Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Plugin para el particionado de tablas en bases de datos PostgreSQL mediante la Herramienta de Administración de Bases de Datos HABD.

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

Autores:

Yehimy Figueredo Falcón

Ramiro Rodríguez Pajares

Tutores:

Ing. Glennis Tamayo Morales

Ing. Anyer Gámez Guedes

La Habana, Junio de 2012 "Año 54 de la Revolución"

"La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos."

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores: _____ _____

Yehimy Figueredo Falcón. Ramiro Rodríguez Pajares.

Tutores: _____ _____

Ing. Glennis Tamayo Morales Ing. Anyer Gámez Guedes

DATOS DE CONTACTO

Tutora: Ing. Glennis Tamayo Morales

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: -

Categoría Científica: -

Años de experiencia en el tema: 1

Años de graduado: 2

Tutor: Ing. Anyer Gámes Guedes

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: -

Categoría Científica: -

Años de experiencia en el tema: -

Años de graduado: 1

AGRADECIMIENTOS

A mis padres por hacer realidad mi sueño, por siempre apoyarme en todas mis decisiones. Por demostrarme que si se puede, por darme tanto amor y cariño, por depositar en mi toda su confianza, por su sacrificio, por su dedicación y comprensión, por la educación y los principios que inculcaron en mi.

A mi hermana Yahima por estar siempre presente, quererme y cuidar de mí.

A mi familia por todo el amor y el apoyo que siempre he recibido de ellos.

A los profesores por educarme y contribuir a en mi formación como profesional, en especial a Nairys y Heidy por apoyarme y creer siempre en mí. A mis tutores Anyer y Glennis por todo el tiempo que dedicaron en la realización de este trabajo.

A Elizabeth que también puso su granito de arena en la realización de este trabajo. Muchas gracias.

A Yanelys y Laura, por ser las mejores amigas del mundo, por siempre estar ahí cuando lo necesitaba, por creer en mí y por su apoyo incondicional.

A mi dúo de tesis por todas las horas dedicadas en la realización de este trabajo, a él le agradezco enormemente que haya compartido este momento tan especial conmigo.

A todos mis amigos por estar conmigo en los buenos y malos momentos: Eddy, Daile, Arleny, Zaily, Yenny, Argelis, Yailenis, Dallana e Ivet.

Agradezco a todos los que de una forma u otra contribuyeron y ayudaron en la realización de este trabajo. A todos muchísimas gracias.

Yehimy Figueredo Falcón

AGRADECIMIENTOS

Agradezco a mi madre en primer lugar, ya que todo se lo debo a ella y que bien pondría su nombre al lado del mío en el título o en los reconocimientos que la investigación pudiera tener.

Agradezco a todos los que en mayor o menor medida han contribuido con que este trabajo se haya culminado satisfactoriamente; que después de tantos años de esfuerzo en esta carrera me haga merecedor del título.

Agradecimientos especiales a los compañeros tutores que han sabido guiar este trabajo para que se desarrollara con una gran calidad.

A mi compañera de tesis por pasar juntos por muchos inconvenientes y malas noches.

A mis amigos en general por aportar alguna idea que siempre se consideraron importantes.

Ramiro Rodríguez Pajares

DEDICATORIA

Dedico este trabajo a las personas más importantes en mi vida:

A mi mamá que es la mejor madre de este mundo, por su ternura, por su amor, por su dedicación, por ser tan compresiva y tener tanta paciencia con su niñita malcriada que la ama con la vida, por siempre aconsejarme y guiarme por el buen camino, a ti mamita te dedico mi tesis con todo mi amor para que te sientas orgullosa de mi y veas que toda la confianza que depositaste en mi no fue en vano.

A mi papito lindo que es el hombre más importante en mi vida, por enseñarme a luchar por lo que quiero en la vida, por los principios y la educación que siempre me ha inculcado, a ti papito te dedico mi tesis, por que se que me quieres tanto como yo a ti, porque sé que siempre estarás ahí para mí como ha sido hasta ahora.

Yehimy Figueredo Falcón

DEDICATORIA

Dedico este trabajo a todas las personas que han hecho posible que llegara hasta aquí, especialmente a mi madre que siempre se ha sacrificado mucho para que yo siguiera adelante a lo largo de toda la carrera y durante toda la vida. Además a mis hermanas por siempre estar ahí cuando las he necesitado, a mi tía que siempre que la he necesito tampoco me ha dado la espalda, y a mi padre que aun estando lejos siempre ha estado ahí para apoyarme. También al resto de la familia que nunca han dejado de darme ánimo y eso también es muy importante.

Ramiro Rodríguez Pajares

Resumen

En el Centro de Tecnologías de Gestión de Datos (DATEC), perteneciente a la Universidad de Ciencias Informáticas (UCI), se encuentra el departamento PostgreSQL en el que se desarrolla la herramienta de administración de bases de datos HABD. Actualmente esta herramienta carece de funcionalidades que permitan el trabajo con el gestor PostgreSQL, una de ellas es el particionado de tablas. Para realizar esta técnica en el departamento existe la herramienta PaDit, pero esta no puede ser integrada a HABD porque su arquitectura no lo permite, además de no explotar todas las variantes de particionado soportadas por el gestor PostgreSQL.

El presente trabajo tiene como objetivo desarrollar un plugin que permita particionar las tablas de las bases de datos aplicando las dos variantes soportadas por el gestor PostgreSQL, por lista y por rango.

Para ello se realizó una investigación en busca de herramientas que facilitaran el desarrollo de la aplicación, encontrándose la herramienta PaDit utilizada como base para la presente investigación. Además, se identificaron las funcionalidades y se realizó el diseño e implementación del plugin. Esta solución se encuentra integrada a la herramienta HABD permitiendo a los administradores aplicar esta técnica que resulta muy eficiente en bases de datos que almacenan grandes volúmenes de información.

Palabras Claves: PostgreSQL, HABD, plugin, PaDit.

Índice de contenidos

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad, con el desarrollo tecnológico alcanzado, la información se ha convertido en el recurso intangible de mayor valor para cualquier empresa u organismo, imprescindible en la toma de decisiones que involucren el crecimiento económico. En el mundo empresarial las pérdidas financieras provocadas por la desaparición de archivos u otros datos de importancia suelen ser la principal preocupación, de ahí la necesidad de protección de este recurso. Hoy, las bases de datos son las herramientas más usadas con este fin, dada la eficiencia demostrada en la protección, almacenamiento y gestión de la información.

Aparejado al avance de las tecnologías de bases de datos han sido desarrollados sistemas gestores de bases de datos (SGBD), los cuales han tenido un extraordinario auge a raíz de la informatización empresarial. Los SGBD permiten un mayor manejo y control de la información, por lo que para garantizar la integridad y veracidad de la misma deben ser lo más seguros y confiables posibles.

El desarrollo de SGBD es liderado por compañías como ORACLE, SQL Server, INFORMIX y DB2 de IBM (International Business Machines), las que ofrecen productos destinados a la administración de grandes volúmenes de información como el DreamCoder y EMS SQL Management Studio; sin embargo, estas son compañías privativas lo que significa que para acceder a sus productos es necesario comprar a la empresa una licencia específica que les permita el uso de los mismos. El desarrollo de SGBD gratuitos ha contribuido a mitigar esta situación. En ocasiones los SGBD gratuitos superan las características y funcionalidades de sus competidores comerciales, entre ellos PostgreSQL el cual sigue un activo proceso de desarrollo a nivel mundial gracias a una gran comunidad de desarrolladores y contribuidores de código abierto.

PostgreSQL está ampliamente considerado como el Sistema de Gestión de Bases de Datos de código abierto (gratuito y con código fuente disponible) más avanzado del mundo. Posee las características de los más potentes sistemas comerciales como: estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Su avanzada funcionalidad le permite hacer consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia y arreglos. Es altamente extensible; soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario. Soporta la integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. (1)

Este SGBD cuenta con herramientas para la administración de bases de datos como pgAccess y

pgAdmin3, siendo esta última una de las más empleadas, por ser una herramienta de código abierto que cuenta con una interfaz intuitiva para el diseño y administración de las bases de datos, diseñada para ejecutarse en la mayoría de los Sistemas Operativos.

En Cuba existe una industria de software que permite ubicar al país en el plano internacional. Demuestra que aun siendo un país bloqueado, con la capacidad de sus profesionales puede estar a nivel de cualquier país primermundista en cualquier esfera, incluso en el plano tecnológico. Existen instituciones que se dedican a fomentar la lucha del país por alcanzar una soberanía tecnológica, como la Universidad Central de Las Villas (UCVL) y la Universidad de Ciencias Informáticas(UCI), mediante el desarrollo de tecnologías basadas en código abierto que permiten el perfeccionamiento de la informatización y el aumento de la productividad económica del país.

La UCI es una de las universidades del país donde se contribuye al desarrollo del software, vinculando la docencia con la producción. Considerada una de las principales industrias del software en el país, dicha universidad ha devenido en una importante fuente de ingresos.

Adscrito al mencionado centro de altos estudios se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC), el cual está compuesto por varias líneas de producción dentro de las que se encuentra PostgreSQL. Esta última tiene entre sus objetivos el desarrollo tecnológico de bases de datos libres tomando de base PostgreSQL, proporcionándole soluciones integrales y consultorías relacionadas con la explotación del gestor.

En el departamento de PostgreSQL se presentó el trabajo de diploma ''Exploración y diseño de la Herramienta de Administración de Bases de Datos para PostgreSQL (HABD) ''. Esta surge a partir de los inconvenientes encontrados en herramientas como PgAdmin3 que no permiten el desarrollo de determinadas funcionalidades y obliga al usuario a apoyarse en aplicaciones externas para tener un control más completo de sus bases de datos. En otras como el pgAccess se encuentra que sus deficiencias mayores radican en sus interfaces visuales y en la interacción con el usuario. HABD evita tener que contar con aplicaciones distintas que puedan resolver los problemas de administración de las bases de datos debido a que cuenta con una arquitectura basada en plugins, la misma posibilitará realizar todas las funcionalidades permitidas por el gestor de manera sencilla y eficiente.

Hasta el momento solo se ha desarrollado un front-end, por lo que no cuenta con las funcionalidades básicas de una herramienta de administración entre las que pueden mencionarse: manipulación de objetos, edición de consultas y funciones, así como particionado de tablas. En bases de datos que almacenan información de crecimiento continuo el particionado de tablas es una funcionalidad muy útil para disminuir el tiempo de respuesta de las consultas.

En PostgreSQL para realizar el particionado de tablas es necesario seguir un conjunto de pasos manuales que resultan engorrosos para trabajar de manera sistemática. Sumado a esto, los usuarios para realizar esta operación deben poseer conocimientos avanzados en programación sql y sobre el gestor. Con el objetivo de mejorar esta técnica en el departamento PostgreSQL se desarrolló una herramienta llamada PaDiT, que permite el particionado de tablas en bases de datos PostgreSQL de forma visual y más optimizada. Esta herramienta a pesar de que existen dos variantes de particionado soportadas por PostgreSQL, por lista y por rango, solo permite particionar las tablas por lista, PaDit no puede ser integrada con la herramienta HABD pues su arquitectura no se lo permite por estar desarrollada en el lenguaje de programación java y HABD en C++. PaDiT,

Dada la situación antes expuesta surge el siguiente **Problema de la Investigación**: ¿Cómo particionar tablas de bases de datos PostgreSQL en la herramienta de administración HABD?

Se propone como **objeto de estudio:** Proceso de Particionado de tablas en bases de datos; enmarcado en el **campo de acción:** Particionado de tablas en bases de datos PostgreSQL**.**

El **objetivo general** del trabajo: Desarrollar un plugin para la herramienta de administración HABD que permita realizar el particionado de tablas en bases de datos PostgreSQL.

En correspondencia con el Objetivo general se definen los siguientes **objetivos específicos:**

- Analizar el proceso de particionado de tablas en bases de datos PostgreSQL.
- Analizar las funcionalidades del plugin para el particionado de tablas en bases de datos PostgreSQL
- Diseñar el plugin para el particionado de tablas en bases de datos PostgreSQL.
- Implementar el plugin para el particionado de tablas en bases de datos PostgreSQL.
- Probar el plugin para el particionado de tablas en bases de datos PostgreSQL.

Para darle cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación:**

- Caracterización del proceso de particionado de tablas en bases de datos PostgreSQL.
- Caracterización de herramientas que realizan el particionado de tablas en bases de datos PostgreSQL.
- Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del plugin para el particionado de tablas en base de datos.

- Identificación de las funcionalidades del plugin para el particionado de tablas en base de datos.
- Obtención de las tarjetas CRC y el diagrama de clases.
- Implementación de las funcionalidades identificadas.
- Diseño de los casos de pruebas para la realización de pruebas al plugin de particionado de tablas.
- Aplicación de los casos de prueba para validar que el plugin desarrollado cumpla con las funcionalidades identificadas.

Capítulo 1: Fundamento Teórico

Introducción

En el capítulo se realiza un análisis sobre el marco teórico del proyecto, mediante conceptos fundamentales que permiten un mejor entendimiento de la investigación. Se hace además un profundo estudio del estado del arte de las herramientas y metodologías que se utilizarán, así como todo lo relacionado con la técnica del particionado de tablas, para de esta forma dejar claras las ventajas de su aplicación en base de datos.

1.1 Sistema Gestor de Bases de Datos

´´Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.´´ (2)

Con el apoyo del concepto antes expuesto se puede definir SGBD como un conjunto de programas que facilitan el trabajo con las bases de datos. Es el intermediario entre la base de datos y el usuario, permite a estos últimos manipular los datos de una forma organizada y según sus privilegios. Su objetivo principal es manejar conjuntos de datos de forma clara, sencilla y ordenada. Los mismos permiten construir, definir y manipular base de datos.

PostgreSQL v9.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (3)

Se ejecuta en casi todos los principales sistemas operativos (Ej.: Linux, Unix, BSDs, Mac OS, Beos, Windows). Su documentación está bien organizada, pública y libre. Presenta un soporte nativo para los lenguajes más populares del medio (Ej.: PHP, C, C++, Perl, Python,). Proporciona soporte a las características de las bases de datos profesionales (Ej.: disparadores, procedimientos almacenados,

funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas). Es altamente adaptable a las necesidades del cliente. La atomicidad, consistencia, aislamiento y durabilidad son algunas de sus características.

Ventajas

❖ *Instalación ilimitada: Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.*

❖ *Soporte: tiene una importante comunidad de profesionales y entusiastas de PostgreSQL de los que su compañía puede obtener beneficios.*

❖ *Ahorros considerables en costos de operación: PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.*

❖ *Estabilidad y Confiabilidad Legendarias: Es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.*

❖ *Extensible: El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales.*

❖ *Multiplataforma: PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable) y ahora en versión nativa para Windows.*

❖ *Diseñado para ambientes de alto volumen: PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC (Acceso concurrente multiversión) para conseguir una mejor respuesta en ambientes de grandes volúmenes.*

❖ *Herramientas gráficas de diseño y administración de BD: Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect).* (4)

1.2 Herramientas de Administración para PostgreSQL HABD

Las herramientas de administración son aplicaciones que facilitan la interacción del usuario con el gestor de base de datos.

Herramienta de Administración de base datos HABD

Es una herramienta desarrollada por el centro DATEC con el objetivo de lograr una mejor interacción entre los usuarios y el gestor PostgreSQL buscando eliminar los inconvenientes presentados en algunas herramientas libres, como las deficiencias en las interfaces visuales y la carencia de servicios que faciliten el trabajo de las personas que interactúan con el gestor. Provee una interfaz amigable y una arquitectura flexible que permite, de manera sencilla, incorporar nuevos servicios, lo que favorece el incremento de funcionalidades.

Este sistema basado en componentes plugins será el encargado de permitir a los usuarios el trabajo con PostgreSQL. En la actualidad solo se cuenta con un Front-End, el cual administrará todos los plugins que se desarrollen para conformar la herramienta. Su interfaz visual será capaz de personalizar el entorno de trabajo tanto como el usuario desee, permitiendo configurar las preferencias de las ventanas de acuerdo a sus necesidades.

1.3 Particionado de Tablas en Bases de datos

El particionado de tablas consiste en la segmentación de los datos, y debe ser aplicado cuando existen tablas con gran volumen de información. Con la técnica de particionado se reduce considerablemente el tamaño de los índices, lo que mejora el rendimiento de las consultas y los procesos de carga y actualización masiva de datos. Permite realizar búsquedas solo en aquellas particiones de una tabla en la que se conoce que se encontrará la información. Esto sucede de forma transparente al usuario cuando se consulta la tabla, al no tener que especificar en qué particiones buscar; simplemente al consultar una tabla esta se ejecutará en las particiones adecuadas.

1.3.1 Tipos de Particionado

El particionado de tablas se realiza de acuerdo a reglas definidas por el usuario, estas reciben el nombre de funciones de particionado, de las cuales existen varios tipos. Entre las más usadas por los gestores están: el particionado por rangos, listas de valores y funciones hash.

❖ *Particionado por rango: la clave de particionado viene determinada por un rango de valores, que determina la partición donde se almacenará un valor.*

❖ *Particionado por funciones hash: la clave de particionado es una función hash, aplicada sobre una columna, que tiene como objetivo realizar una distribución equitativa de los registros sobre las diferentes particiones. Es útil para particionar tablas donde no hay un criterio de particionado claro, pero en la que se quiere mejorar el rendimiento.*

❖ *Particionado por lista: la clave de particionado es una lista de valores, que determina cada una*

de las particiones. (5)

Ventajas

El particionado es una técnica muy útil en el trabajo con base de datos donde la información crece considerablemente lo cual permite reducir el tiempo de respuesta al consultar las bases de datos, esta es una de las tantas ventajas que se muestran a continuación:

- ❖ Se obtienen índices más pequeños.
- ❖ Permite realizar backups más rápidos.
- ❖ Los procesos de borrado masivos, pueden ser muy sencillos al eliminar particiones en vez de registros.
- ❖ Se logra un desempeño mejorado para la obtención de registros y para la actualización. Los datos pueden ser migrados o respaldados en medios económicos y pequeños como DVDs o discos duros extraíbles.
- ❖ Se consigue gran optimización a la hora de acceder a los datos almacenados ya que se pueden unificar datos comunes en las mismas particiones.
- ❖ Aporta mucha flexibilidad si se piensa en escalar en la base de datos, al permitir guardar cada partición en diferentes direcciones físicas.
- ❖ Acelera las transacciones o consultas.
- ❖ Mejora las prestaciones del sistema.
- ❖ Perfecciona el rendimiento cuando el servidor está configurado para realizar procesamiento paralelo de consultas, ya que puede haber un proceso de trabajo por partición en una búsqueda basada en particiones.
- ❖ Permite hacer inserciones múltiples a una misma tabla.
- ❖ Incrementa el rendimiento al posibilitar que las operaciones de lectura/escritura se distribuyan sobre diferentes dispositivos de datos.

1.3.2 Particionado de tablas en PostgreSQL

El tipo de particionado soportado por PostgreSQL, es el particionado mediante herencia de tablas. Cada partición puede ser creada como una tabla hija de una única tabla padre, la cual normalmente debe ser vacía, que representará a todo el conjunto de datos. En PostgreSQL, con el particionado de

tablas se reduce la cantidad de datos a recorrer en cada consulta SQL, lo cual mejora el rendimiento considerablemente.

Existen dos tipos de particionado en PostgreSQL: El particionado por rangos, donde la tabla es particionada mediante rangos definidos en base a la columna de llave primaria o cualquier columna que no corresponda a los rangos de valores asignados a diferentes tablas hijas (usado mayormente en campos de fecha o de valores numéricos) y particionado por lista, en el que la tabla es particionada listando los valores de cada una de las llaves en cada partición.

El particionado de tablas resulta una funcionalidad muy útil en aquellas bases de datos donde los registros crecen considerablemente. Es una forma de organizar los datos clasificándolos según criterios de agrupación, de manera que cada transacción realizada en una tabla padre se guarda automáticamente a un menor grupo de datos que se encuentran en las tablas hijas, esto mejora ostensiblemente el tiempo de respuesta de las consultas y logra una mejor satisfacción del usuario.

Pasos para particionar en PostgreSQL

Para aplicar correctamente esta técnica se deben seguir seis pasos, para la ejecución de los cuales es necesario contar con un conocimiento avanzado en la programación SQL. A continuación se muestran los pasos a seguir:

1. Crear la tabla "maestra", de la cual todas las tablas hijas heredarán. Esta tabla no contendrá datos, no debe de tener ningún tipo de restricción (check), no debe de tener índice ni nada por el estilo.

2. Crear todas las tablas hijas heredando de la tabla maestra. Por lo regular estas tablas heredarán todos los campos de la tabla padre y no es necesario que el usuario los cree el mismo.

3. Agregar a todas las tablas hijas las restricciones correspondientes sobre los datos que albergarán. Algunos ejemplos de esto serían: CHECK (país IN ('México', 'Argentina')), CHECK (id_cliente BETWEEN 100 AND 200)... CHECK (id_cliente BETWEEN 5000 AND 5200), en estos dos últimos ejemplos para evitar traslapes entre los ids de clientes.

4. Para cada partición, crear un índice para la(s) columna(s) llave preferentemente o los que se estimen convenientes. El índice para la llave primaria no es estrictamente necesario, pero en la mayoría de los casos ayuda mucho. Si se necesita una llave única o llave primaria, se deberá crear para cada tabla hija.

5. Definir una regla (RULE) o función disparadora (TRIGGER) para redirigir las modificaciones de la tabla padre (maestra) a la partición apropiada.

6. *Verificar que esté habilitado el parámetro constraint_exclusion (SET constraint_exclusion = on;) en el archivo de configuración postgresql.conf para que las consultas sean optimizadas para el particionado.*(6)

1.4 Herramientas existentes para el particionado

Existen sistemas gestores de base datos que permiten el particionado de tablas como PostgreSQL, MySQL y ORACLE, estos últimos realizan esta técnica mediante consultas SQL, las cuales ya están implementadas.

Durante la investigación y la bibliografía consultada solo se encontró una herramienta nombrada PaDiT, la cual permite realizar el particionado de tablas para base de datos PostgreSQL.

Herramienta PaDiT

PaDiT es una aplicación de escritorio que consiente en realizar el particionado de tablas por lista. Esta consta de varias funcionalidades como la administración de conexiones con el servidor de base de datos y la partición de tablas, que es el objetivo principal de la herramienta. Permite particionar por el método lista una tabla seleccionada de un determinado esquema, especificando sus campos, así como, admitir al usuario nombrar las particiones creadas. Esta herramienta no puede ser integrada al HABD porque su arquitectura no lo admite por estar desarrollada en Java, aunque sirve como base de estudio para desarrollar una herramienta que le brinde al usuario la posibilidad de realizar el particionado por la variante que desee, sea por lista o por rango, y de esta forma perfeccionar el trabajo con las base de datos.

1.5 Metodología de desarrollo de software

No es más que un conjunto de pasos y procedimientos a seguir para desarrollar un software. Es la encargada de estructurar, planificar y controlar el proceso de desarrollo para lograr una mejor calidad de las aplicaciones. Facilita el trabajo haciéndolo más entendible mediante la división del proyecto en varias etapas.

Existen disímiles metodologías de desarrollo ágil, la mayoría minimizan el riesgo del desarrollo de software en corto tiempo. Estas enfatizan la comunicación cara a cara en vez de la documentación. Entre las metodologías de desarrollo ágil se encuentra Xp (eXtreme Programing), creada por Kent Beck fue dada a conocer seis años atrás, es una metodología basada en la simplicidad, la comunicación y el reciclado continuo de código.

XP es una metodología que trata de proporcionar al cliente el software que necesita cuando lo necesita. Debe responder las necesidades de forma rápida, incluso cuando los cambios sean al final

del ciclo de programación. Potencia el trabajo en grupo, los jefes de proyecto, clientes y desarrolladores son parte del equipo, y están involucrados en el desarrollo del software. Define cuatro variables para proyectos de software: coste, tiempo, calidad y ámbito. (7)

En el presente trabajo se decide utilizar la metodología XP dado el ajuste de sus características con las necesidades del proyecto. Su simplicidad consiste en desarrollar el sistema que realmente se necesita, implica resolver en cada momento solo las necesidades actuales. Pone en comunicación directa y continua a clientes y desarrolladores, integrando el cliente al equipo lo, que facilita la definición de prioridades y el esclarecimiento de dudas. Requiere un pequeño grupo de programadores (entre 2 – 15 personas). Confiere gran adaptabilidad al producto al asimilar proyectos cuyos requisitos cambian a menudo.

1.6 Herramientas y tecnologías a utilizar

En este epígrafe se explican las herramientas y tecnologías que se utilizarán para el desarrollo del plugin, las ventajas de las mismas y el porqué de su selección.

1.6.1 Herramientas CASE (por sus siglas en inglés Computer Aided Software Engineering)

Son aplicaciones utilizadas para aumentar la productividad en el desarrollo de software reduciendo el tiempo y coste de los mismos. Estas herramientas facilitan asistencia a los analistas, ingenieros de software y desarrolladores durante todo el ciclo de vida del software.

Visual Paradigm v6.4

"Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida y mejor construcción de aplicaciones de calidad así como a la disminución del coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML." (8)

Además, brinda ayuda a los programadores garantizando la generación de código para algunos lenguajes de programación como C++ en el que está desarrollada la herramienta propuesta. El uso del Visual Paradigm ahorra tiempo debido a que es una herramienta dominada por el analista del proyecto, lo que influye en la realización de un diseño de mejor calidad.

Las características antes expuestas son el soporte fundamental de la decisión de adoptar Visual Paradigm como herramienta de modelado para el plugin.

1.6.2 Lenguaje de modelado UML

"UML (Lenguaje Unificado de Modelado) es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar y mantener y controlar la información sobre los sistemas a construir. Cuenta con una serie de diagramas prácticos y útiles, divididos en categorías (Estructura, Comportamiento e Interacción), que son capaces de documentar todo el proceso de modelado de cualquier tipo de sistema informático. No es un método o proceso ya que es independiente de los métodos de análisis y diseño. El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. Es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes metodólogos en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson. El lenguaje ha ganado un significante soporte de la industria de varias organizaciones vía el consorcio de socios de UML y ha sido presentado al Object Management Group (OMG) y aprobado por éste como un estándar (noviembre 17 de 1997)". (9)

Se utiliza extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para mostrar y comunicar dicho plan a todo un equipo de desarrolladores. UML tiene como objetivo entregar un material de apoyo que le permita al lector poder definir diagramas propios como también poder entender el modelamiento de diagramas ya existentes, el mismo ofrece un estándar para representar un plano, o sea un modelo del sistema. Para ello contiene aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Para poder trabajar con UML no necesariamente se tiene que conocer un lenguaje de programación, debido a que este fue diseñado para modelar cualquier tipo de proyecto.

1.6.3 Entorno de desarrollo

Un entorno de desarrollo no es más que un programa compuesto por un conjunto de herramientas de programación, utilizadas para programar en varios lenguajes.

Un IDE (entorno de desarrollo integrado) es un entorno de programación que ha sido unido como un programa de aplicación. Estructurado por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

QtCreator v4.7

Es un IDE creado para el desarrollo de aplicaciones en C++ de manera sencilla y rápida. Es una herramienta multiplataforma, basada en la biblioteca QT, consta de un depurador visual, completado automático y un editor avanzado para C++. Dotado de herramientas para la administración y construcción de proyectos, diseñador de formularios GUI (Graphical User Interface) integrado, ayuda sensible al contexto integrada, y soporte para refactorización de código.

Está completamente integrado a todas las documentaciones Qt y a los ejemplos, a través de la Ayuda Qt plugin. También, está integrado a Qt Designer para facilitar al desarrollador el diseño de la interfaz de usuario.

Una de las principales ventajas de QtCreator es que permite que un equipo de desarrolladores comparta un proyecto a través de diferentes plataformas de desarrollo con una herramienta común para el desarrollo y depuración.

Con su estructura de creación de proyecto permite:

- Agrupar archivos.
- Agregar pasos de generación personalizada.
- Incluir formularios y archivos de recursos.
- Especificar la configuración de las aplicaciones en ejecución.

1.7 Lenguaje de programación

´´Un lenguaje de programación no es más que un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano. Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador´´. (10)

Lenguaje C++

El lenguaje de programación C++ fue creado en los años 80 por Bjarne Stroustrup basado en el lenguaje C. Es uno de los lenguajes más potentes, permite programar a alto y bajo nivel, es orientado a objeto, tiene gran compatibilidad con C debido a que comparten gran cantidad de códigos. No es orientado a objeto puro porque fue desarrollado a partir de otro lenguaje.

*Entre sus características se encuentra el permitir la agrupación de instrucciones, presentar un conjunto completo de instrucciones de control, incluir el concepto de puntero (variable que contiene la dirección de otra variable), los argumentos de las funciones se transfieren por su valor, permite la separación de un programa en módulos que admiten compilación independiente, es un lenguaje estructurado, puede manejar actividades de bajo-nivel, implementación de apuntadores, uso extensivo de apuntadores para la memoria, arreglos, estructuras y funciones, los argumentos de las funciones se transfieren por su valor. Un punto en contra es que tiene una detección pobre de errores, lo cual en ocasiones es problemático para los principiantes. (*11)

Dada las ventajas y características enunciadas se decide utilizar el lenguaje de programación C++ en el desarrollo del presente trabajo. Otra de las razones que llevan a la decisión anterior es el hecho de que el plugin que se desarrollará se integrará a una herramienta de bases de datos HABD desarrollada en dicho lenguaje, además, de que las librerías de PostgreSQL son programadas en C++.

1.8 Framework de desarrollo

Un framework es una estructura conceptual que permite organizar y desarrollar cualquier proyecto de software, incluye soportes de programación, bibliotecas, entre otras herramientas; lo que ayuda a desarrollar y unir los diferentes componentes de un proyecto.

Son herramientas implementadas tanto con objetivos específicos como generales que reúnen un conjunto de características y funcionalidades para facilitar la implementación de las aplicaciones a las cuales va dirigida.

El término framework es usado en programación orientada a objetos para delimitar un conjunto de clases que definen un diseño abstracto para solucionar un conjunto de problemas relacionados.

Framework Qt V2.2

Es un framework para el desarrollo de aplicaciones multiplataforma. Entre sus funciones está la creación de interfaces de usuario. Funciona en todas las principales plataformas, cuenta con una excelente documentación, y tiene una arquitectura lista para plugins, ofrece soporte para lenguajes como Python mediante PyQt, Java mediante QtJambi, o C# mediante Qyoto. Es un framework muy

poderoso que ofrece una suite de aplicaciones para facilitar y agilizar las tareas de desarrollo, comparable incluso con .NET de Microsoft. Utiliza C++ de manera nativa, por lo que la convierte en la estructura idónea para el desarrollo del plugin.

Se decide, además, trabajar con Qt porque es orientado a objetos y permite la reutilización de código, lo que facilita el trabajo al programador. Dado que está escrito en lenguaje C++ responde con gran velocidad. El API (Application Programming Interface) de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL.

Conclusiones del capítulo

En la investigación realizada sobre el particionado de tablas en los sistemas gestores de base de datos y las herramientas de administración se encontró solo una herramienta que realiza el particionado de tablas para PostgreSQL, llamada PaDit, que se utilizará como base para la presente investigación.

Se realizó un análisis de las herramientas y metodologías a emplear, donde se define XP como metodología, Visual Paradigm como herramienta CASE y UML como lenguaje de modelado. El lenguaje de programación a utilizar es C++ con el frameworkQt y como IDE QtCreator.

Capítulo 2: Características del Sistema

Introducción

En este capítulo se describe la propuesta de solución al problema de investigación planteado. Se exponen las historias de usuarios (donde se reflejan las características con que contará el plugin), los diferentes artefactos que se generan en la metodología Xp, la arquitectura del sistema y los patrones de diseño a utilizar. También se muestra una breve descripción de la solución, presentando varios diagramas que ayudarán a una mejor comprensión de la misma.

2.1 Modelo de dominio

El modelo de dominio es una alternativa para la identificación de requisitos y la comprensión del contexto cuando existe poca estructuración en los procesos de negocio, y con la que se le puede mostrar al usuario, de manera visual, los principales conceptos que se manejan en el dominio del sistema, sus partes y sus relaciones. Esto permite a todos los que de alguna manera están involucrados en el proceso de desarrollo del producto, manejar un vocabulario común que posibilite el entendimiento del contexto en que se sitúa el sistema. Este modelo se realiza a través de un diagrama de clases de UML simplificado, en el cual se representan las clases conceptuales que pueden intervenir en el sistema y sus asociaciones preliminares, así como los objetos más importantes en el mismo. Estos objetos del dominio representan "cosas" que existen o los eventos que acontecen en el medio en el que se desenvuelve la aplicación. (12)

A pesar de que la metodología Xp no utiliza el modelo de dominio se decidió utilizar este para un mejor entendimiento del problema a resolver, debido a la existencia de solapamiento de roles y al hecho de no contar con un negocio bien definido (ver figura 1).

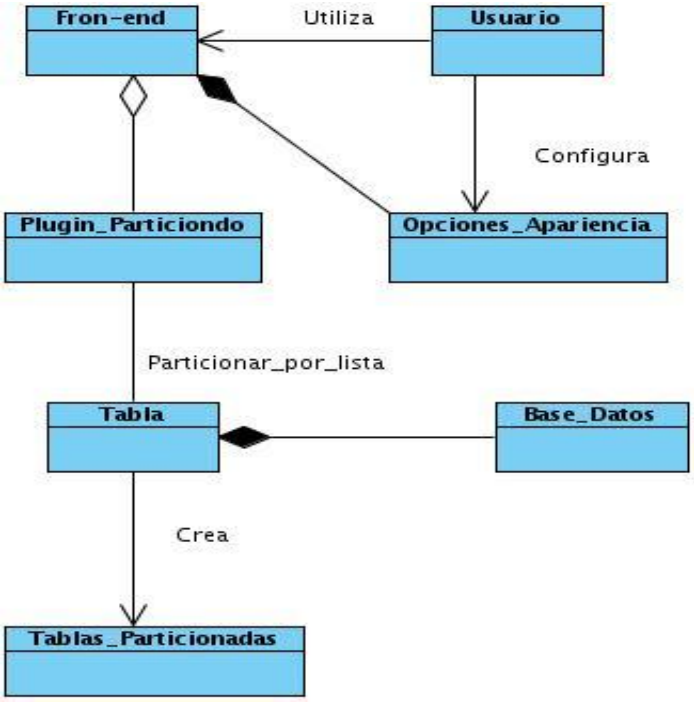


Figura 1: Modelo de dominio

❖ La **clase Frond-end** representa la interfaz de la herramienta de administración de base de datos HABD, a la cual se le agregarán varios plugin.

❖ La **clase Usuario** se encarga de utilizar el frond-end, la misma configura las opciones de apariencia.

❖ La **clase Opciones_Apariencia** es la encargada de modificar la apariencia al fron-end.

❖ La **clase Plugin_Particionado** se encarga de particionar las tablas de la base de datos.

❖ La **clase Tabla** representa las tablas, las cuales se particionarán por lista.

❖ La **clase Base_Datos** contiene las tablas, las cuales se particionarán por lista.

❖ La **clase Tablas_Particionadas** representa el resultado de realizar el particionado por lista, es decir, las tablas particionadas.

2.2 Descripción de la solución propuesta

Con el propósito de solucionar el problema de investigación planteado se propone desarrollar un plugin para ser integrado a la herramienta de administración HABD. Este plugin debe permitir: Realizar el particionado de tablas en las dos variantes que soporta el gestor PostgreSQL (el particionado por lista y el particionado por rango), que el usuario seleccione la columna por la que desea particionar (las cuales aparecerán en un componente de texto para su selección), así como la posibilidad de

seleccionar el tipo de particionado mostrado en un menú (también cuenta con varios componentes de texto que permitirán ver los datos de la columna seleccionada e introducir otros criterios de particionado, si así lo desea el usuario).

El plugin contará con un botón de avanzada donde al hacer clic se muestre otra interfaz capaz de crear particiones en otro esquema, además concederá la capacidad de especificar en que tablespace quiere que estén las particiones creadas y le permitirá forzar el particionado.

2.3 Historias de usuarios

Las Historias de Usuarios (HU) representan una breve descripción del comportamiento del sistema. Las mismas emplean terminologías del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema. Se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos; reemplazan documentos de requisitos y presiden la creación de las pruebas de aceptación. Las HU difieren de los casos de uso porque son escritas por el cliente, no por los programadores, empleando terminologías del cliente. Las historias de usuario son más "amigables" que los casos de uso formales. (13)

Durante el análisis y diseño se identificaron seis historias de usuarios (Mostrar los campos de la tabla que se desea particionar, Particionar por lista, Particionar por rango, Administrar partición, Mostrar las particiones realizadas y Generar ayuda), las cuales contienen una breve descripción de lo que el cliente desea. Estas fueron clasificadas en Muy Alta, Alta, Media y Baja; según la prioridad en el negocio y el riesgo que representa para el desarrollo del proyecto (de esta forma no se pone en riesgo toda la realización del proyecto a desarrollar). Le fueron asignados puntos estimados para su implementación, establecidos según la complejidad de la HU, teniendo como máximo tres semanas.

A continuación se muestra la HU número tres, nombrada “Particionar por rango”. El desarrollo de la misma está planificado para la iteración uno, con un tiempo estimado de tres semanas, esta HU presenta una prioridad muy alta tanto en el negocio como en el riesgo de su desarrollo, por ser una de las principales funcionalidades con que debe contar el plugin, ya que permite al usuario seleccionar el esquema, la tabla y el campo por el que desea particionar por rango, además le permite introducir el nombre que quiere que tomen las particiones creadas (ver tabla 1).

Tabla 1.Historia de usuario particionar por rango

Historia de Usuario	
Número:3	**Nombre de Historia de Usuario:** Particionar por rango
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Yehimy Figueredo Falcón	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 Semanas
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 2 Semanas
Descripción: Permite al usuario seleccionar el esquema, la tabla y el campo por el que desea particionar por rango, además le permite introducir el nombre que quiere que tomen las particiones creadas.	
Observaciones: El usuario debe escoger el esquema, la tabla y el campo por el que desea particionar.	
Prototipo de interfaces: Ver Anexo 1	

2.4 Lista de Reserva del Producto

Al iniciar un proyecto es muy difícil tener claro todas las funcionalidades o requerimientos con que debe contar el producto, por tal razón la metodología Xp tiene como una de las actividades fundamentales la confección de la Lista de Reserva del Producto (LRP). En este listado se recoge una lista priorizada de todas las funciones y cualidades que debe realizar el proyecto, esta puede crecer y modificarse a medida que se obtiene mayor conocimiento del producto y el cliente. El objetivo es asegurar que al terminar la lista, el producto cumpla con todas las expectativas del cliente referidas a los requerimientos técnicos y del negocio.

Mediante las HU se identificaron las funcionalidades que el cliente desea que tenga la aplicación. A continuación se muestra la LRP donde se encuentran todas estas funcionalidades con su prioridad y la estimación para su implementación, como también se puede ver un listado de los requisitos no funcionales con que debe contar el producto. Todas estas funcionalidades fueron estimadas por el Analista del proyecto para facilitar el entendimiento de las cualidades con que debe contar la aplicación (ver tabla 2).

Tabla 2.Lista de reserva producto

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Mostrar los campos de la tabla que se desea particionar.	1 Semana	Analista
2	Particionar por lista	2 Semanas	Analista
3	Particionar por rango	3 Semanas	Analista
Prioridad: Alta			
4	Eliminar Partición	1.5 Semanas	Analista
5	Adicionar Partición	1.5 Semanas	Analista
Prioridad: Media			
6	Mostrar las particiones realizadas.	1 Semana	Analista
Prioridad: Baja			
7	Generar ayuda	1 Semana	Analista
Requisitos no funcionales.			

1	**Apariencia o interfaz externa:** Interfaz intuitiva, amigable, organizada, con una navegabilidad flexible y de fácil comprensión.		Analista
2	**Facilidad de uso:** Para utilizar el sistema es necesario poseer conocimientos elementales de computación, así como de base de datos.		Analista
3	**Soporte**: Se dispondrá de documentación técnica que describa todas las funcionalidades del sistema, de modo que existirá un manual para el uso de la aplicación orientada a clientes y usuarios finales. Sistema Multiplataforma con licencia GPL/, PostgreSQL 9.1.x.		Analista
4	**Software**: Sistema Operativo: Multiplataforma. Librerías QT. Servidor de Bases de datos: SGBD PostgreSQL 9.1.x y con la herramienta de administración HABD.		Analista

2.5 Plan de iteraciones

Después de la identificación de las HU descritas por el cliente se procedió a realizar el plan de iteraciones. *En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.* (14)

Iteración 1: Esta iteración tiene como objetivo desarrollar las HU de prioridad Muy Alta, de las que depende la realización de las demás. Las historias a desarrollar en esta primera iteración son: uno (Mostrar los campos de la tabla que se desea particionar), dos (Particionar por lista) y tres (Particionar por rango), las cuales permiten obtener los objetos de la base de datos necesarios y realizar el particionado por lista y por rango. Esta iteración tiene un tiempo de duración de seis semanas.

Iteración 2: Esta iteración tiene como objetivo desarrollar la HU de prioridad Alta, la cual se desarrolla después de haber implementado las historias de la iteración uno. La historia a desarrollar en esta iteración es la cinco (administrar Partición), la cual permite eliminar y adicionar particiones. Esta iteración tiene un tiempo de duración de tres semanas.

Iteración 3: Esta iteración tiene como objetivo desarrollar las HU de menor prioridad. Las historias a desarrollar en esta última iteración son: cuatro (Mostrar las particiones realizadas) y la seis (Generar ayuda), las cuales permiten al usuario contar con una ayuda de la aplicación y mostrar los resultados finales. Esta iteración tiene un tiempo de duración de dos semanas.

2.6 Modelo de diseño

El diseño (y los modelos UML resultantes) expanden y detallan los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. Los propósitos del diseño son: Especificar una solución que trabaje y pueda ser fácilmente convertida a código fuente, construir una arquitectura fuerte, simple y fácilmente extensible.

Las clases definidas en el análisis fueron detalladas, y se añadieron nuevas clases para manejar áreas técnicas como bases de datos, interfaz de usuario, comunicación y dispositivos.

Para el diseño de aplicaciones informáticas, XP propone técnicas como las tarjetas CRC (Contenido, Responsabilidad, Colaboración), y la realización de diagramas de clase.

2.6.1 Diagrama de clases

Es un tipo de diagrama estático que describe la estructura de un sistema donde se visualizan sus relaciones entre las clases, pueden ser asociativas, de herencia, entre otros tipos de relaciones. El diagrama de clases es el diagrama principal para el análisis y diseño.

La siguiente imagen muestra el diagrama de clases del plugin a realizar:

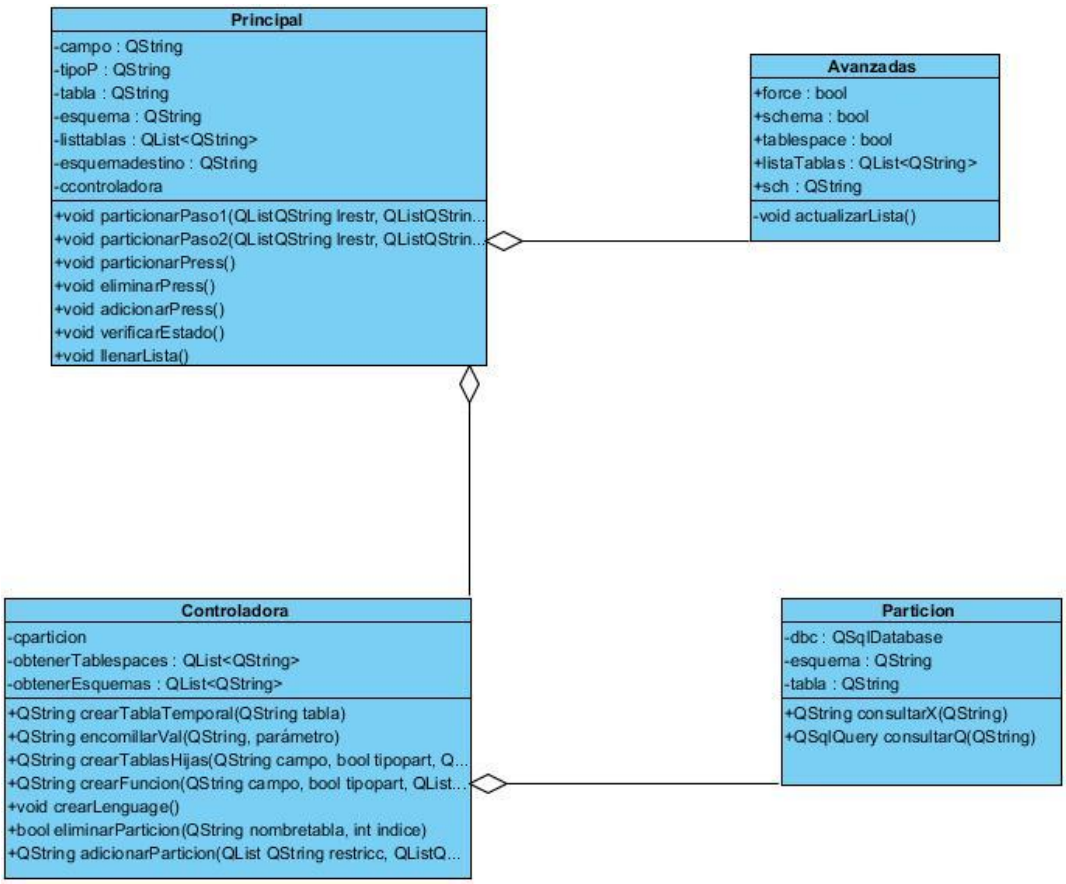


Figura 2.Diagrama de clase

Clase Particion: Esta clase se utiliza como capa de acceso a los datos en BD.

Clase Controladora: Maneja todas las funcionalidades necesarias para la aplicación. Contiene un objeto de tipo Particion.

Clase Principal: Clase encargada de manejar todos los componentes visuales así como la información recogida o mostrada por estos. Contiene un objeto de tipo Controladora.

Clase Avanzadas: Clase encargada de editar las opciones avanzadas para realizar cualquier particionado.

2.6.2 Tarjetas CRC

El uso de las tarjetas C.R.C permite al programador centrarse y apreciar el desarrollo orientado a objetos.

Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u

objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad. (15)

A continuación se muestra un ejemplo de las tarjetas CRC generadas para el diseño del plugin:

Tabla 3.Tarjeta CRC de la clase Partición

Tarjeta CRC	
Clase: *Partición*	
Responsabilidades	**Colaboraciones**
Particionar por lista *Particionar por rango* *Eliminar Partición* *Adicionar Partición* *Mostrar Particiones realizadas*	

Esta tarjeta como todas las demás desarrolladas se obtuvo del agrupamiento de las HU. En este caso las HU agrupadas fueron el particionado por lista y el particionado por rango, que dieron lugar a la clase partición, la cual no tiene colaboración con ninguna otra clase, pues no lo necesita para realizar sus responsabilidades.

2.7 Patrones de arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (16)

Estilo arquitectónico utilizado

"Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. (17)

El Patrón utilizado en el sistema es el modelo-vista-controlador este, permite dividir los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

- **Vista**: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario.
- **Controlador**: Recibe las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas,

entre otras. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista. Es el que debe de controlar los eventos.

❖ **Modelo:** Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos.

En el siguiente diagrama se muestra en el paquete vista, las clases principal y avanzada, que son las interfaces visuales que interactúan con el usuario. En el paquete modelo, se encuentra la clase partición, utilizada como capa de acceso a datos para trabajar con la base de datos; mientras que el paquete controlador contiene la clase controladora, pues es la encargada de toda la lógica del negocio. (Ver figura 3).

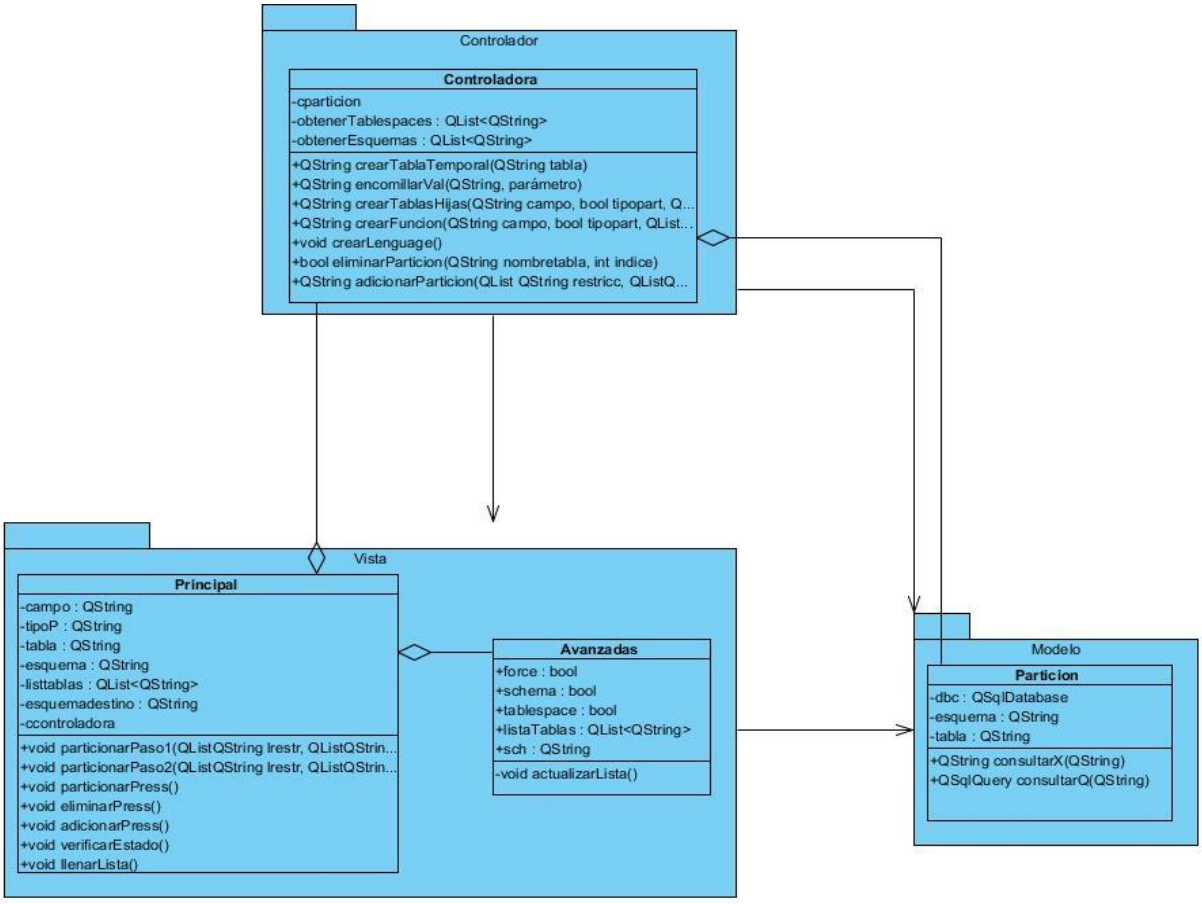


Figura 3. Modelo Vista Controlador

2.8 Patrones de Diseño

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. (18)

Son principios generales que describen un problema y describen también el núcleo de su solución, de tal forma que se pueda utilizar varias veces, sin necesidad de hacer dos veces lo mismo

Un patrón es un par/problema solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. Muchos patrones ayudan a asignar responsabilidades a los objetos.

Patrones GRASP (General Responsibility Assignment Software Patterns)

Estos patrones constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos, un patrón es una descripción de un problema y la solución a la que se le da un nombre, y que además se puede aplicar a nuevos contextos, los patrones GRASP se dividen en 6 ellos son: experto, creador, controlador, fachada, alta cohesión y bajo acoplamiento. (19)

Patrón Bajo Acoplamiento

En el patrón de bajo acoplamiento las clases deben estar lo menos ligadas entre sí que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre ellas.

En una de las clases que se evidencia el uso del patrón, es en la clase Particion, porque no depende de ninguna otra clase para realizar sus funcionalidades.

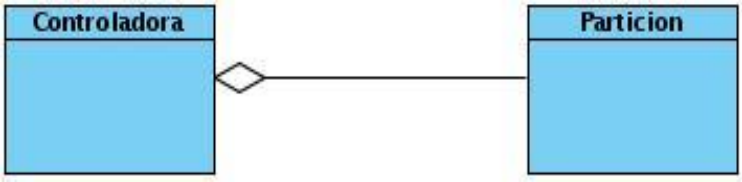


Figura 4. Patrón bajo acoplamiento

Patrón Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases, guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debe conectar con el objeto producido en cualquier evento. (20)

El uso de este patrón se evidencia en la clase Controladora la cual crea objetos de la clase Particion para realizar sus responsabilidades.

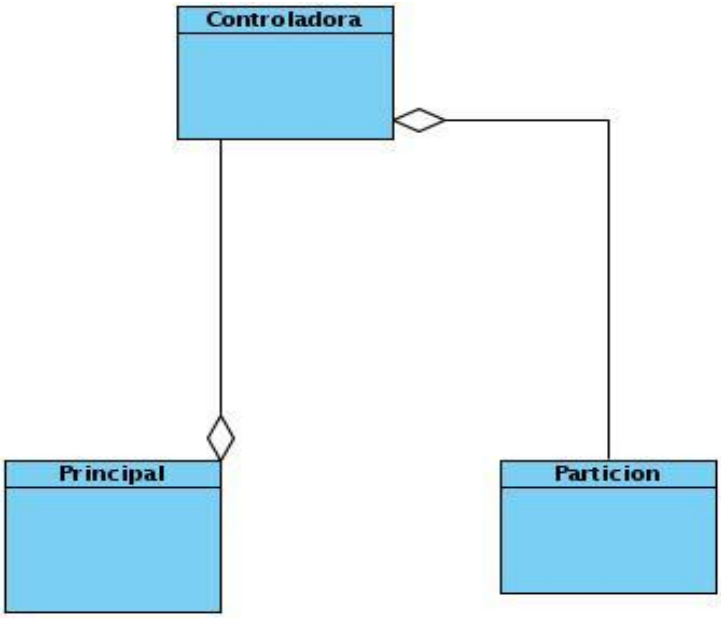


Figura 5. Patrón Creador

Patrón Experto

El patrón experto indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo; el cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. (20)

El uso de este patrón se evidencia en la clase principal, esta no necesita de la colaboración de ninguna otra clase para llevar a cabo sus responsabilidades. El uso de este patrón contribuye a un adecuado encapsulamiento, lo que favorece la robustez y fácil mantenimiento del sistema.

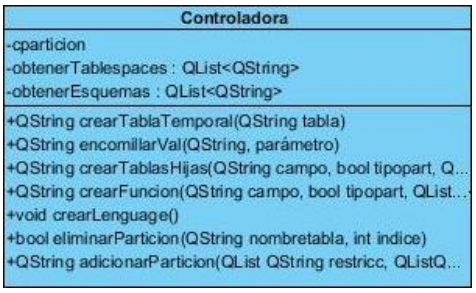


Figura 6. Patrón Experto

Patrón Alta Cohesión

El patrón Alta Cohesión se utiliza para que una clase tenga una responsabilidad moderada en un área funcional y colabore con otras clases para llevar a cabo las tareas.

En cuanto al diseño de objetos, la cohesión (o de manera más específica, la cohesión funcional) es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. (21)

En una de las clases que se evidencia el uso del patrón, es en la clase Particion, pues su responsabilidad solo radica en ejecutar consultas a la base de datos.

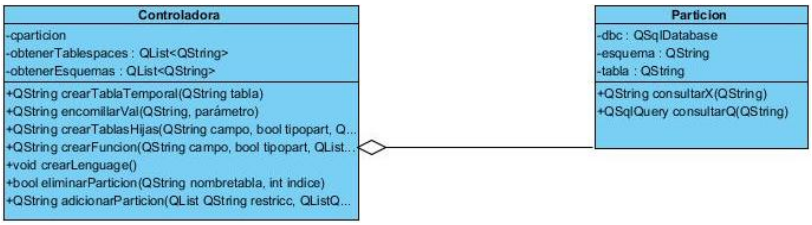


Figura 7.Patrón Alta Cohesión

Patrón Controlador

Es el patrón encargado de asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las siguientes opciones. (21)

El uso de este patrón se evidencia en la clase Controladora ya que esta es la encargada de manejar todos los eventos del sistema.

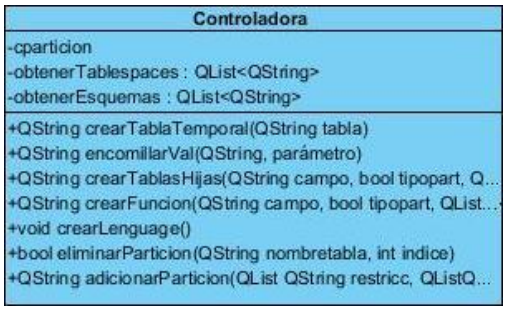


Figura 8.Patrón Controlador

Conclusiones del Capitulo

Durante la etapa de análisis y diseño del plugin se identificaron seis historias de usuario, las cuales se planificaron en tres iteraciones, desarrollando primeramente las HU de mayor prioridad. Se confeccionaron tres tarjetas CRC, y se utilizaron los patrones de diseño: Bajo Acoplamiento, Creador, Controlador, Experto y Alta Cohesión; además del patrón arquitectónico MVC.

Capítulo 3: Implementación y prueba

Introducción

En el presente capítulo se describe la implementación del plugin para solucionar a las historias de usuario especificadas en el capítulo anterior. Se muestra el estándar de codificación utilizado y parte del código de los algoritmos de las principales funcionalidades, además se especifican las pruebas a las que fue sometido el plugin, en cada una de las iteraciones, reflejadas en el informe mediante los casos de prueba.

3.1 Implementación del sistema

En la implementación el equipo de desarrollo se encarga de realizar las funcionalidades con que debe contar el sistema, las cuales fueron especificadas por el cliente mediante las historias de usuarios. La metodología Xp propone varias técnicas de programación, una de ellas es la programación en pareja, la cual permite detectar errores en el código con más facilidad, de esta forma el producto final cuenta con menos errores, los diseños son mejores, y el tamaño del código menor, además el equipo de desarrollo puede intercambiar conocimientos, lo que facilita el trabajo a la hora de resolver problemas.

3.1.1 Tareas de la ingeniería

Durante el análisis y diseño del plugin se obtuvieron las Historias de usuarios correspondientes, para un mejor desarrollo de las mismas se desglosaron en varias tareas de ingeniería, estas son asignadas a los programadores para su implementación durante una iteración, donde se encuentra una breve descripción de la tarea, las tareas de ingeniería no son para conocimiento del usuario, sino para uso estricto del programador.

A continuación se muestra una de las 16 tareas de ingeniería identificadas según las HU, esta tarea corresponde a la HU número dos nombrada “Particionar por lista”, de la cual se desglosaron seis tareas, ya que son los pasos a seguir para llevar a cabo el particionado, esta tiene un tiempo de duración de dos días llevado a una escala decimal de 0.4.

Tabla 4.Tarea de ingeniería Crear tablas hijas

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea : Crear tablas hijas	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 08/02/2012	**Fecha Fin:** 09/02/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Se crean tablas hijas para guardar la información según el criterio de partición.	

3.1.2 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código.

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios. (22)

A continuación se presentan algunas de estas convenciones utilizadas para la programación del plugin:

Identación

La unidad de identado es de 4 espacios. El uso de la tabulación debe ser evitado porque no existe un estándar que determine con precisión el ancho que va a producir la tabulación.

Ejemplo:

```
while(q.next())
{
    valor.append(q.value(0).toString());//el primer valor de la lista indica categoria
    valor.append(q.value(1).toString());//el segundo indica tipodato dentro de esa categoria
    break;
}
return valor;
```

Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas (posiblemente usted mismo) que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código.

Ejemplo:

```
QString table=cparticion->getTable();//almacena tabla que se desea particionar
QString schema=cparticion->getSchema();//almacena el esquema de dicha tabla
```

Declaración de variables

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente:

Ejemplo:

```
QString table=cparticion->getTable();//almacena tabla que se desea particionar
QString schema=cparticion->getSchema();//almacena el esquema de dicha tabla
```

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera:

```
While (condition)
{
        Statements
}
```

Ejemplo:

```
while(q.next())
{
    valor.append(q.value(0).toString());//el primer valor de la lista indica categoria
    valor.append(q.value(1).toString());//el segundo indica tipodato dentro de esa categoria
    break;
}
return valor;
```

Sentencia return

Una sentencia return no debe utilizar paréntesis “()” alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminada con un punto y coma.

Ejemplo:

```
return valor;
```

3.1.3 Interfaces de la aplicación

A continuación se muestran varias interfaces de la aplicación acompañadas de una breve descripción de cada una de ellas.

Interfaz Particionar por lista

La aplicación desarrollada consta de una interfaz que permite realizar el particionado por lista. En la parte izquierda se encuentra un listado con las columnas por las cuales se desea particionar, mostrando el nombre de la columna y su tipo de dato, más arriba se puede ver un texto que indica el esquema y la tabla en la que se encuentra. En la parte derecha de la interfaz se muestra la opción de particionar por Lista, debajo se muestran los valores detectados en la columna que esté seleccionada en ese momento, y más abajo se encuentra un campo para agregar otros criterios de particionado (ver figura 7).

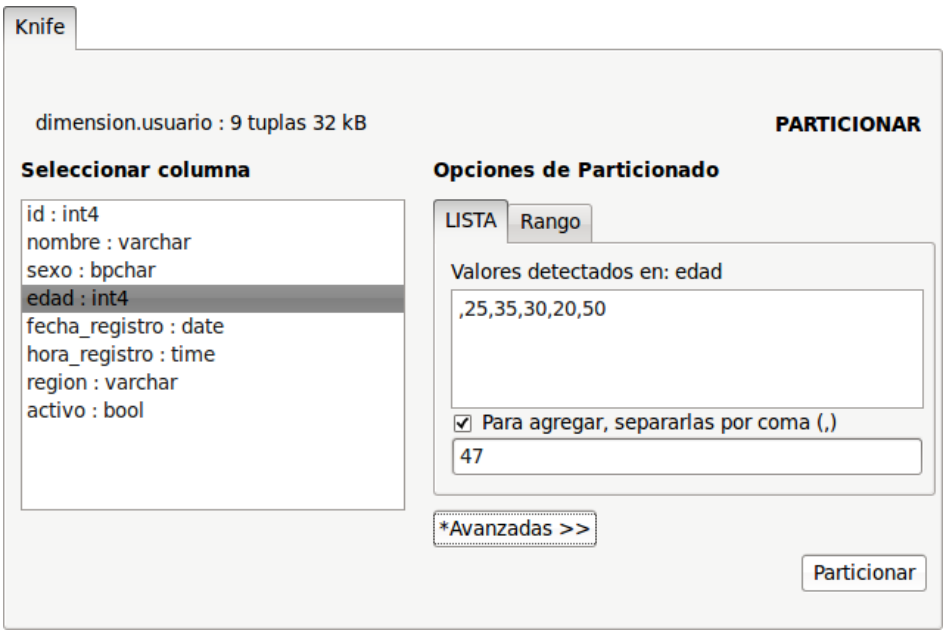


Figura 9. Particionar por lista

Interfaz Particionar por rango

En la parte derecha de la interfaz se muestra la opción de particionar por Rango, debajo se muestran los valores detectados en la columna que esté seleccionada en ese momento, más abajo se encuentran varios componentes desplegables, donde se selecciona el inicio y el final del rango que desee el usuario. En la parte inferior a la selección de rangos se ubican dos botones, el adicionar (ícono del signo más) que va adicionando los rangos que son los que establecen las particiones, y el botón eliminar (ícono del signo menos) que elimina el último rango añadido. Al terminar estas operaciones se presiona el botón que se encuentra en la parte inferior llamado “Particionar” (ver figura 8).

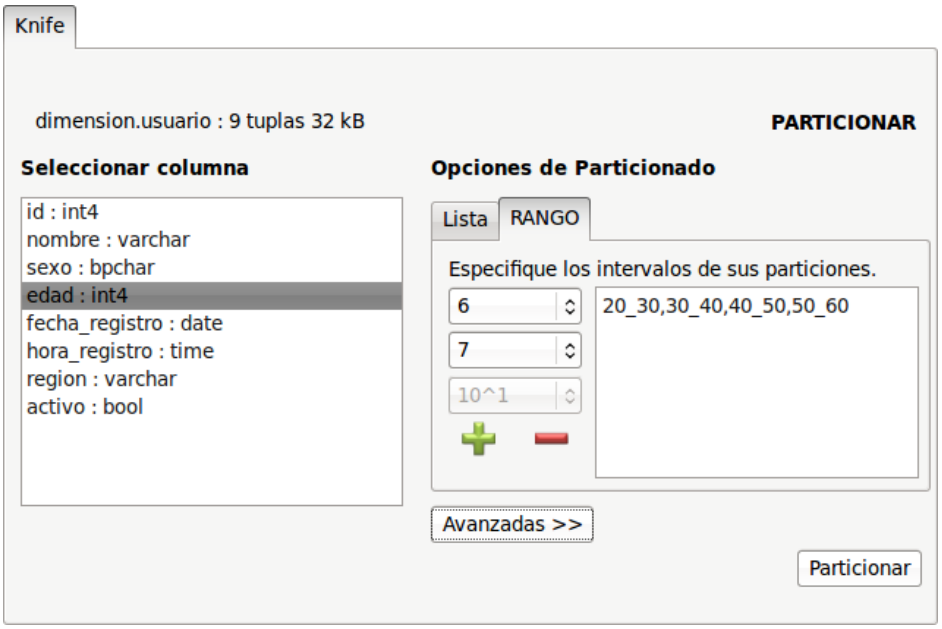


Figura 10. Particionar por rango

Interfaz Avanzada

En la interfaz Avanzada se muestra en la parte superior un componente que permite forzar el particionado, es decir, si al particionar una tabla existe algún valor que no cumple con las restricciones se fuerza el particionado para que este se guarde en una partición aparte. Inmediatamente debajo de este se encuentra el componente “Crear todas las particiones en otro esquema”, que como su nombre lo indica permite ubicar todas las particiones en otro esquema. Otro de los componentes que se muestran es el “Especificar tablespace para cada partición”, el cual al seleccionarlo muestra en el campo de texto todos las particiones a crear, de esta forma el usuario marca las que desea cambiar. Después de marcar las particiones se escoge en el componente desplegable de la parte inferior el tablespace y se presiona el botón “Cambiar a:”. Luego de realizar alguna de estas operaciones, se

presiona el botón “OK” el cual se activa si se realizó algún cambio, de lo contrario se presiona el botón “Cancel” (ver figura 9).

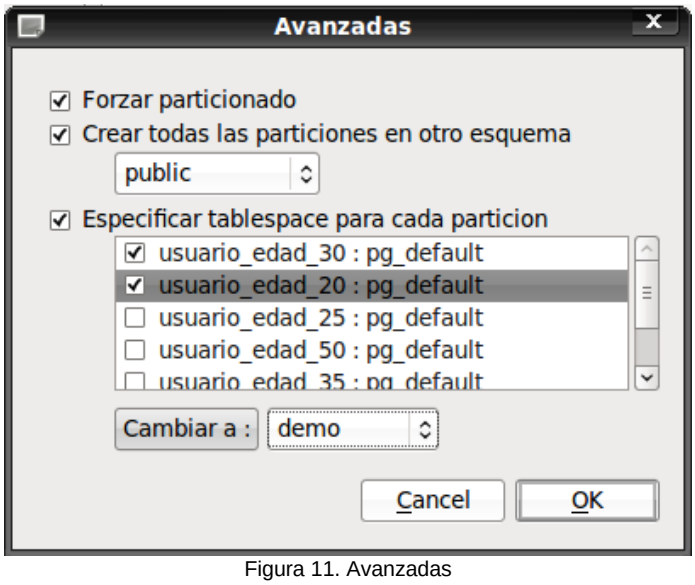


Figura 11. Avanzadas

3.2 Validación del sistema

La validación es un proceso general. Se debe asegurar que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado. Es importante llevar a cabo la validación de los requerimientos del sistema de forma inicial. Es fácil cometer errores y omisiones durante la fase de análisis de requerimientos del sistema y, en tales casos, el software final no cumplirá las expectativas de los clientes. Dentro del proceso de validación se utilizan dos técnicas de comprobación y análisis de sistemas, las inspecciones del software y las pruebas del software. (23)

Las pruebas del software permiten saber si lo que se implementó satisface las necesidades del cliente planteadas en las historias de usuarios. Las pruebas indican que el trabajo funciona.

No se debe escribir tan solo una prueba, ver que funciona y dar por cumplido el trabajo. Se debe pensar en todas las posibles pruebas para el código, las mismas deben ser sensatas y valientes, no se pueden hacer pruebas simples que no testen a fondo el sistema, los agujeros que se van dejando al final salen a relucir. Todas las características del programa deben ser probadas. Los programadores son los encargados de escribir las pruebas para chequear el correcto funcionamiento del programa y los clientes realizan las pruebas funcionales.

3.2.1 Enfoques de diseños de pruebas

Existen tres enfoques principales para el diseño de casos de pruebas:

- *El enfoque estructural o de caja blanca. Se centra en la estructura interna del programa (analiza los caminos de ejecución).*
- *El enfoque funcional o de caja negra. Se centra en las funciones, entradas y salidas.*
- *El enfoque aleatorio consiste en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.* (24)

Para el desarrollo de la aplicación se utiliza el método de caja negra y en el posterior epígrafe se explica en que consiste este método.

3.2.2 Diseño de Casos de pruebas basados en HU

Los casos de prueba representan los datos que se utilizarán como entrada para ejecutar el software a probar. Más concretamente los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Como se verá posteriormente, cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba. Por lo tanto, durante la tarea de generación de estos casos, se han de confeccionar los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso. (25)

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione bien.

Existen varios tipos de pruebas que se le pueden realizar a un sistema. Después del estudio realizado se decidió efectuar las pruebas de aceptación y pruebas de funcionalidad mediante el método de caja negra, y la técnica de partición de equivalencia. A continuación se explica dicha prueba.

***Prueba de aceptación*:** *El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.* (26)

Las pruebas de aceptación se utilizan como técnica para garantizar que los requerimientos hayan sido cumplidos y que la aplicación es realmente lo que el cliente necesita, además de asegurar su correcto

funcionamiento son realizadas las pruebas de aceptación. Las cuales son creadas a partir de las historias de usuario y desde la perspectiva del cliente. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento.

***Prueba Funcional*:** *La prueba funcional es un proceso para procurar encontrar discrepancias entre el programa y la especificación funcional. Una especificación funcional es una descripción exacta del comportamiento del programa desde el punto de vista del usuario final*. (26)

Método de caja Negra.

El método de caja negra permite definir las entradas al sistema y los resultados esperados de estas entradas. Se realiza con el objetivo de asegurar que las funcionalidades del sistema cumplan con lo que se espera de ellas, y son definidas por el cliente para cada historia de usuario. Este método permite identificar claramente las entradas y salidas, estudiar las relaciones que existen entre ellas, permitiendo así maximizar la eficiencia de los sistemas sin tener que introducirse en los procesos complejos que se encuentran en la Caja Negra.

Técnica de partición de equivalencia

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error.

El diseño de casos de prueba según esta técnica consta de dos pasos:

1. Identificar las clases de equivalencia.

2. Identificar los casos de prueba. (27)

Para la realización de las pruebas se diseñaron seis casos de prueba teniendo en cuenta el principio de diseñar un caso de prueba por HU. A continuación se representa el Diseño de caso de prueba para la HU Administrar Partición (ver tabla 5).

Dicho caso de prueba cuenta con cuatro escenarios: eliminar partición, eliminar partición incorrectamente, adicionar partición por lista y adicionar partición por rango. Estos escenarios son las operaciones que se realizan al administrar una partición, en las cuales pueden ocurrir errores a la hora de su aplicación, para probar estos errores en el plugin se definieron cuatro variables, donde la variable uno es la partición que se desea eliminar, la variables dos son las restricciones para adicionar una partición por lista, la variable tres y cuatro son el inicio y fin del rango que se introduce al adicionar una partición por rango.

Tabla 5.Caso de Prueba: Administrar Partición

Escenario	Descripción	Variable1	Variable 2	Variable 3	Variable 4	Respuesta	Flujo central
EC 1 Eliminar partición.	*Permite eliminar una partición de las existentes.*	*V*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje: "Particionado eliminado exitosamente".*	*1-Presionar el botón de la opción eliminar en la barra de herramienta. 2-Seleccionar la partición a eliminar 3-Presionar botón Eliminar.*
		(Persona)					
EC 1.1 Eliminar partición incorrectamente.		*I*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje: "Seleccione una partición".*	
		()					

EC 2 Adicionar partición por lista.	*Permite adicionar una partición a una tabla ya particionada*	*NA*	*V*	*NA*	*NA*	*El sistema muestra un mensaje: "Particionado realizado correctamente".*	*1-eleccionar la opción adicionar en la barra de herramienta. 2-Agregar las particiones. 3-Presionar botón Agregar.*
			(femenino)				
EC 1.2 Adicionar partición por rango		*NA*	*NA*	*V*	*V*	*El sistema muestra un mensaje: "Particionado realizado correctamente".*	
				(20)	*(30)*		

Después de aplicados los casos pruebas en la aplicación, se detectaron cinco resultados no satisfactorios; los cuales pasaron a ser no conformidades y se emitieron en el registro de defectos detectados. Estas no conformidades fueron detectadas en la primera iteración, las cuales ya en una segunda iteración fueron corregidas satisfactoriamente y de esta forma se cumplieron con todas las expectativas del cliente.

A continuación se muestran las no conformidades detectadas durante las pruebas a la aplicación.

Tabla 6.No conformidades

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	El mensaje que se muestra cuando se elimina una partición aparece con falta de ortografía (Partición). [ver anexo 1]	Administrar partición	Prueba	NS	10/05/2012 PD 11/05/2012 RA	Se corrigió la tilde en la palabra Partición.
Aplicación	2	No muestra el mensaje Particionado eliminado	Administrar partición	Prueba	S	10/05/2012 PD 11/05/20	Al eliminar la partición ya muestra el mensaje

		exitosamente.				12 RA	Particionado eliminado exitosamente.
Aplicación	3	Al presionar el botón “Avanzadas >>” sin haber seleccionado una columna se cierra la aplicación.	Particionar por lista	Prueba	S	10/05/2012 PD 12/05/2012 RA	El sistema al presionar el botón “Avanzadas>>” muestra un mensaje: No se pudo mostrar la ventana por alguna de las razones siguientes: -No ha seleccionado una columna de la tabla. -No ha especificado restricciones a aplicar. -Las restricciones especificadas aparecen repetidas o coinciden con las detectadas
Aplicación	4	AL presionar el botón de adicionar intervalo si no hay columnas seleccionadas no sucede nada.	Particionar por rango	Prueba	R	10/05/2012 PD 11/05/2012 NP	El sistema no debe informar sobre esa operación.
Aplicación	5	No se puede eliminar múltiples rangos, solo eliminar el último añadido.	Particionar por rango	Prueba	R	10/05/2012 PD 11/05/2012 NP	Esto sucede para garantizar los intervalos consecutivos, y que no existan valores fuera de rango.

Conclusiones del capítulo

De las seis historias de usuario identificadas en la fase de análisis se logró implementar el 100% de las mismas. La realización de pruebas funcionales a la aplicación a través del diseño de 6 casos de prueba uno por historias de usuario, arrojaron como resultado 5 no conformidades en una primera iteración solucionadas todas en una segunda iteración, demostrando que la herramienta desarrollada cumple con las características especificadas por el cliente.

Conclusiones generales

❖ Se identificaron seis HU de las que se definieron siete funcionalidades: Mostrar los campos de la tabla que se desea particionar, Particionar por lista, Particionar por rango, Eliminar partición, Adicionar partición, Mostrar las particiones realizadas y Generar ayuda. Estas funcionalidades permiten automatizar el proceso de particionado de tablas en bases de datos PostgreSQL.

❖ Se obtuvo el diagrama de clases estructurado por la utilización del patrón arquitectónico MVC y los patrones de diseño Bajo acoplamiento, Alta cohesión, Creador, Controlador, Experto permitiendo un diseño robusto del plugin.

❖ Se implementó un plugin que permite realizar el particionado de tablas en las variantes soportadas por el gestor PostgreSQL, por lista y por rango.

❖ Se realizaron pruebas de aceptación por parte del cliente a través del diseño de 6 casos de prueba lo que permitió verificar que la aplicación funciona como se diseñó y que los requisitos se cumplen cabalmente.

Recomendaciones

- Implementar un trigger para las consultas UPDATE que se le realicen a la tabla.
- Sugerirle al usuario el tipo de particionado más adecuado según el tipo de dato de la columna seleccionada.

Referencias bibliográficas

1. Manual de Ayuda. [En línea] [Citado el: 25 de Septiembre de 2011.] http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql/caracteristicas-de-postgresql-0184.

2. Cavsi. [En línea] 2008. [Citado el: 25 de Septiembre de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

3. Sitio Oficial de PostgreSQL. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.postgresql.org.es/sobre_postgresql .

4. **Humberto, Espinoza.** PostgreSQL, Una Alternativa de DBMS Open Source. [En línea] 2005. [Citado el: 15 de Octubre de 2011.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

5. DATA Prix. [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.particionado-tablas-oracle.htm.

6. PostgreSQL. [En línea] [Citado el: 6 de Diciembre de 2011.] http://www.postgresql.org/docs/9.1/static/ddl-partitioning.html .

7. ExplicaXP. [En línea] [Citado el: 10 de Enero de 2012.] www.willydev.net/descargas/prev/ExplicaXP.pdf .

8. Sitio de descargas de software. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_14720_p/.

9. Definición de UML. MASTERMAGAZINE. [En línea] 17 de Febrero de 2012. www.mastermagazine.info/termino/7006.php..

10. LENGUAJES DE PROGRAMACIÓN. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm.

11. Lenguajes de Programación. [En línea] [Citado el: 12 de Enero de 2012.] es.scribd.com/doc/20243205/Lenguaje-de-Programacion.

12. **Jacobson, I. y Booch, G.** El proceso unificado de desarrollo de software. 2004. Vol. 1.

13. **Letelier, Patricio, Sanchez, Emilio A y Canos, Jose H.** Código de calidad: Integrando Patrones de Refactoring. [En línea] [Citado el: 26 de Enero de 2012.] http://lsi.ugr.es/~gedes/actividades/Dolmen4/a1.pdf.

14. Ciclo de vida de un proyecto XP. [En línea] [Citado el: 23 de Enero de 2012.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

15. Fases de la Programación Extrema. [En línea] [Citado el: 19 de Febrero de 2012.] http://programacionextrema.tripod.com/fases.htm#segundaFase.

16. Introducción a Patrones. [En línea] Facultad de Ciencias, UNAM. [Citado el: 10 de Febrero de 2012.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

17. **Billy, Reynoso y ,Carlos .** [En línea] 20 de Febrero de 2012. www.willydev.net/descargas/prev/IntroArq.pdf.

18. Ingenieros Software. [En línea] 3 de Mayo de 2012. www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

19. Patrones . [En línea] [Citado el: 10 de Febrero de 2012.] http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf .

20. **Veloso Hernández, Pedro.** *Uso de patrones de arquitectura.*

21. **Larman, Craig.** *UML y Patrones.*

22. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] Departamento de Sistemas Informáticos y Computación (DSIC). [Citado el: 15 de Marzo de 2012.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

23. Validacion de sistemas ingenieria de software. [En línea] [Citado el: 9 de Mayo de 2012.] http://www.ctr.unic.es.

24.Pruebas del Software. [En línea] [Citado el: 13 de Abril de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf.

25. Ingienería del software un enfque práctico. [En línea] Universidad Pontificia de Salamanca. [Citado el: 6 de Mayo de 2012.] http://es.scribd.com/doc/7978336/Ingenieria-de-Software-Un-Enfoque-Practico-Pressman-5th-Ed.

26. Pruebas de Software. [En línea] [Citado el: 8 de Marzo de 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node55.html.

27. TÉCNICAS DE EVALUACIÓN DINÁMICA . [En línea] [Citado el: 5 de Mayo de 2012.] www.lsi.us.es/docencia/get.php?id=361.

Bibliografía

1. Manual de Ayuda. [En línea] [Citado el: 25 de Septiembre de 2011.] http://www.manualesdeayuda.com/manuales/bases-de-datos/postgresql/caracteristicas-de-postgresql-0184.

2. LENGUAJES DE PROGRAMACIÓN. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm.

3. Lenguajes de Programación. [En línea] [Citado el: 12 de Enero de 2012.] es.scribd.com/doc/20243205/Lenguaje-de-Programacion.

4. **Jacobson, I. y Booch, G.** El proceso unificado de desarrollo de software. 2004. Vol. 1.

5. **Letelier, Patricio, Sanchez, Emilio A y Canos, Jose H.** Código de calidad: Integrando Patrones de Refactoring. [En línea] [Citado el: 26 de Enero de 2012.] http://lsi.ugr.es/~gedes/actividades/Dolmen4/a1.pdf.

6. Ciclo de vida de un proyecto XP. [En línea] [Citado el: 23 de Enero de 2012.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

7. Fases de la Programación Extrema. [En línea] [Citado el: 19 de Febrero de 2012.] http://programacionextrema.tripod.com/fases.htm#segundaFase.

8. Introducción a Patrones. [En línea] Facultad de Ciencias, UNAM. [Citado el: 10 de Febrero de 2012.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

9. **Billy, Reynoso y ,Carlos .** [En línea] 20 de Febrero de 2012. www.willydev.net/descargas/prev/IntroArq.pdf.

10. Ingenieros Software. [En línea] 3 de Mayo de 2012. www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

11. Patrones . [En línea] [Citado el: 10 de Febrero de 2012.] http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf .

12. Cavsi. [En línea] 2008. [Citado el: 25 de Septiembre de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

13. **Veloso Hernández, Pedro.** *Uso de patrones de arquitectura.*

14. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] Departamento de Sistemas Informáticos y Computación (DSIC). [Citado el: 15 de Marzo de 2012.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

15. Validacion de sistemas ingenieria de software. [En línea] [Citado el: 9 de Mayo de 2012.] http://www.ctr.unic.es.

16. Pruebas del Software. [En línea] [Citado el: 13 de Abril de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf.

17. Ingienería del software un enfque práctico. [En línea] Universidad Pontificia de Salamanca. [Citado el: 6 de Mayo de 2012.] http://es.scribd.com/doc/7978336/Ingenieria-de-Software-Un-Enfoque-Practico-Pressman-5th-Ed.

18. Pruebas de Software. [En línea] [Citado el: 8 de Marzo de 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node55.html.

19. TÉCNICAS DE EVALUACIÓN DINÁMICA . [En línea] [Citado el: 5 de Mayo de 2012.] www.lsi.us.es/docencia/get.php?id=361.

20.**Tolava , Martin.** La vanguardia, el mejor camino para la gestión de proyectos. [En línea] 2011. [Citado el: 15 de Abril de 2012.] http://www.taringa.net/posts/linux/8011989/La-vanguardia_-el-mejor-camino-para-la-gestion-de-proyectos.html..

21.[En línea] 2011. [Citado el: 20 de Febrero de 2012.] http://www.postgresql.org.es/sobre_postgresql.

22.**Milián Iglesias, Ridosbey.** *“Propuesta de un entorno de gestión de indicadores sobre herramientas libres para medir el proceso de desarrollo del software”.* 2009.

23. Sitio Oficial de PostgreSQL. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.postgresql.org.es/sobre_postgresql .

24.Taller temático. [En línea] [Citado el: 25 de Septimbre de 2011.] http://tallertematico.fordes.co.cu/index.php/ttca/ttca2.

25.**Sanchez E y Letelier P.** *Mejorando la gesti_on de historias de usuario en eXtreme Programming.*

26.**C. Alexander, y otros, y otros.** *"A Pattern Language".* s.l. : C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, y S. Angel,, Oxford University Press, New York.

27. **Humberto, Espinoza.** PostgreSQL, Una Alternativa de DBMS Open Source. [En línea] 2005. [Citado el: 15 de Octubre de 2011.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

28. DATA Prix. [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.particionado-tablas-oracle.htm.

29. PostgreSQL. [En línea] [Citado el: 6 de Diciembre de 2011.] http://www.postgresql.org/docs/9.1/static/ddl-partitioning.html .

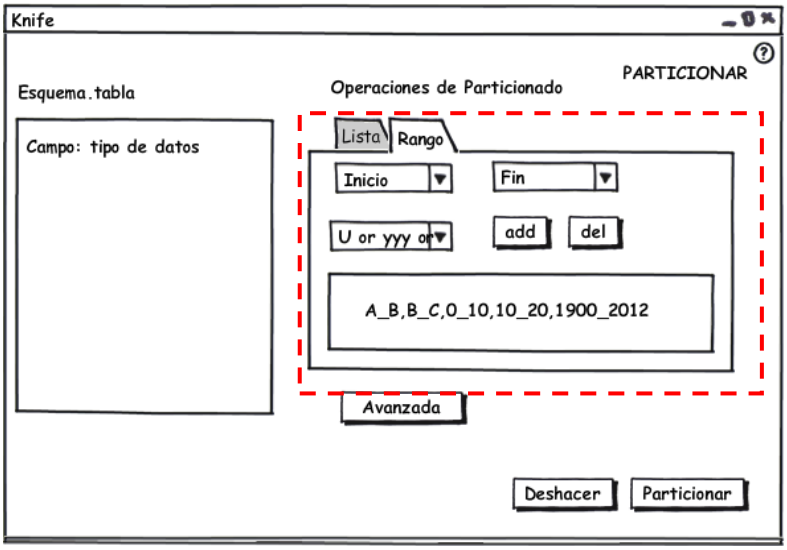
30. ExplicaXP. [En línea] [Citado el: 10 de Enero de 2012.] www.willydev.net/descargas/prev/ExplicaXP.pdf .

31. Sitio de descargas de software. [En línea] [Citado el: 5 de Diciembre de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_14720_p/.

32. Definición de UML. MASTERMAGAZINE. [En línea] 17 de Febrero de 2012. www.mastermagazine.info/termino/7006.php..

33. **Larman, Craig.** *UML y Patrones.*

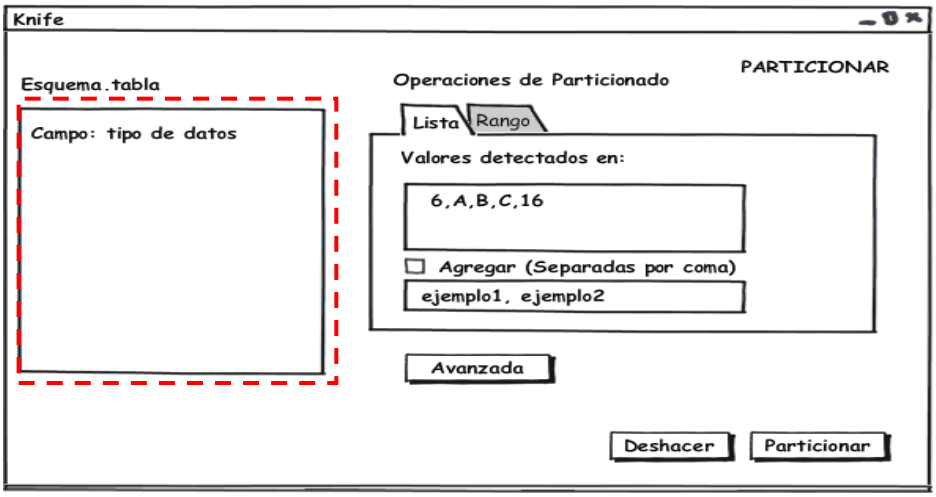
Anexos



Anexo 1: Prototipo de interfaz **"Particionar por rango"**

Historia de Usuario	
Nombre de la Historia de Usuario: Mostrar los campos de la tabla que se desea particionar.	
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Yehimy Figueredo Falcón	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 1 Semana
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 1 Semana
Descripción: Permite al usuario ver los campos de la tabla que desea particionar.	
Observaciones: Para mostrar al usuario los campos de la tabla debe estar autenticado el servidor de base de datos.	

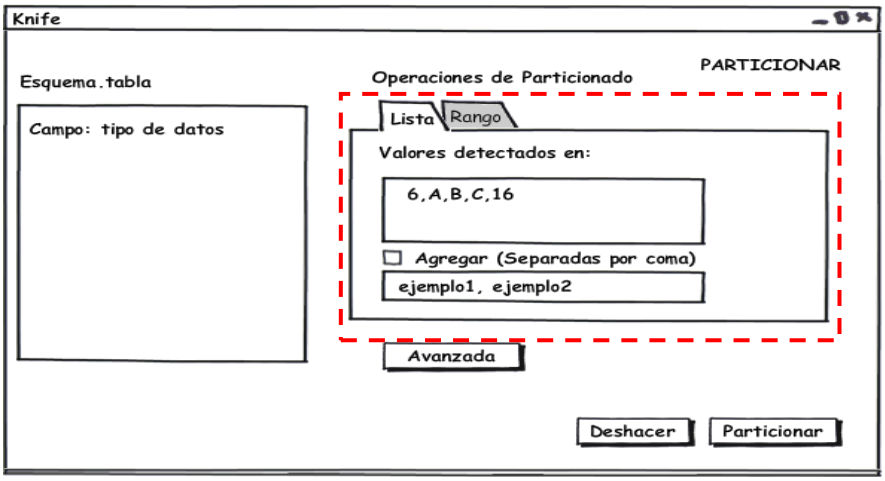
Prototipo de interfaces:



Anexo 2: HU **"Mostrar los campos de la tabla que se desea particionar"**

Historia de Usuario	
Nombre de la Historia de Usuario: Particionar por lista	
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Yehimy Figueredo Falcón	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 3 Semanas
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 3 Semanas
Descripción: Permite al usuario seleccionar el esquema, la tabla y el campo por el que desea particionar por lista, además le permite introducir el nombre que quiere que tomen las particiones creadas.	
Observaciones: El usuario debe escoger el esquema, la tabla y el campo por el que desea particionar.	

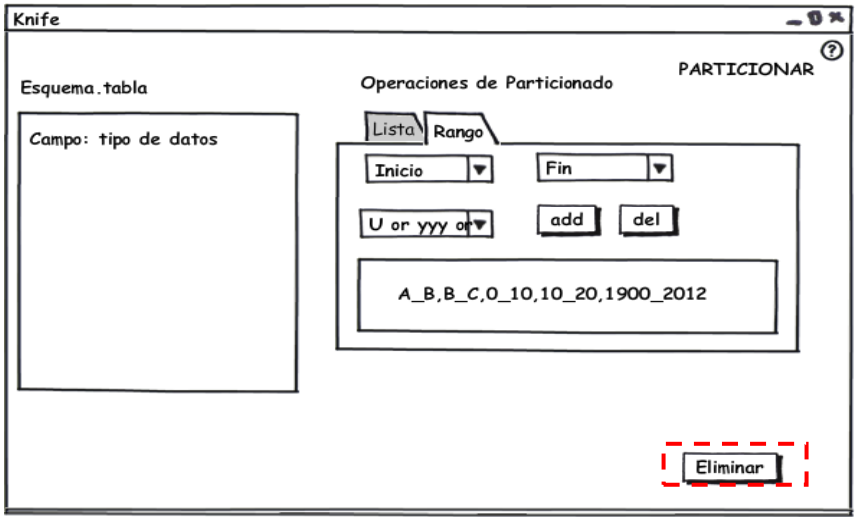
Prototipo de interfaces:

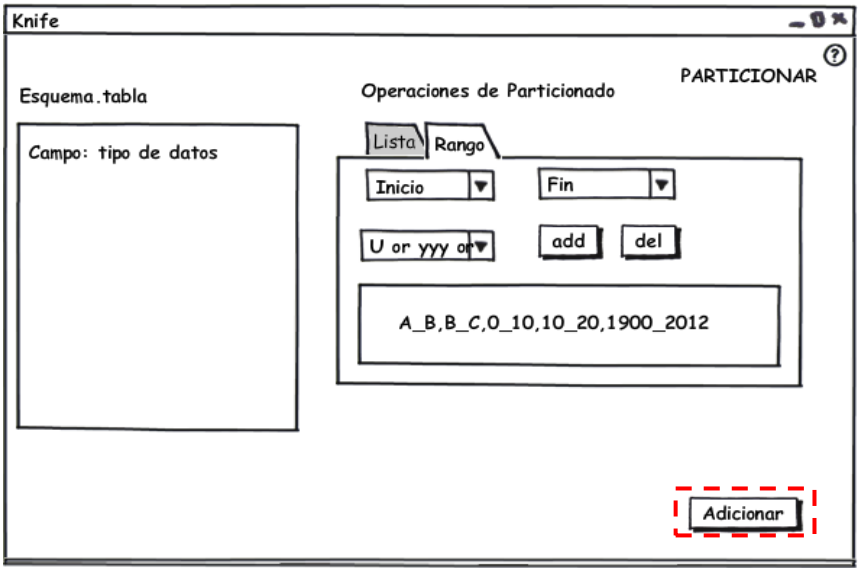


Anexo 3: HU **"Particionar por lista"**

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Administra Partición
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Ramiro Rodríguez Pajares	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 3 Semanas
Riesgo en desarrollo: Alta	**Puntos reales:** 3 Semanas
Descripción: El usuario debe poder realizar operaciones en sus particiones como eliminar y adicionar	
Observaciones:	

Prototipo de interfaces:





Anexo 4: HU **"Administrar Partición"**

Tarjeta CRC	
Clase: *Ayuda*	
Responsabilidades	**Colaboraciones**

Generar ayuda	

Anexo 5: Tarjeta CRC **"*Ayuda*"**

Tarjeta CRC	
Clase: Tabla.	
Responsabilidades	**Colaboraciones**
Mostrar los campos de la tabla que se desea particionar.	

Anexo 6: Tarjeta CRC **"Tabla"**

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Capturar atributos de la tabla.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 30/01/2012	**Fecha Fin:** 01/02/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se realiza la captura los atributos de la tabla que se seleccione.	

Anexo 7: Tarea de Ingeniería **"Capturar atributos de la tabla"**

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Mostrar los atributos de la tabla.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 02/02/2012	**Fecha Fin:** 03/02/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se muestran los atributos de la tabla seleccionada.	

Anexo 8: Tarea de Ingeniería **"Mostrar los atributos de la tabla"**

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea : Crear tabla temporal	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 06/02/2012	**Fecha Fin:** 07/02/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Se crea una tabla temporal para guardar la información de la tabla que se quiere particionar; y de esta forma garantizar la permanencia de los datos.	

Anexo 10: Tarea de Ingeniería **"Crear tabla temporal"**

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 2
Nombre Tarea : Agregar restricciones a las tablas hijas	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 10/02/2012	**Fecha Fin:** 14/02/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Se agregan restricciones a las tablas hijas para que la información se guarde en la tabla que le corresponde según el criterio de partición.	

Anexo 9: Tarea de Ingeniería **"Agregar restricciones a las tablas hijas"**

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 2
Nombre Tarea : Crear índices a cada partición	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 15/02/2012	**Fecha Fin:** 16/02/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Crear índices a cada una de las particiones.	

Anexo 10: Tarea de Ingeniería **"Crear índices a cada partición"**

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 2
Nombre Tarea : Definir reglas o trigger	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 17/02/2012	**Fecha Fin:** 21/02/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se definen reglas o trigger para redirigir las modificaciones de la tabla padre.	

Anexo 11: Tarea de Ingeniería **"Definir reglas o trigger"**

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 2
Nombre Tarea : Verificar el parámetro constraint_exclusion	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 22/02/2012	**Fecha Fin:** 24/02/2012

Programador Responsable: Yehimy Figueredo Falcón
Descripción: Se verifican si están habilitados los parámetros

Anexo 12: Tarea de Ingeniería **"Verificar el parámetro constraint_exclusion**"

	Tarea de Ingeniería
Número Tarea:	**Número Historia de Usuario:** 3
Nombre Tarea : Crear tablas hijas para particionado por rango	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 27/02/2012	**Fecha Fin:** 06/03/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Se crean tablas hijas para guardar la información según el criterio de partición.	

Anexo 13: Tarea de Ingeniería **"Crear tablas hijas para particionado por rango**"

	Tarea de Ingeniería
Número Tarea:	**Número Historia de Usuario:** 3
Nombre Tarea : Definir reglas o trigger para el particionado por rango	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 07/03/2012	**Fecha Fin:** 13/03/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Se definen reglas o trigger para redirigir las modificaciones de la tabla padre.	

Anexo 14: Tarea de Ingeniería **"Definir reglas o trigger para el particionado por rango**"

	Tarea de Ingeniería
Número Tarea:	**Número Historia de Usuario:** 4
Nombre Tarea: Capturar particiones realizadas	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 14/03/2012	**Fecha Fin:** 16/03/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se capturan las particiones realizadas para posteriormente mostrarlas.	

Anexo 15: Tarea de Ingeniería **"Capturar particiones realizadas**"

	Tarea de Ingeniería
Número Tarea:	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar particiones realizadas	

Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 19/03/2012	**Fecha Fin:** 21/03/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se muestran las particiones realizadas.	

Anexo 16: Tarea de Ingeniería **"Mostrar particiones realizadas**"

Tarea de Ingeniería	
Número Tarea:	**Número Historia de Usuario:** 5
Nombre Tarea: Eliminar Partición	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1Semana
Fecha Inicio: 22/03/2012	**Fecha Fin:** 28/03/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Permite eliminar cualquier partición seleccionada.	

Anexo 17: Tarea de Ingeniería **"Eliminar Partición**"

Tarea de Ingeniería	
Número Tarea:	**Número Historia de Usuario:** 5
Nombre Tarea: Adicionar partición	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1Semana
Fecha Inicio: 29/03/2012	**Fecha Fin:** 04/04/2012
Programador Responsable: Ramiro Rodríguez Pajares	
Descripción: Permite adicionar una nueva particiones.	

Anexo 18: Tarea de Ingeniería **"Adicionar partición**"

Tarea de Ingeniería	
Número Tarea:	**Número Historia de Usuario:** 6
Nombre Tarea: Generar ayuda	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 Semana
Fecha Inicio: 05/04/2012	**Fecha Fin:** 11/04/2012
Programador Responsable: Yehimy Figueredo Falcón	
Descripción: Se muestra una ayuda donde explique como utilizar el plugin.	

Anexo 19: Tarea de Ingeniería **"Generar ayuda**"

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1 Generar ayuda.	*Muestra la ayuda.*	*El sistema muestra la ayuda.*	*1-Seleccionar lel icono de la ayuda.*

Anexo 20: Caso de prueba **"Generar ayuda**"

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1 Mostrar los campos de la tabla que se desea particionar	*Visualiza las columnas de la tabla seleccionada, a la cual se le realizará el particionado, permitiendo seleccionar uno de estos.*	*Muestra los campos de la tabla seleccionada, junto a su tipo de dato.*	*1- Seleccionar la tabla.*

Anexo 21: Caso de prueba **"Mostrar los campos de la tabla que se desea particionar**"

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Mostrar las particiones realizadas	*Muestra las particiones de la tabla.*	*V(Persona)*	*Muestra un formulario con los nombres de las particiones de la tabla seleccionada.*	*1-Seleccionar la tabla particionada.*

Anexo 22: Caso de prueba **"Mostrar las particiones realizadas**"

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7	Variable 8	Variable 9	Respuesta del sistema	Flujo central
EC 1Particionar por lista agregando restricciones correctas.	*Permite particionar por lista satifactoriamente.*	*V*	*V*	*V*							*El sistema muestra un mensaje: "Particionado realizado exitosamente".*	*1- Seleccionar la columna.* *2- Marcar opción Agrega restricciones que es opcional.* *3- Añadir las restricciones.* *4- Precionar botón Particionar*
		(nombre :varchar)		*(Femenino,Masculino)*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*		
					NA	*NA*	*NA*	*NA*	*NA*	*NA*		

EC 1.2 Particionar por lista con restricciones incorrectas.		*V (nombre:varchar)*	*V V*	*I (femenino,femenino)*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje de error:"Se encontraron restricciones repetidas que serán tomadas solo una vez".*	
EC 1.3 Particionar por lista sin restricciones.		*V (nombre:varchar)*	*V*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje: "Tabla vacía. Debe agregar restricciones".*	
EC 1.4 Particionar por lista incorrectamente.		*I ()*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje de error: "Debe seleccionar una columna de la tabla".*	
EC 1.5 Particionar por lista con la opcion Avanzadas.	*Permite particionar por lista satifactoriamente y utilizar otras opciones de para el particionado.*	*V (nombre:varchar)*	*V*	*V*	*V*	*V*	*V(public)*	*V*	*V(Persona_sexo_f)*	*V(pg_default)*		*1- Si se presiona el botón avanzada, se muestra un nuevo formulario con las siguen*

												tes opcion es: - Forzar Partici onado. -Crear tablas en otro esque ma. - Especi ficar tables pace para cada tabla. 2- Luego se presio na el botón ok o cancel ar(el botón ok se activa mientr as se selecci one al menos una de las opcion es).

EC 1.5 Particionar por lista con la opcion Avanzadas incorrectamente.		*V ()*	*V*	*V*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje de error: "No se pudo mostrar el dialogo por alguna de las razones siguientes: -No ha selecionado una columna de la tabla. -No ha especificado restricciones a aplicar. -Las restricciones especificadas aparecen repetidas o coinciden con las detectadas" .*	

Anexo 23: Caso de prueba **"Particionar por lista**"

Escenario	**Descripción**	**Variable 1**	**Variable 2**	**Variable 3**	**Variable 4**	**Variable 5**	**Variable 6**	**Variable 7**	**Variable 8**	**Variable 9**	**Respuesta del sistema**	**Flujo central**
EC 1 Particionar	*Permite particionar*	*V*	*V(20)*	*V(30)*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema*	*1- Selecci*

por rango correctamente.	*por rango una tabla.*	*(edad:int3)*									*muestra un mensaje: Particionado realizado exitosamente.*	*onar la columna.* *2- Especificar el inicio y fin del intervalo.* *3- Añadir restricciones.* *- Eliminar restricciones que se elimina la ultima restricción.* *5- Precionar botón Particionar.*
EC 1.2 Particionar por rango sin restricciones.		*V (edad:int3)*	*V(20)*	*V(30)*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje: Tabla vacia. Debe agregar restricciones.*	

EC 1.3 Particionar por rango incorrectamente.		*I (edad:int3)*	*V(20)*	*I()*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje de error: debe Incapaz de agregar intervalo sin final.Para garantizar estos datos, fuerce el particionado.*	
EC 1.4 Particionar por rango con la opcion Avanzadas.	*Permite particionar por lista satifactoriamente y utilizar otras opciones para el particionado.*	*V (edad:int3)*	*V(20)*	*V(30)*	*V*	*V*	*V(public)*	*V*	*V(Persona_sexo_f)*	*V(pg_default)*	*El sistema muestra un mensaje: Particionado realizado exitosamente.*	*1- Si se presiona el botón avanzada, se muestra un nuevo formulario con las siguientes opciones: - Forzar Particionado. -Crear tablas en otro esquema. - Especificar tablespace para cada tabla. 2- Luego se*

												presiona el botón ok o cancelar (el botón ok se activa si escoge al menos una de las opciones).
EC 1.5 Particionar por rango con la opcion Avanzadas incorrectamente.		*V ()*	*V*	*V*	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*	*El sistema muestra un mensaje de error : El dialog no se puede mostrar por alguna de las siguientes razones: -No ha seleccionado un campo. -No existen restricciones.*	

Anexo 24: Caso de prueba **"Particionar por rango**"

Glosario de términos

A

Arquitectura: La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Una Arquitectura de Software, también denominada *Arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco.

Artefacto: Son el resultado de trabajo parcial o final que es producido y usado durante un proyecto. Los artefactos son usados para capturar y llevar la información del proyecto.

Aplicación: Una **aplicación** es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo.

B

Base de Datos: Una **base de datos** o **banco de datos** (en ocasiones abreviada con la sigla *BD* o con la abreviatura *b. d.*) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Backup: Una **copia de seguridad** o ***backup*** (su nombre en inglés) en tecnología de la información o informática es una copia de seguridad - o el proceso de copia de seguridad - con el fin de que estas copias adicionales puedan utilizarse para restaurar el original después de una eventual pérdida de datos.

C

Columna: En el contexto de una tabla de base de datos relacional, una **columna** es un conjunto de valores de datos de un simple tipo particular, uno por cada fila de la tabla. Las columnas proporcionan la estructura según la cual se componen las filas.

Consulta: Una consulta es el método para acceder a los datos en las bases de datos.
Componente: Un componente es cualquiera de los elementos que podemos insertar en una ficha, tanto si su función es visual como si no lo es (por supuesto un componente es también un objeto).

E

Esquema: El esquema de una base de datos (en inglés, Database Schema) describe la estructura de una Base de datos, en un lenguaje formal soportado por un Sistema administrador de Base de datos (DBMS). En una Base de datos Relacional, el Esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.

F

Fron-end: El front-end es la parte del software que interactúa con el o los usuarios.

H

Herencia: La herencia es un tipo de relación entre una entidad “padre” y una entidad “hijo”. La entidad “hijo” hereda todos los atributos y relaciones de la entidad “padre”. Por tanto, no necesitan ser representadas dos veces en el diagrama.
Hardware: Corresponde a todas las partes tangibles de un sistema informático; sus componentes son: eléctricos, electrónicos, electromecánicos y mecánicos.

M

Multiplataforma: es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

P

Plugin: Un complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como **plugin** (del inglés "enchufable"), add-on (agregado), complemento, conector o extensión.

Proceso: Un **proceso** puede informalmente entenderse como un programa en ejecución. Formalmente un proceso es "Una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual, y un conjunto de recursos del sistemas asociados".

Q

Query: Consulta realizada contra una base de datos. Se usa para obtener datos, modificarlos o bien borrarlos.

S

Software: Se conoce como **software** *al equipamiento lógico o soporte lógico* de un sistema informático, comprende el conjunto de los componentes **lógicos** necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware.

R

Registro: En informática, o concretamente en el contexto de una base de datos relacionales, un **registro** (también llamado **fila** o **tupla**) representa un objeto único de datos implícitamente estructurados en una tabla. Un registro es un conjunto de campos que contienen los datos que pertenecen a una misma repetición de entidad.

T

Trigger: Un ***trigger*** (o disparador) en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación. Dependiendo de la base de datos, los triggers pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

Transacción: Una **transacción** en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Tablespace: Un tablespace es una unidad lógica de almacenamiento dentro de una base de datos. Es un puente entre el sistema de ficheros del sistema operativo y la base de datos.