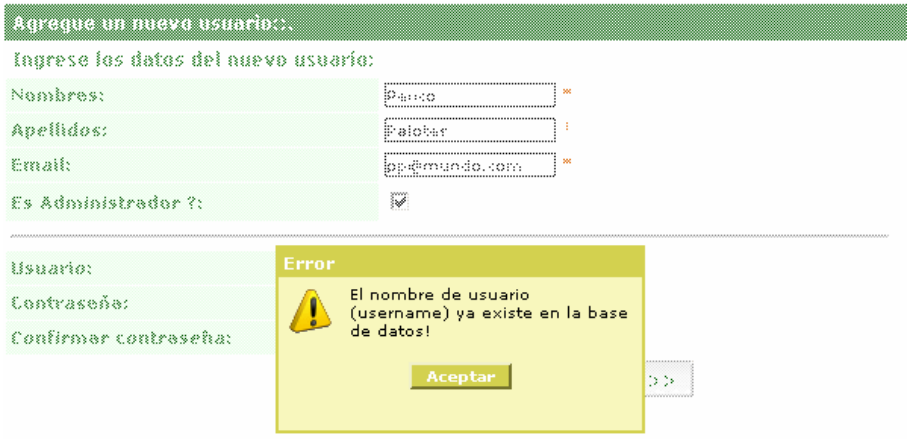


Figura B.5 – Error por usuario existente

d. Otros errores. En caso de encontrar otros errores, contactar al administrador.

8. Si todo se procesó exitosamente, se muestran los datos ingresados ya en la base de datos. Hacer clic en **Finalizar**.

Figura B.6 – Ingreso exitoso de un usuario

- **EDITAR USUARIOS**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Seguridad**.
4. Hacer clic en la opción **Editar Usuarios**.
5. Se mostrará una pantalla con todos los usuarios del sistema.

Figura B.7 – Edición de usuarios

6. Para editar un usuario, hacer clic sobre el nombre de usuario o sobre el ícono en la columna **Editar** .
7. Modificar los campos necesarios sin olvidar de llenar los campos obligatorios y hacer clic en **Guardar**.

Figura B.8 – Editando un usuario

8. Posibles errores:
    a. No se han completado los campos obligatorios.
    b. Otros errores. En caso de encontrar otros errores, contactar al administrador.
9. Si todo se ha procesado correctamente, el sistema mostrará la pantalla de la figura B.12.

- **ELIMINAR USUARIOS**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Seguridad**.
4. Hacer clic en la opción **Editar Usuarios**.
5. Se mostrará una pantalla con todos los usuarios del sistema.
6. Escoger el usuario a eliminar y hacer clic en el icono Eliminar de la barra de navegación inferior.

Figura B.9 – Eliminación de usuarios

7. El sistema refrescará la pantalla inmediatamente.

- **AGREGAR BASES DE DATOS**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Sistema**.

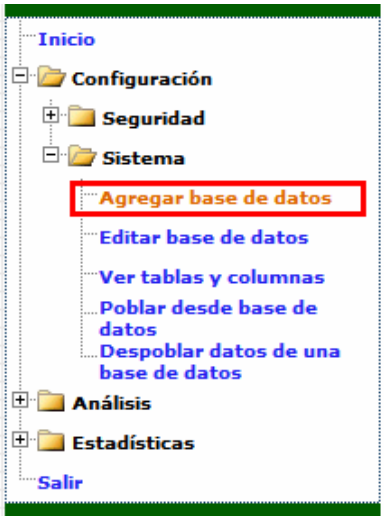


Figura B.10 – Opciones del submenú Sistema

4. Hacer clic en la opción **Agregar Base de Datos**.
5. La primera pantalla le dará bienvenida, hacer clic en **Continuar**.
6. Llenar los datos del formulario. Aquellos datos marcados con asterisco (*) son obligatorios.
7. Si todo se procesa correctamente, el sistema volverá a la página de opciones del submenú Sistema.

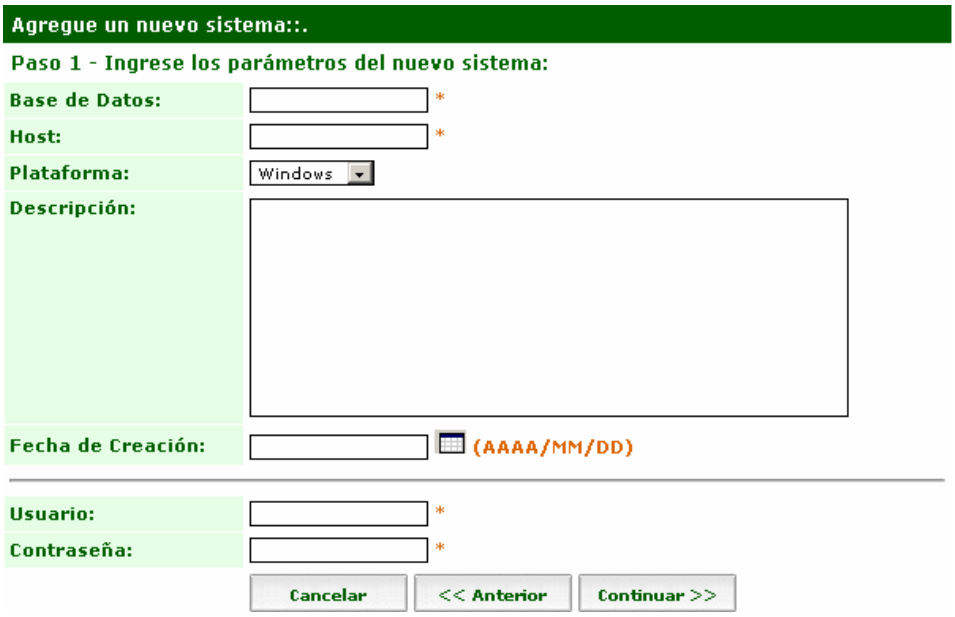


Figura B.11 – Agregar base de datos

- **EDITAR BASES DE DATOS**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Sistema**.
4. Hacer clic en la opción **Editar Base de Datos**.
5. El sistema mostrará la pantalla de todas las bases de datos agregadas.

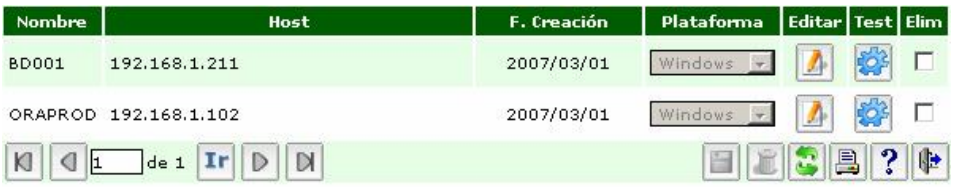


Figura B.12 – Edición de base de datos

6. Para editar una base de datos, hacer clic en el icono de la columna **Editar** .
7. Modificar los campos necesarios sin olvidar de llenar los campos obligatorios y hacer clic en **Guardar**.

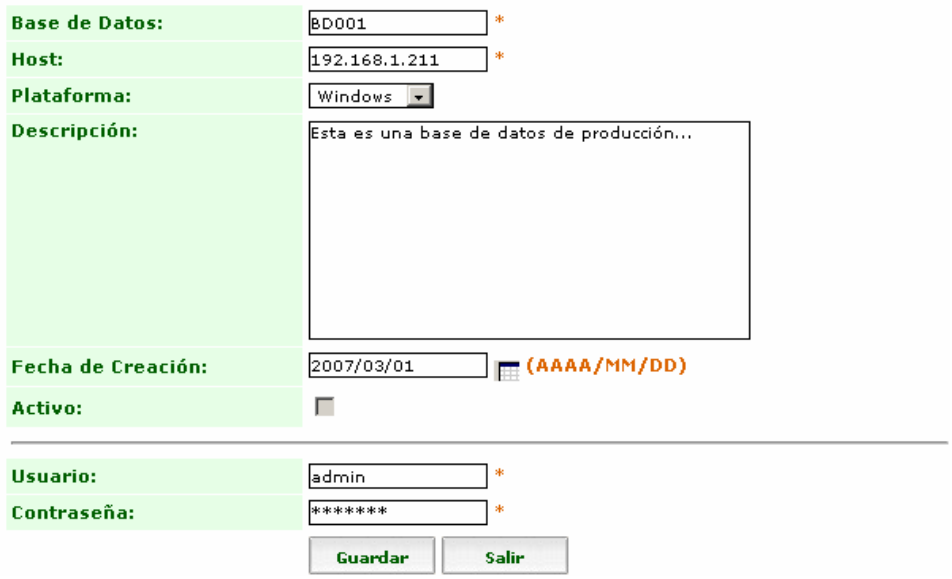


Figura B.13 – Editando una base de datos

8. Posibles errores:
    a. No se han completado los campos obligatorios.
    b. Otros errores. En caso de encontrar otros errores, contactar al administrador.
9. Si todo se ha procesado correctamente, el sistema mostrará la pantalla de la figura B.12.

- **ELIMINAR BASES DE DATOS**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.

3. Escoger el submenú **Sistema**.
4. Hacer clic en la opción **Editar Base de Datos**.
5. El sistema mostrará la pantalla de todas las bases de datos agregadas.
6. Escoger la base de datos a eliminar y hacer clic en el icono Eliminar  de la barra de navegación inferior.

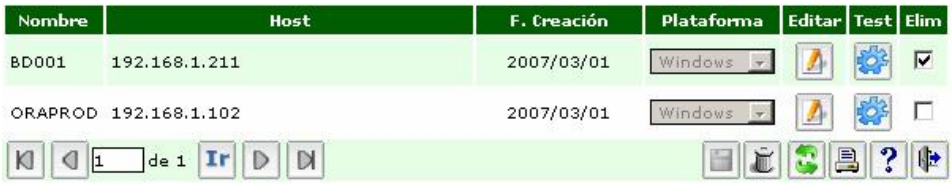


Figura B.14 – Eliminación de base de datos

7. El sistema refrescará la pantalla inmediatamente.

- **POBLAR DATOS DE UNA BASE DE DATOS AL SISTEMA**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Sistema**.
4. Seleccionar la opción **Poblar Datos de una Base de Datos al Sistema**. Previamente se debe haber agregado la base de datos.
5. El sistema mostrará una pantalla en la cual aparecerán los errores, en caso de haberlos, que se generen durante el procesamiento.

6. Esta función trae todas las tablas y su estructura al sistema para comenzar con el monitoreo.

- **ELIMINAR DATOS DE UNA BASE DE DATOS DEL SISTEMA**

1. Ingresar al módulo de administración con credenciales administrativas.
2. Ir al menú **Configuración**.
3. Escoger el submenú **Sistema**.
4. Seleccionar la opción **Eliminar Datos de una Base de Datos del Sistema**.
5. El sistema mostrará una pantalla en la cual aparecerán los errores, en caso de haberlos, que se generen durante el procesamiento.
6. Esta función elimina todas las tablas y su estructura del sistema de la base de datos seleccionada.

BIBLIOGRAFIA Y REFERENCIAS

[1]. Universidad Nacional de Colombia. *Los DBMS (Sistemas Administradores de Bases de Datos)*, [en línea]. Medellín, Colombia: Tabares, Martha Silvia. Recuperado el 18 de Julio de 2006, de http://www.unalmed.edu.co/~mstabare/Dbms.htm

[2]. Kennesaw State University. *Database History*, [en línea]. Kennesaw, EEUU: Guimara, M. Recuperado el 20 de Julio de 2006, de http://science.kennesaw.edu/~mguimara/4490/oo.ppt

[3]. University of Houston Computer Science Department. *Introduction to Database Systems I*, [en línea]. Houston, EEUU: Rusinkiewicz, Marek. Recuperado el 14 de Agosto de 2006, de http://www2.cs.uh.edu/~marek/notes/lecture1/sld017.htm

[4]. University of Waterloo Department of Computer Science. *Architecture of PostgreSQL*, [en línea]. Ontario, Canadá: Dong, Xinyi. Zou, Lijie. Lin, Yuan. Recuperado el 18 de Agosto de 2006, de http://swag.uwaterloo.ca/~lzou/cs798arch/as2/a2.htm

[5]. Universitat Jaume I. *Historia de los Sistemas de Bases de Datos*, [en línea]. Castelló, España: Marquéz, María Mercedes. Recuperado el 10 de Agosto de 2006, de http://www3.uji.es/~mmarques/f47/apun/node6.html

[6]. Internacional Business Machine (IBM). *Database Performance Tuning on AIX. Capítulo 12*. Darmawan, Budi. Groenwald, Gary. Irving, Allan. Soares Monteiro, Sergio Enrique. M. Snedeker, Keirnan. Enero del 2003.

[7]. Thomas J. Cook. (2005, 21 de Mayo). *A Database Management System Design Philosophy*

[8]. Observatorio Tecnológico. *Centro Nacional de Información y Comunicación Educativa*, [en línea]. Recuperado el 05 de Mayo del 2006 de http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=267

[9]. Universidad de Santiago de Chile. *Programación Heurística*, [en línea]. Santiago, Chile. Recuperado el 18 de Agosto de 2006, de http://www.optimos.usach.cl/PH.htm

[10]. Universidad de Valencia. *GROC: Grup de Recerca en Optimització Combinatòria, Presentación de Líneas de Investigación*, [en línea]. Valencia, España. Recuperado el 30 de Agosto de 2006, de http://centros.uv.es/web/departamentos/D130/valenciano/investigacion/lineas.xml?id=GRPV17

[11]. Chang Camacho, Violeta (2004, 27 de Octubre). *Método de Dos Fases para la Asignación de Datos en una Base de Datos Distribuida (PARTE 1/3)*, [en línea]. Trujillo, Perú: Universidad Nacional de Trujillo. Recuperado el 27 de Agosto de 2006, de http://www.informatizate.net/articulos/metodo_de_dos_fases_para_la_asignacion_de_datos_en_una_base_de_datos_distribuida_parte_01_27102004.html

[12]. Enciclpedia Wikipedia en Español. *Minería de Datos*, [en línea]. Recuperado el 10 de Septiembre de 2006, de http://es.wikipedia.org/wiki/Miner%C3%ADa_de_datos

[13]. Universidad Interamericana de Puerto Rico (1998, 9 de Septiembre). Sistema Numérico Binario, [en línea]. Álvarez, Irma.. Recuperado el 8 de Octubre de 2006, de http://coqui.lce.org:1600/ialvarez/SISBIN.HTM

[14]. World Wide Web Consortium (1997, 1 de Octubre). *SHA1 Secure Hash Algorithm - Version 1.0*, [en línea]. DesAutels, Philip A. Recuperado el 15 de Septiembre de 2006, de http://www.w3.org/PICS/DSig/SHA1_1_0.html

[15]. Oracle Corporation (2006, 5 de Octubre). *Presentación Digital: Java Server Pages*, [en línea]. Mills, Duncan. Recuperado el 10 de Octubre de 2006, de www.groundside.com/tech/docs/JSF-Overview.pdf

[16]. Internacional Business Machines (IBM). *JSF for nonbelievers: The JSF application lifecycle*, [en línea]. Hightower, Richard. Recuperado el 10 de Octubre de 2006, de http://www-128.ibm.com/developerworks/java/library/j-jsf2/