Universidad de las Ciencias Informáticas.

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

Diseño de una Base de datos para la Suite de Ingeniería de Software.

Autor: *Iliana Aurora Tamayo Colas*

Tutor*: Ing. Yusleydi Fernández del Monte*

Ing. Vladimir Urquia Cordero

Ing. Mairim Delgado Muñiz

La Habana, Cuba, 2013

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Iliana Aurora Tamayo Colás

Firma del Autor

_____ _____

Ing. Yusleydi Fernández del Monte

Firma del Tutor

Ing. Vladimir Urguia Cordero

Firma del Tutor

_____

Ing. Mairim Delgado Muñiz

Firma del Tutor

“ES PRECISO SOÑAR, PERO CON LA CONDICIÓN DE CREER EN NUESTROS SUEÑOS. DE EXAMINAR CON ATENCIÓN LA VIDA REAL, DE CONFRONTAR NUESTRA OBSERVACIÓN CON NUESTROS SUEÑOS, Y DE REALIZAR ESCRUPULOSAMENTE NUESTRA FANTASÍA.”

LENIN

AGRADECIMIENTOS

Primero que todo le doy las gracias a dios y a la virgen por tener hoy a mi mamá conmigo, por darme la fuerza necesaria y poder seguir con mis estudios.

A mi mamá que es mi vida, gracias por ser como eres, por luchar para vivir por mí, para estar en estos momentos conmigo y verme triunfar.

A mi papá por guiarme siempre y apoyarme, por cuidar a mi mamá, por estar siempre ahí, gracias te quiero mucho aunque a veces no te lo diga.

A mi tía Ángela, mami Mary, Yeni, Gleydis, Yobi, Bernarda, a todos mis vecinos y maestros compañeros de mi mamá muchísimas gracias por apoyarme y cuidar de mi mamá mientras yo estaba estudiando.

A mi familia por parte de madre, por apoyarme y cuidar a mi mamá para que yo pudiera estudiar y llegar a este momento.

A María Chema y a Chelo muchísima gracias, este triunfo es gracias a su apoyo, su perseverancia, gracias por brindarme esa fuerza que tanto necesite, de todo corazón le doy la gracia por estar donde estoy hoy.

A mi tía Isabel, Alicia, mi tío Bárbaro gracias por sus atenciones para conmigo y su cariño.

A Lili gracias por soportarme todo este tiempo, por estar ahí siempre que lo necesite en los buenos y malos momentos, eres como mi hermana, gracias.

Y que sería de mí sin **mis compañeros**, a la Yuce, gracias por tu amistad, a Dayli, Nuri, los mellos, Arianne, Dennis, Oscar, Alexander, Merlyn, Daniel, Yilian, Samuel, Barban, Mayi, Made, Ramón, a todos gracias por su apoyo cuando lo necesite.

A los profe que me apoyaron para que pudiera seguir estudiando, Mirtha y especialmente a Gladys que ha sido una madre aquí en la universidad.

A mis compañeros de proyecto, especialmente Carlos, Ana y Johan gracias por su ayuda siempre que lo necesité.

A mis tutores Yusla, Vladimir, gracias por su ayuda, su apoyo, por enseñarme y guiarme, gracias por todo

A Lisi y Mairim gracias por su apoyo cuando lo necesité

A los profes Yaili, Eridniel y Yonelbys muchísimas gracias por su ayuda en el desarrollo de la tesis.

DEDICATORIA

A mi mamá, mi papá y mi abuelo Gerardo por ser las personas más importante en mi vida.

Iliana.

RESUMEN

La tesis que se presenta es el resultado de una investigación realizada por el proyecto Nova QALIT, perteneciente al centro de software libre (CESOL) de la Universidad de las Ciencias informáticas, en el mismo se desea implementar una Suite de Ingeniería de Software (ISW), herramienta Case[1] para el modelado, las pruebas, la generación de reportes y la automatización de la gestión de la información en un proceso de desarrollo de software desde su inicio hasta su terminación.

El tema seleccionado es de gran importancia por su contribución al desarrollo de la Suite lo cual permite una mejor gestión de la información en un proceso de desarrollo de software, por lo anterior se define como objetivo principal: Diseñar una base de datos (BD) que permita gestionar la información que generan los subsistemas de la Suite de Ingeniería de Software.

A partir del estudio realizado se comprueba que la Suite no presenta un mecanismo de integración entre el flujo de información que se genera. Por lo que la autora propone en esta investigación el diseño de una base de datos capaz de integrar toda la información generada por los subsistemas que componen la Suite, el modelo de datos obtenido en la solución está conformado por 41 tablas como resultado de la integración de todos los módulos de la Suite, el cual fue validado su rendimiento a través de pruebas de carga y estrés.

Palabras claves: Base de Datos, Suite de Ingeniería de Software (ISW), Modelo de datos, Arquitectura de software.

[1]**C**omputer **A**ided **S**oftware **E**ngineering; y en su traducción al español significa Ingeniería de Software Asistida por Computación.

ÍNDICE

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde que existe la humanidad, la necesidad de almacenar información ha sido una constante, en forma de dibujos en cuevas, filigranas sobre papiro, escritos en antiguos manuscritos, todo por preservar conocimientos. El escriba era un personaje admirado y respetado, sobre todo en el antiguo Egipto, ya que solía copiar textos, una ardua profesión que, sin duda, ha servido para que los conocimientos del pasado llegaran hasta la actualidad.

Con el desarrollo de las tecnologías han surgido programas que permiten almacenar y procesar grandes volúmenes de información, estos son los sistemas de base de datos, los cuales empezaron a tomar auge desde mediados de la década de los 50 y han ido evolucionando hasta convertirse en grandes almacenes de datos. También ha posibilitado una mayor eficiencia en el registro y procesamiento de la información, alcanzando altos estándares de estabilidad y permitiendo facilitar sustancialmente el trabajo humano.

Este factor constituye un elemento ineludible de la aplicación de las nuevas tecnologías de la informática y las comunicaciones donde los sistemas gestores de bases de datos juegan un papel primordial. Las operaciones manuales, con un alto costo en tiempo y recursos humanos se traducen actualmente en simples consultas que demoran apenas segundos. Esto supone una escalada en cuanto a la posibilidad de utilizar los datos en complejos procesos como la toma de decisiones.

Es por ello que un aspecto importante a tener en cuenta es la seguridad, pues un error en el desarrollo o administración de las bases de datos puede facilitar la intrusión de un atacante, por lo que se hace necesario el cumplimiento de tres aspectos básicos: confidencialidad, integridad y disponibilidad de la información. Además brindan una serie de ventajas entre las que se pueden apreciar: coherencia de resultados, independencia, tratamiento y mejora en la disponibilidad de datos, y la principal característica es que mantiene las propiedades ACID (Atomicidad, Consistencia, Integridad y Durabilidad de los Datos), permitiendo reducir la información duplicada.

Las bases de datos en la actualidad son las principales herramientas para el almacenamiento estructurado de datos, permiten realizar tareas como: modificaciones, actualizaciones, eliminaciones, respaldos de información y consultas de acuerdo a las necesidades para obtener información requerida.

La información de una base de datos se encuentra organizada de manera que en el momento de una consulta se pueda obtener la información clara y necesaria, sobre la cual puede realizarse nuevas

consultas para mejor detalle o realizar estadísticas sobre el historial. [1] Por ello es importante como se ha definido la arquitectura, siendo esto un aspecto fundamental, brinda una visión completa del sistema, un elemento relevante dentro de la arquitectura de software son las vistas, representan las diferentes perspectivas del diseño, permitiendo así corregir cualquier deficiencia antes de programar.

La arquitectura de software(AS) es una de las etapas principales del desarrollo de un software, es aquí donde los profesionales aportan todos sus conocimientos, creatividad y experiencia para crear la mejor propuesta de solución que se brinda al cliente que cumpla con los requisitos funcionales y no funcionales establecidos para el sistema en desarrollo. Al no existir estos planos que permitan guiar a los desarrolladores, se construye algo que viene a la imaginación justo en el momento de realizarlo. Por ello, también es importante que los diferentes interesados en el sistema se involucren en el diseño de la arquitectura pues, con ello, se acuerda de una mejor manera la solución a la que se llegue después de conocer los requisitos.

Para el correcto diseño de una base de datos se hace uso de herramientas que ofrecen a los diseñadores mayor facilidad y eficiencia a la hora de diseñar, permiten obtener una mayor calidad en la arquitectura. Ejemplo de estas herramientas son: ER/Studio, herramienta de modelado de datos, fácil de usar, multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico, equipado para crear y manejar diseños de bases de datos funcionales y confiables; Erwin, herramienta de diseño de base de datos, brinda productividad en el diseño, generación, y mantenimiento de aplicaciones, permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos; Visual Paradigm, es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software, presenta licencia gratuita y comercial, fácil de instalar, actualizar y compatible entre ediciones.

Además existen herramientas que brindan un soporte adecuado para el almacenamiento de información, estos son los Sistema de Gestores de Bases Datos (SGBD) que permiten almacenar y acceder a los datos de forma rápida y estructurada. Los SGBD relacionales están en plena transformación para adaptarse a tres tecnologías de éxito reciente, fuertemente relacionadas: la multimedia, la de orientación a objetos (OO), internet y la web. Actualmente los más utilizados son PostgreSQL, MySQL, Oracle, SQL Server.

En el centro CESOL dentro del Departamento Sistema Operativo se encuentra el proyecto Nova QALIT, el cual está inmerso en el desarrollo de una Suite de Ingeniería de Software (ISW), herramienta CASE, que debe permitir el modelado, las pruebas, la generación de reportes y la automatización de la gestión de la

información en un proceso de desarrollo de software desde su inicio hasta su terminación, no existe forma de guardar la información generada por la Suite de Ingeniería de Software.

Teniendo en cuenta que los subsistemas que la integran deben permitir registrar la misma información por vías diferentes, los datos almacenados deben poderse utilizar por cualquiera de los módulos existentes, y no hay un mecanismos de integración entre todo el flujo de información que ocurre en la Suite de Ingeniería de Software. No contando con la arquitectura de software del sistema, por lo que hay grandes posibilidades de construir un sistema que no alcanzará el total de los requisitos establecidos, generando mayor trabajo o, peor aún, puede llevar al fracaso del sistema cuando se encuentre en operación.

Por tanto se identifica como **problema científico**: ¿Cómo estructurar la información que generan los subsistemas de la Suite de Ingeniería de Software, en una Base de Datos?

En consecuencia con lo anterior se determinó como **objeto de estudio**: proceso de diseño de las Bases de Datos y como **campo de acción**: proceso de diseño para la base de datos de la Suite de Ingeniería de Software.

El **objetivo general** de este trabajo de diploma es: diseñar una base de datos que permita gestionar la información que generan los subsistemas de la Suite de Ingeniería de Software.

Para garantizar el cumplimiento del objetivo general, se definen los siguientes **objetivos específicos:**

- Realizar un estudio sobre las tendencias de las arquitecturas de bases de datos.
- Diseñar la base de datos de la Suite de Ingeniería de Software.
- Validación del diseño propuesto.

Para dar cumplimiento a los objetivos planteados, se definieron como **tareas de investigación**:

- Realización de un estudio de las arquitecturas de bases de datos que permiten el almacenamiento de información generada por una herramienta de escritorio para determinar los mecanismos que se utilizan con este fin.
- Estudio de patrones de diseño de base de datos para determinar las mejores formas de estructurar una base de datos.
- Caracterización de arquitecturas de herramientas CASE que utilizan bases de datos como

medio de almacenamiento de su información para identificar las mejores prácticas.

- Diseño de la base de datos para almacenar la información que se genera en la Suite de Ingeniería de Software.
- Evaluación del diseño propuesto de la Suite de Ingeniería de Software, para verificar si es correcta.

Para guiar la investigación se plantea como **idea a defender** el diseño correcto de la base de datos de la Suite de Ingeniería de Software permitirá definir la estructura de la gestión de la información generada por los subsistemas que la componen.

A continuación se muestran los **métodos científicos** utilizados para el desarrollo de esta investigación:
Métodos teóricos: Estos métodos permiten estudiar las características del problema que no son observables directamente.

- **Histórico-Lógico**: Permite realizar un estudio para analizar cómo ha sido la evolución de las arquitectura de bases de datos y las formas de representarla, sus antecedentes, revelando las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.
- **Modelación**: Permite modelar el diseño que va a tener la base de datos de la Suite de Ingeniería de Software.

Métodos Empíricos: Estos métodos permiten revelar las relaciones esenciales y las características del objeto de estudio.

- **Experimental**: Se ve reflejado a la hora de realizar las pruebas al diseño de base de datos que se propone para validar que es correcto.

El presente trabajo de diploma se encuentra estructurado en tres capítulos:
Capítulo 1: Fundamentación teórica de las Bases de Datos. En este capítulo se exponen los resultados de la investigación realizada sobre los principales aspectos a tener en cuenta en el diseño de la arquitectura de una base de datos.

Capítulo 2: Diseño de la Base de Datos para la Suite de Ingeniería de Software. Se presenta una propuesta del diseño de la base de datos, capaz de gestionar la información que generan los subsistemas de la Suite de Ingeniería de Software

Capítulo 3: Validación del diseño propuesto para la Suite de Ingeniería de Software. En este capítulo se realizan las pruebas necesarias para determinar si el diseño de la base de datos definida cumple con los requisitos especificados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LAS BASES DE DATOS

1.1 Introducción

En el presente capítulo se aborda brevemente sobre los aspectos a tener en cuenta en el proceso de desarrollo de una base de datos, principales conceptos, evolución y ejemplos de los sistemas gestores de base de datos. Se realiza un estudio sobre los patrones aplicable a las base de datos, características de las herramientas y metodología de desarrollo, definiéndose el SGBD a utilizar y la justificación de su elección.

1.2 Definición de base de datos

El término de base de datos es muy utilizado en la actualidad, principalmente por los desarrolladores de software, el mismo presenta disímiles definiciones por lo que se hace necesario conceptualizar por varios autores.

- Una base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos [2].
- Conjunto de información relacionada y organizada que se encuentra recopilada en dispositivos de almacenamiento, teniendo como principal objetivo almacenar grandes cantidades o volúmenes de información siguiendo un determinado esquema o modelo de datos.[3]
- Según García 1999 define que es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo.

1.3 Tipos de bases de datos

Las bases de datos desde su surgimiento han tenido una vertiginosa evolución, estas se clasifican de varias formas atendiendo a múltiples criterios que se han definido como:

1. **Variabilidad de los datos almacenados**

Atendiendo a este criterio se pueden clasificar en:

Bases de datos estáticas:

Son consideradas como sólo lectura[2], ya que son usadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

Bases de datos dinámicas:

Las bases de datos dinámicas permiten que la información almacenada se modifique con el tiempo, permitiendo operaciones como: actualización, inserción y eliminación de datos, además de las operaciones fundamentales de consulta.

2. **Contenido**:

Atendiendo a este criterio se pueden clasificar en:

Bases de datos bibliográficas:

Son base de datos que puede contener un resumen o extracto de la publicación original, pero nunca el texto completo. Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

Bases de datos de texto completo:

Denominadas también directorios, son aquellas cuyo contenido está referido a la descripción de otros recursos de información, son bases de datos que almacenan las fuentes primarias[3], como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

1.4 Usuarios que interactúan con las bases de datos.

Entre los usuarios de una base de datos podemos distinguir:

- **Finales:** son aquellos que interactúan con los datos normales del sistema de información. Un usuario final no modifica la estructura de la base de datos. Puede tener permiso para insertar, modificar o eliminar datos, siendo la operación clásica la consulta. Este usuario no puede modificar la estructura de una tabla. Pueden ser usuarios sin conocimientos informáticos, tan solo manejan un programa, también pueden saber SQL y hacer una consulta a través de un intérprete de SQL.
- **Programador de aplicación:** es un usuario con conocimientos informáticos con la responsabilidad de escribir código para la aplicación de gestión. Habitualmente tiene la posibilidad de crear

[2]De **solo lectura** se refiere a que la información almacenada no se puede modificar, actualizar o eliminar.
[3]Las **fuentes primarias** contienen información nueva y original, resultado de un trabajo intelectual.

subesquemas en la base de datos y por tanto se le permite modificar y estructurar la base de datos. Tiene permiso para crear código.

- **Administrador (DBA):** Es el usuario con permisos más alto de la base de datos y tiene la responsabilidad de mantener el funcionamiento de la base de datos, definir todos los parámetros de inicialización y almacenamiento. Crea usuarios y les otorga los permisos pertinentes. Tiene que definir la forma de recuperar la base de datos y todas las políticas de seguridad.[4]

1.5 Tipos de modelos de datos

Un modelo de datos es un conjunto de conceptos y reglas que permiten describir el mundo real, se utilizan para representar la estructura de datos y las relaciones entre ellos dentro de la base de datos, es una abstracción que permite la implementación de sistemas eficientes. Estos se clasifican en diferentes tipos:

Modelo de datos jerárquico

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se les conoce como hojas. Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

Modelo de datos en red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Modelo de datos relacional

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tarda en consolidarse como un nuevo paradigma en los modelos de

datos. Su idea fundamental es el uso de "relaciones". Estas relaciones pueden considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se define de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

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

Modelo de datos orientada a objeto

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

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

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

1.6 Sistemas gestores de bases de datos

El Sistema de Gestión de Bases de Datos (SGBD), denominado en ingles *Data Base Management System* (DBMS) es un paquete de software, que se ejecuta en un ordenador anfitrión que es quien centraliza los accesos a los datos y actúa de interfaz entre los datos almacenados y los usuarios [5].

1.6.1 Objetivos principales de los Sistemas gestores de bases de datos

Consultas no predefinidas y complejas: Los usuarios pueden hacer consultas de cualquier tipo y complejidad directamente al SGBD, este tiene que responder inmediatamente sin que estas consultas estén preestablecidas; es decir, sin que se tenga que escribir, compilar y ejecutar un programa específico

para cada consulta.

Flexibilidad e independencia: Interesa obtener la máxima independencia posible entre los datos y los procesos de los usuarios, para que se pueda llevar a cabo todo tipo de cambios tecnológicos y variaciones en la descripción de la base de datos, sin que se deban modificar los programas de aplicación ya escritos ni cambiar la forma de escribir las consultas (o actualizaciones) directas.

Problemas de la redundancia: Facilitar la eliminación de la redundancia. El SGBD debe permitir que el diseñador defina datos redundantes, pero entonces tiene que ser el mismo gestor el que realice automáticamente la actualización de los datos en todos los lugares donde estén repetidos.

Integridad de los datos: Es necesario que los SGBD aseguren el mantenimiento de la calidad de los datos en cualquier circunstancia. Se podría perder la corrección o la consistencia de los datos por muchas otras razones: errores de programas, errores de operación humana, avería de disco, transacciones incompletas por corte de alimentación eléctrica, etc.

Concurrencia de usuarios: Permitir que varios usuarios puedan acceder concurrentemente a la misma base de datos.

Seguridad: En el campo de los SGBD, el término seguridad se suele utilizar para hacer referencia a los temas relativos a la confidencialidad, las autorizaciones, los derechos de acceso, etc.

A medida que los SGBD evolucionan, se imponen nuevos objetivos adaptados a las nuevas necesidades y las nuevas tecnologías.

- Servir eficientemente los almacenes de datos, denominado en ingles *Data Warehouse*.
- Adaptarse al desarrollo orientado a objetos.
- Incorporar el tiempo como un elemento de caracterización de la información.
- Adaptarse al mundo de Internet. [6]

1.6.2 Lenguaje que soporta el Sistema Gestor de Base de Datos

Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales. Los lenguajes van a permitir al administrador especificar los datos que componen la base de datos, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes del SGBD se clasifican en:

- **Lenguaje de definición de datos (LDD o DDL):** se utiliza para especificar el esquema de la base de datos, las vistas de los usuarios y las estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la base de datos.
- **Lenguaje de manipulación de datos (LMD o DML):** se utilizan para leer y actualizar los datos de la base de datos. Es utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones. Los hay procedurales, en los que el usuario es normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un LMD procedural están embebidas en un lenguaje de alto nivel llamado anfitrión. Las bases de datos jerárquicas y en red utilizan estos LMD procedurales.

No procedurales son los lenguajes declarativos. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde un terminal, también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta, o los datos a modificar, mediante sentencias sencillas. Las bases de datos relacionales utilizan lenguajes no procedurales como SQL (*Structured Quero Language*) o QBE (*Query By Example*).

La mayoría de los SGBD comerciales incluyen **lenguajes de cuarta generación (4GL)** que permiten al usuario desarrollar aplicaciones de forma fácil y rápida, también se les llama herramientas de desarrollo. Ejemplos de esto son las herramientas del SGBD ORACLE: SQL *Forms* para la generación de formularios de pantalla y para interactuar con los datos; SQL *Reports* para generar informes de los datos contenidos en la base de dato; PL/SQL lenguaje para crear procedimientos que interactúen con los datos de la base de dato. [7]

1.6.3 Componentes de los Sistemas Gestores de Bases de Datos

Los SGBD son paquetes de software muy complejos y sofisticados, no se puede generalizar sobre los elementos que componen un SGBD ya que varían mucho unos de otros. Sin embargo, es muy útil conocer sus componentes y cómo se relacionan cuando se trata de comprender lo que es un sistema de bases de datos.

Un SGBD tiene varios módulos, cada uno de los cuales realiza una función específica. El sistema

operativo proporciona servicios básicos al gestor, que es construido sobre él.

- El procesador de consultas es el componente principal de un SGBD. Transforma las consultas en un conjunto de instrucciones de bajo nivel que se dirigen al gestor de la base de datos.
- El gestor de la base de datos es el interfaz con los programas de aplicación y las consultas de los usuarios. El gestor de la base de datos acepta consultas y examina los esquemas externo y conceptual para determinar qué registros se requieren para satisfacer la petición. Entonces el gestor de la base de datos realiza una llamada al gestor de ficheros para ejecutar la petición.
- El gestor de ficheros maneja los ficheros en disco en donde se almacena la base de datos. Este gestor establece y mantiene la lista de estructuras e índices definidos en el esquema interno. Si se utilizan ficheros dispersos, llama a la función de dispersión para generar la dirección de los registros. Pero el gestor de ficheros no realiza directamente la entrada y salida de datos. Lo que hace es pasar la petición a los métodos de acceso del sistema operativo que se encargan de leer o escribir los datos en el buffer del sistema.
- El preprocesador del LMD convierte las sentencias del LMD embebidas en los programas de aplicación, en llamadas a funciones estándar escritas en el lenguaje anfitrión. El preprocesador del LMD debe trabajar con el procesador de consultas para generar el código apropiado.
- El compilador del LDD convierte las sentencias del LDD en un conjunto de tablas que contienen metadatos. Estas tablas se almacenan en el diccionario de datos.
- El gestor del diccionario controla los accesos al diccionario de datos y se encarga de mantenerlo. La mayoría de los componentes del SGBD acceden al diccionario de datos.
- Control de autorización. Este módulo comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita.
- Procesador de comandos. Una vez que el sistema ha comprobado los permisos del usuario, se pasa el control al procesador de comandos.
- Control de la integridad. Cuando una operación cambia los datos de la base de datos, este módulo debe comprobar que la operación a realizar satisface todas las restricciones de integridad necesarias.
- Optimizador de consultas. Este módulo determina la estrategia óptima para la ejecución de las consultas.
- Gestor de transacciones. Este módulo realiza el procesamiento de las transacciones.

- Planificador. Este módulo es el responsable de asegurar que las operaciones que se realizan concurrentemente sobre la base de datos tienen lugar sin conflictos.
- Gestor de recuperación. Este módulo garantiza que la base de datos permanezca en un estado consistente en caso de que se produzca algún fallo.
- Gestor de buffers. Este módulo es el responsable de transferir los datos entre memoria principal y los dispositivos de almacenamiento secundario. A este módulo también se le denomina gestor de datos. [8]

1.6.4 Características de algunos Sistemas Gestores de Bases de Datos

Los SGBD prestan servicios para el desarrollo y el manejo de bases de datos, en la actualidad existe diversos, por lo que es complejo definir cuál es el mejor, dentro de los más usados tenemos:

ORACLE

Es un sistema de base de datos relacional, potente herramienta cliente/servidor para la gestión de Bases de Datos. Es el conjunto de datos que proporciona la capacidad de almacenar y acudir a estos de forma recurrente con un modelo definido como relacional.

VENTAJAS

- Oracle es la base de datos con más orientación hacía internet.
- Soporta todas las funciones que se esperan de un servidor serio: un lenguaje de diseño de bases de datos muy completo (PL/SQL) que permite implementar diseños activos, con disparadores y procedimientos almacenados, con una integridad referencial declarativa bastante potente.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- El software del servidor puede ejecutarse en multitud de sistemas operativos.

DESVENTAJAS

- El precio, incluso las licencias de Personal Oracle son excesivamente caras.
- Un error frecuente consiste en pensar que basta instalar el Oracle en un servidor y enchufar directamente las aplicaciones clientes.
- Un Oracle mal configurado puede ser muy lento.
- También es elevado el coste de la formación, y sólo últimamente han comenzado a aparecer

buenos libros sobre asuntos técnicos distintos de la simple instalación y administración.

MYSQL

Es un sistema de administración de bases de datos para bases de datos relacionales, licenciado bajo la GPL de la GNU, su diseño multihilo le permite soportar una gran carga de forma muy eficiente, escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl, Java y su integración en distintos sistemas operativos.

VENTAJAS

- El servidor de bases de datos relacionales MySQL es muy rápido, fiable y fácil de usar.
- Buena velocidad a la hora de conectar con el servidor y de respuesta a consultas.
- Posee un buen control de acceso de usuarios y seguridad en los datos.
- Soporte completo para cláusulas, funciones, tipos de datos, comandos estándar y extendidos del estándar SQL.

DESVENTAJAS

- Actualmente, el soporte para disparadores es básico, por lo tanto, hay ciertas limitaciones en lo que puede hacerse con ellos.
- Los privilegios para una tabla no se eliminan automáticamente cuando se borra una tabla. Debe usarse explícitamente un comando *REVOKE* para quitar los privilegios de una tabla.
- Cuando MySQL maneja la integridad referencial, con tablas “NO’’ transaccionales de tipo *MyISAM*, aunque admite la declaración de claves ajenas o foráneas en la creación tablas, internamente no las trata de forma diferente al resto de los campos.

SQL SERVER

Conjunto de objetos eficientemente almacenados. En el centro de SQL Server está el motor de SQL Server, el cual procesa los comandos de la base de datos. Los procesos se ejecutan dentro del sistema operativo y entienden únicamente de conexiones y de sentencias SQL, SQL Server incluye herramientas para la administración de los recursos que el ordenador nos proporciona y los gestiona para un mejor rendimiento de la base de datos.

VENTAJAS

- El atractivo principal: lo barato del sistema, y la tendencia de los directivos a aceptar preferentemente productos de Microsoft.
- Otro punto importante a favor de SQL Server es la interfaz de acceso OLE DB y ADO. Aunque se trata de una interfaz universal, SQL Server es una de las primeras bases de datos en soportarla.
- Mejor utilización de la CPU.
- Menor necesidad de limpieza de las memorias intermedias durante el procesamiento de las transacciones.

DESVENTAJAS

- La principal desventaja de Microsoft SQL SERVER es la enorme cantidad de memoria RAM que utiliza para la instalación y utilización del software.
- La relación calidad-precio está muy debajo comparado con Oracle.
- Usa *Address Windowing* extensión (AWE) para hacer el direccionamiento de 64-bit esto le impide usar la administración dinámica de memoria y sólo permite alojar 64Gb de memoria compartida.
- Sólo permite aproximadamente 16 instancias distintas concurrentes en una máquina.
- No maneja compresión de datos por tanto ocupa mucho espacio en disco.
- Está atado a la plataforma de sistema operativo sobre la cual puede instalarse.[9]

POSTGRESQL

Es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo licencia PostgreSQL basada en la BSD, programa de código abierto, por lo que está dirigido por una comunidad de desarrolladores llamada PGDG (*PostgreSQL Global Development Group*). Entre sus principales características se encuentra, la alta concurrencia, la amplia variedad de tipos nativos, y diversas funciones más específicas.

VENTAJAS:

- Instalación ilimitada: No se puede demandar a una empresa por instalarlo en más ordenadores de los que la licencia permite, ya que no hay costo asociado a la licencia de software.
- Ahorros considerables de costos de operación: PostgreSQL ha sido diseñado para tener un mantenimiento y ajuste menor que los productos de proveedores comerciales, conservando todas las características, estabilidad y rendimiento.
- Estabilidad y confiabilidad: No se han presentado caídas de la base de datos.

- Extensible: El código fuente está disponible de forma gratuita, para que quien necesite extender o personalizar el programa pueda hacerlo sin costes.
- Multiplataforma: Está disponible en casi cualquier Unix, con 34 plataformas en la última versión estable, además de una versión nativa de Windows.
- Diseñado para ambientes de alto volumen: Utilizando una estrategia de almacenamiento de filas llamada MVCC (*Multi-Version Concurrency Control*), consigue mejor respuesta en grandes volúmenes de información. Además, MVCC permite a los accesos de solo lectura continuar leyendo datos consistentes durante la actualización de registros, permitiendo copias de seguridad en caliente.
- Herramientas gráficas de diseño y administración de bases de datos.
- Buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas.
- Gran capacidad de almacenamiento.
- Buena escalabilidad ya que es capaz de ajustarse a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.
- Soporta replicación de bases de datos asíncrona, realizando primero las transacciones en un "servidor maestro" para que se puedan actualizar en los "servidores esclavos" dando alta disponibilidad al sistema

DESVENTAJAS:

- Soporte en línea: está divido entre libre(lista de correos) y comercial
- La sintaxis de algunos de sus comandos o sentencias no es nada intuitiva. .[10]

1.7 Arquitectura de Base de Datos

1.7.1 Niveles de abstracción de las Bases de Datos

Las bases de datos cuentan con 3 niveles de abstracción:

- **Externo**: Es el nivel de los datos que tienen los usuarios finales de una base de datos. Un usuario trata sólo una visión parcial de la información, sólo aquella que interviene en el dominio de su actividad. Se corresponde con los Esquemas o Subesquemas Externos que tiene cada usuario de la base de datos.
- **Conceptual**: Es la representación abstracta del problema. Una base de datos representa la

información de un problema del mundo real. Es independiente de: cómo va a ser tratada la información, las visiones externas que tenga, cómo va almacenarse esta información físicamente. Se corresponde con el Esquema Conceptual.

- **Físico:** Es la representación de cómo la información es almacenada en los dispositivos de almacenamiento. Describe las estructuras u organizaciones físicas, dispositivos, ficheros, tipos de datos, etc. Se corresponde con el Esquema físico.

1.7.2 Patrones arquitectónicos para base de datos.

Los patrones arquitectónicos son una guía, un modelo a seguir que ayuda a resolver un problema, muchos autores han brindado diferentes definiciones algunas de ellas son:

- "Un patrón de arquitectura de software describe un problema particular de diseño recurrente en contextos específicos y presenta un esquema genérico probado para su solución."[Buschmann-1996]
- "Un patrón es una plantilla. Esta plantilla se utiliza para dar solución a un problema recurrente de la realidad. Es una plantilla para una solución, no una solución en sí. [Coad-1994]
- "Un patrón proporciona una solución probada a un problema común, documentada en un formato coherente." [Erl-2009]

Los patrones de diseño tienen varias clasificaciones, los aplicables a las bases de datos se clasifican en:

Árboles

Árboles fuertemente codificados: En este caso a cada nivel del árbol se le asocia una entidad. Normalmente constituyen relaciones de 1 a muchos (n). Utilizado para representar jerarquías donde es bien conocida la estructura y es importante representar la correspondencia, por ejemplo las estructuras organizacionales. Es importante señalar que este patrón debe utilizarse sólo en los casos en que los cambios en la estructura a representar sean poco probables. Así como aclarar que el patrón admite tantos niveles como requiera la jerarquía que se vaya a representar. Un ejemplo práctico y aplicado al contexto de la Universidad, puede ser un árbol fuertemente codificado para almacenar la estructura organizativa de las facultades. Donde en el Nivel1 se encuentren las Facultades, en el Nivel2 los Centros y en el Nivel 3 los Proyectos.

Árboles simples: Patrón normalmente utilizado cuando el árbol es la representación de una estructura de datos. Los elementos a almacenar son del mismo tipo, es decir, pueden ser almacenados en la misma

entidad. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre. Un ejemplo práctico del uso del patrón lo constituye la representación de la relación Jefe-Subordinado en una empresa, donde se conoce que las personas pueden tener o no un jefe superior y que un jefe puede tener varios subordinados. En este caso se garantiza que una persona nunca es subordinada de ella misma.

Árbol estructurado: Este modelo es usado cuando se necesitan diferenciar los nodos hojas, de aquellos que generan una nueva rama, porque ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre. La generalización tiene cubrimiento total y exclusivo, cada elemento de la Entidad Nodo, debe tener su correspondiente elemento en la entidad hoja o en la entidad rama. Un ejemplo de aplicación práctica de este patrón lo constituye la representación de un directorio de ficheros, en este caso un fichero puede ser o un fichero de datos o un subdirectorio que dentro tiene otros ficheros. Se representa la condición de que cada fichero puede tener un subdirectorio que lo contiene y cada subdirectorio puede contener muchos ficheros dentro.

Grafos

Grafo dirigido simple: Se utiliza cuando todos los nodos contienen el mismo tipo de datos. Es similar al árbol simple, la diferencia es que en este caso la relación recursiva sobre Nodo tiene cardinalidad de muchos a muchos y por tanto se genera una nueva entidad. Por ejemplo, este patrón se puede utilizar para representar la relación entre dirigentes y subordinados en los contextos donde cada subordinado puede tener varios jefes y cada jefe tiene varios subordinados.

Grafo dirigido estructurado: Este modelo es usado cuando se necesita diferenciar los nodos hojas, de aquellos que generan una nueva rama, porque ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. Es similar al árbol estructurado, la diferencia es que en este caso la relación recursiva sobre Nodo tiene cardinalidad de muchos a muchos. Una aplicación práctica para este patrón sería la representación de las listas de contactos en un servicio web de correo electrónico, donde se permita la creación de listas anidadas, así como incluir a un mismo contacto o una misma lista dentro de varias listas.

Patrón de llaves subrogadas:

Este patrón es muy utilizado pues facilita la interacción con la base de datos en un futuro. El mismo plantea que se genere una llave primara única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Normalmente se usan enteros en columnas identidad o GUID (*Global Unique*

Identifier) que están demostradas que no se repiten o con una probabilidad extremadamente baja.
Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas. Ejemplo si se tiene una entidad Solicitud-préstamo, referente a una solicitud de un libro en la biblioteca, un identificador de un préstamo debería estar constituido por el título del libro, el cliente, la hora y la fecha. Sin embargo aplicando este patrón se adicionaría un atributo id de tipo autoincremental, que sería la llave primaria para esta entidad, en vez de la combinación de todos aquellos atributos.

Patrones para flujos de trabajo aplicables a las base de datos

Máquinas de estado para un tipo de entidad: representa los posibles cambios de estado por los que puede atravesar un tipo de entidad, no refleja el almacenamiento de las ocurrencias sino las propiedades del flujo de trabajo, por lo tanto esto es el diseño del diagrama de flujo. Un ejemplo de la utilización de este patrón sería precisamente si se quiere representar en una base de datos el flujo de actividades desde que un cliente pide una orden hasta que termina el proceso dentro de un bar o restaurante.

Máquina de estado para escenarios: representa la ocurrencia del cambio de estado en un escenario de una entidad dada, por lo tanto considera el tiempo y la persistencia del mismo en las tablas resultantes. También representa la ocurrencia de un estímulo en una fecha y los estados por los que ha pasado, caracterizados por la fecha de inicio y la fecha fin, permite guardar la información específica de un flujo en un momento determinado, por ejemplo permite almacenar los datos de las diferente órdenes de un cliente en un bar, teniendo en cuenta la aplicación práctica vista para el patrón anterior.

Modelo Entidad-Atributo-Valor:

El patrón entidad-atributo-valor es la representación de un modelo flexible donde se pueden representar objetos con sus atributos. Es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad *Class* representa las clases, la entidad *Attribute* representa los atributos de las clases, por su parte la entidad *Object* representa las instancias de las clases, mientras que la entidad *Value* representa los valores de cada atributo para cada objeto dado. Un ejemplo del modelo entidad-atributo-valor se puede presentar en el almacenamiento de documentos captados a través de un sistema, que como pueden ser disímiles y surgen nuevos a cada momento, deben almacenarse en una estructura flexible.

Patrón para la seguridad de las aplicaciones aplicable a las base de datos.

Patrón Control de acceso basado en roles (RBAC): En una organización, los roles son creados para varias funciones de trabajo. Los permisos para ejecutar ciertas operaciones son asignados a roles específicos. Se les asigna roles particulares a los miembros del equipo (o a otros usuarios del sistema), y a través de esos roles asignados obtienen permiso para ejecutar funciones determinadas en el sistema. Como a los usuarios no se les asigna permisos directamente, sino que los adquieren a través de su rol (o roles), el manejo de los permisos de cada usuario se convierte en una cuestión de simplemente asignar los roles apropiados al usuario, esto simplifica las operaciones comunes, como adicionar un usuario, o cambiar el departamento de un usuario.

Se definen tres reglas primarias para el modelo RBAC:

- Asignación de roles: Un usuario puede ejecutar una transacción sólo si se le ha asignado un rol.
- Autorización de roles: El rol activo de un usuario debe estar autorizado para dicho usuario. Esta regla, junto con la 1, asegura que los usuarios puedan tener solo los roles que se les han autorizado.
- Autorización de transacciones: Un usuario puede ejecutar una transacción solo si esta está autorizada para el rol activo de dicho usuario. Con las reglas 1 y 2, esta regla asegura que los usuarios solo puedan ejecutar las transacciones que se les han autorizado.[11]

1.8 Infraestructura tecnológica para el desarrollo de la Base de Datos.

La infraestructura tecnológica, son las condiciones técnicas que fueron tomadas en el proyecto para el desarrollo de la investigación.

1.8.1 Metodología de desarrollo

La metodología de desarrollo es una colección de políticas y procedimientos que intervienen en el desarrollo del software, cuya finalidad es garantizar la eficacia en el proceso de generación de software. [12]

OpenUp

La metodología de desarrollo a utilizar es OpenUp, metodología ágil que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software, se enfoca en la naturaleza colaborativa del desarrollo de software, apropiado para proyectos pequeños y de bajos recursos.

OpenUp cuenta con 4 fases Inicio, Elaboración, Construcción y Transición en el presente trabajo solo se

llega hasta elaboración porque el alcance del mismo solo abarca la definición y evaluación de la arquitectura.

La fase de Inicio como la primera de las cuatro fases, busca comprender el alcance del proyecto y sus objetivos y también obtener suficiente información para confirmar que el proyecto debe continuar o convencer que debe cancelarse. El propósito es alcanzar un acuerdo entre todos los *stakeholders* sobre los objetivos del proyecto.

Hay cuatro objetivos en esta fase que clarifican el alcance, objetivos del proyecto y factibilidad de la solución pretendida:

- Entender lo que se va a construir. Determinar una visión general, incluyendo alcance del sistema y sus límites. Identificar *stakeholders*.
- Identificar las funcionalidades claves del sistema. Decidir qué requisitos son más críticos.
- Determinar por lo menos una posible solución. Evaluar si la visión es técnicamente factible. Esto puede involucrar identificar una arquitectura candidata de alto nivel o realizar prototipos técnicos, o ambas cosas.
- Entender a un alto nivel la estimación de costos, calendario y riesgos asociados al proyecto.

En la segunda fase Elaboración es donde se tienen en cuenta los riesgos estructuralmente significativos. El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proveer una base estable para la mayor parte del esfuerzo de desarrollo de la siguiente fase.

Hay tres objetivos que ayudan a tratar los riesgos asociados con los requisitos, arquitectura, costos y calendario:

- Obtener un entendimiento más detallado de los requisitos. Asegurarse de tener un conocimiento profundo de los requisitos más críticos.
- Diseñar, implementar, validar y establecer la línea base de la arquitectura (para el esqueleto de la estructura del sistema).
- Mitigar riesgos esenciales y producir un calendario apropiado y estimación de costos.

Esencialmente, los principales objetivos para la elaboración están relacionados con el mejor entendimiento de los requerimientos, creando y estableciendo una línea base de la arquitectura para el sistema, y mitigando los riesgos de mayor prioridad. [13]

La presente metodología cuenta con diferentes disciplinas, las mismas presentan varias actividades a realizar generando diferentes artefactos, de estas se desarrollan en el presente trabajo: Requerimientos y Arquitectura, además de las disciplinas de apoyo Administración de la configuración, Administración de proyecto.

Los artefactos generados son a partir de la combinación entre la metodología y los propuestos en el programa de mejora.

Tabla: 1 Artefactos generados

Disciplinas	Tareas	Artefactos
Requerimientos	• Identificar y resaltar requerimientos • Detallar Escenarios de Caso de Uso • Detallar requerimientos generales del sistema • Desarrollar una visión técnica	• Glosario de términos • Especificación de requisitos • Descripción de los requisitos
Arquitectura	• Refinar la arquitectura • Visualizar (Prever) la arquitectura	• Vista de infraestructura tecnológica • Vista de Datos • Vista de Seguridad • Vista de entorno de desarrollo tecnológico • Vista de Integración
Administración de Proyecto	• Evaluar resultados • Administrar La iteración • Planear la iteración • Planear el proyecto • Solicitar cambios	• Plan de desarrollo de software • Proyecto técnico
Administración de la configuración y cambio		• Plan de gestión de la configuración de software(va dentro del plan de desarrollo de software)

1.9 Herramientas CASE para el diseño de la base de datos

Se puede definir a las Herramientas CASE como:

- Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software.
- Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.
- Unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.[14]

Herramienta Case para el diseño de la Base de datos de la Suite de Ingeniería de Software.

En la actualidad existen varias herramientas para el diseño de base de datos que facilitan el trabajo a la hora de diseñar, permitiendo obtener modelos más completos y robustos ejemplo de estas son Visual Paradigm y ER/Studio.

Visual Paradigm

Herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación.

Entre sus principales funcionalidades:

- Modelado de base de datos: proporciona una mayor documentación de la base de datos y diagramas de mapeo de relación de objetos.
- Mapa de relación de objetos.
- Generador de código.
- Soporte ORM, generación de objetos Java desde la base de datos.
- Generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Beneficios que ofrece:

- Demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente
- Superior entorno de modelado visual
- Diagramas de diseño automático sofisticado.

1.10 Caracterización de la suite objetivo

La Suite de Ingeniería de Software es una herramienta CASE que permite el modelado de software, la automatización de la gestión de la información del proceso de desarrollo de software desde su inicio hasta su terminación, la medición e interpretación de los procesos, así como las pruebas a productos y generación de reportes. La misma está compuesta por diferentes subsistemas:

Módulo Principal: Integra el funcionamiento de todos los módulos a la vista de los usuarios, garantizando que estos tengan los privilegios adecuados según su rol, y permitiendo la gestión general de un proyecto determinado.

Módulo Base de Datos: Constituye el núcleo de almacenamiento del sistema pues tiene la tarea fundamental de almacenar e integrar toda la información que va a ser procesada en el restos de los módulos.

Módulo de Pruebas: El propósito fundamental de un proceso de pruebas es encontrar la mayor cantidad de errores y no conformidades en las primeras versiones funcionales de un software, lo que permite corregirlos antes de liberarlo, incidiendo directamente en la calidad y el prestigio del producto. El sistema de pruebas de software pretende automatizar el proceso de pruebas de un software, principalmente la evaluación de los resultados y la generación de los informes relacionados con este proceso. En primera instancia se especializará en el proceso que llevan a cabo las distribuciones GNU/Linux y posteriormente se ampliará hacia un proceso más genérico para todo tipo de software.

Módulo de Gestión de la Información del proceso de desarrollo de software: Conjunto de funcionalidades que permiten guardar el historial de un proceso de desarrollo de software en productos de trabajo predefinidos, así como las planificaciones y los resultados que se van obteniendo de ejecutarlas, elementos que sirven de evidencias para la toma de decisiones en el proyecto y en otros similares.

Módulo de Generación de Reportes: Reportes que permiten socializar los resultados de un proyecto extrayendo la información necesaria del sistema y almacenarla en documentos o .pdf para lograr un mejor

intercambio y divulgación de la misma.

Módulo de Modelado de Software: Permite modelar software teniendo en cuenta todas las perspectivas desde las que se puede analizar este tipo de producto, para esto brinda opciones para utilizar la notación UML, así como para detallar los negocios de las organizaciones a través de BPMN, se pueden representar además modelos de las vistas lógicas y físicas de las bases de datos y la trazabilidad de los requisitos. [15]

1.11 Consideraciones finales del capítulo

Una vez concluido este presente capítulo, el análisis realizado sobre principales conceptos y aspectos de las bases de datos y los sistemas gestores de bases de datos, brindaron un mayor conocimiento y dominio sobre el campo de acción en que se desarrolla la solución que se propone. Por otra parte fue realizada la selección del modelo de datos relacional, el SGBD PostgreSQL y Visual Paradigm como herramientas para realizar diseño de la base de datos que se propone. De esta manera fueron sentadas las bases teóricas para darle solución al objetivo planteado en la presente investigación.

CAPÍTULO 2: DISEÑO DE LA BASE DE DATOS PARA LA SUITE DE INGENIERÍA DE SOFTWARE

2.1 Introducción

En el presente capítulo se describe la solución propuesta, se presentan los requisitos funcionales del sistema, el diagrama físico que se ha generado, se describen las tablas fundamentales y se justifica la utilización de patrones.

2.2 Requisitos Funcionales de la base de datos.

RF_1. Gestionar Metodología de desarrollo de software. Debe permitir adicionar, eliminar, modificar o mostrar metodología.

RF_1.1 Adicionar metodología, de la misma se registran los siguientes datos:

- Nombre
- Fases
- Disciplinas
- Roles

RF_1.2 Eliminar metodología.

RF_1.3 Modificar metodología, los aspectos a modificar son:

- Nombre
- Fases
- Disciplinas
- Roles

RF_1.4 Mostrar metodología, permite mostrar los datos de la misma, estos son:

- Nombre
- Fases
- Disciplinas
- Roles

RF_2. Gestionar actividades. Debe permitir adicionar, eliminar, modificar o mostrar la actividad.

RF_2.1 Adicionar actividad, de la misma se registran los siguientes datos:

- Descripción

- Artefacto de entrada
- Artefacto de salida
- Orden (Cada actividad tiene un orden que dice que actividad ocurre primero, después o de manera concurrente con otra)

RF_2.2 Eliminar actividad.

RF_2.3 Modificar actividad, los aspectos a modificar son:

- Descripción
- Artefacto de entrada
- Artefacto de salida
- Orden (Cada actividad tiene un orden que dice que actividad ocurre primero o de manera concurrente con otra)

RF_2.4 Mostrar actividad.

RF_3. Gestionar artefacto. Debe permitir adicionar, eliminar y modificar el artefacto.

RF_3.1 Adicionar artefacto, de la misma se registran los siguientes datos:

- Nombre
- Descripción
- Ubicación

RF_3.2 Eliminar artefacto.

RF_3.3 Modificar artefacto, los aspectos a modificar son:

- Nombre
- Descripción
- Ubicación

RF_4. Generar reporte de artefacto.

RF_5. Gestionar ciclos de pruebas. Debe permitir adicionar, eliminar, modificar o mostrar ciclos de pruebas.

RF_5.1 Adicionar ciclo de prueba, de estos se registran los siguientes datos:

- Nombre
- Fecha de inicio
- Fecha de fin

- Pertenece a un producto o componente de un producto
- Iteraciones
- Responsable
- Evaluación, que puede tomar los valores B, R, M

RF_5.2 Eliminar ciclo de prueba.

RF_5.3 Modificar ciclo de prueba, los aspectos a modificar son:

- Nombre
- Fecha de inicio
- Fecha de fin
- Pertenece a un producto o componente de un producto
- Iteraciones
- Responsable
- Evaluación, que puede tomar los valores B, R, M

RF_6 Gestionar iteración. Debe permitir adicionar, eliminar, modificar o mostrar la iteración.

RF_6.1 Adicionar iteración, de la misma se registran los siguientes datos:

- Nombre
- Fecha de Inicio
- Fecha de fin
- Responsable
- Casos de pruebas
- Pertenece a uno o varios ciclos de pruebas
- Evaluación, que puede tomar los valores B, R, M

RF_6.2 Eliminar iteración.

RF_6.3 Modificar iteración, los aspectos que se pueden modificar son:

- Nombre
- Fecha de Inicio
- Fecha de fin
- Responsable
- Casos de pruebas

- Pertenece a uno o varios ciclos de pruebas
- Evaluación, que puede tomar los valores B, R, M

RF_6.4 Mostrar iteración.

RF_7 Gestionar caso de prueba. Debe permitir adicionar, eliminar, modificar o mostrar los casos de prueba.

RF_7.1 Adicionar caso de prueba, de los mismos se registran:

- Identificador
- Nombre
- Descripción
- Respuesta
- Flujos

RF_7.2 Eliminar caso de prueba.

RF_7.3 Modificar caso de prueba, los aspectos que se pueden modificar son:

- Identificador
- Nombre
- Tipo de prueba
- Descripción
- Respuesta
- Flujos

RF_7.4 Mostrar caso de prueba.

RF_8 Gestionar resultado de ejecutar un caso de pruebas. Permitiendo adicionar, eliminar, modificar o mostrar los resultados de los casos de prueba.

RF_8.1 Adicionar resultado de ejecutar un caso de pruebas, de estos se registran los siguientes datos:

- Identificador
- Identificador del caso de prueba al que responde
- Probador (Persona que registra el resultado de la prueba)
- Elemento revisado
- Resultado de la prueba
- No conformidad (Opcional, solo si se encuentran errores en el

- elemento revisado)
- Revisado (Persona a la que se le notifica el resultado de la prueba)

RF_8.2 Eliminar resultado de caso de prueba.

RF_8.3 Modificar resultado de caso de prueba, los aspectos que se pueden modificar son:

- Identificador
- Identificador del caso de prueba al que responde
- Probador (Persona que registra el resultado de la prueba)
- Elemento revisado
- Resultado de la prueba
- No conformidad (Opcional, solo si se encuentran errores en el elemento revisado).
- Revisado (Persona a la que se le notifica el resultado de la prueba)

RF_9 Gestionar métricas, permitiendo adicionar, eliminar, modificar.

RF_9.1 Adicionar métricas, registrando los siguientes datos:

- Nombre de las métricas
- Descripción

RF_9.2 Eliminar métrica.

RF_9.3 Modificar métrica, los aspectos modificables son:

RF_10 Gestionar usuarios. Debe permitir adicionar, eliminar, modificar u visualizar los usuarios.

RF_10.1 Adicionar usuario, registrando los siguientes datos:

- Nombre y Apellidos
- Usuario
- Proyecto/s de trabajo
- Módulo/s al que pertenece el proyecto/s
- Categoría (Estudiante/Profesor)
- Contraseña

RF_10.2 Eliminar usuario.

RF_10.3 Modificar usuario, los datos modificables son:

- Nombre y Apellidos
- Usuario

- Proyecto/s de trabajo
- Módulo/s al que pertenece el proyecto/s
- Categoría (Estudiante/Profesor)
- Contraseña

RF_10.4 Mostrar usuario.

RF_11 Gestionar Roles. Debe permitir adicionar, eliminar, modificar o visualizar los roles.

RF_11.1 Adicionar roles, de los mismos se registran los siguientes datos:

- Nombre
- Permisos
- Descripción
- Usuarios asociados

RF_11.2 Eliminar rol.

RF_11.3 Modificar roles, los datos modificables son:

- Nombre
- Permisos
- Descripción
- Usuarios asociados

RF_11.4 Mostrar rol.

RF_12 Gestionar proyecto. Permitir adicionar, eliminar, modificar o mostrar los proyectos.

RF_12.1 Adicionar proyecto, de estos los datos que se registran son:

- Nombre
- Integrantes con roles asociados
- Módulos a usar
- Autor
- Institución perteneciente

RF_12.2 Eliminar proyecto.

RF_12.2 Modificar proyecto, los datos modificables son:

- Nombre
- Integrantes con roles asociados

- Módulos a usar
- Autor
- Institución perteneciente

RF_12.4 Mostrar proyecto.

RF_13 Gestionar diagramas. Debe permitir Adicionar, Eliminar y Modificar los diagramas.

RF_13.1 Adicionar diagrama, los datos que se registran son los siguientes:

- Identificador
- Nombre

RF_13.2 Eliminar diagrama.

RF_13.3 Modificar diagrama, los datos modificables son:

- Identificador
- Nombre

RF_14 Gestionar proyecto de modelado. Debe permitir Adicionar, Eliminar y Modificar un proyecto de modelado.

RF_14.1 Adicionar proyecto de modelado, de los mismos se registran los siguientes datos:

- identificador
- nombre
- nombre del producto que representa

RF_14.2 Eliminar proyecto de modelado.

RF_14.3 Modificar proyecto de modelado, los datos modificables son:

- identificador
- nombre
- nombre del producto que representa

RF_15 Gestionar vistas arquitectónicas. Debe permitir Adicionar, Eliminar y Modificar las vistas arquitectónicas.

RF_15.1 Adicionar vistas arquitectónicas, de las mismas se registran los siguientes datos:

- identificador
- nombre
- descripción

RF_15.2 Eliminar vistas arquitectónicas.

RF_15.3 Modificar vistas arquitectónicas, los datos modificables son:

- identificador
- nombre
- descripción

2.3 Diseño de la Base de Datos de la Suite de Ingeniería de Software

El diseño de una base de datos consiste en definir la estructura de los datos y sus relaciones en la misma, para un sistema de información determinado. [16] El artefacto principal que genera esta actividad es el modelo de datos el cual describe la representación lógica y física de los datos persistentes.

El proceso de diseño consta de los pasos siguientes:

Determinar la finalidad de la base de datos: esto ayuda a estar preparado para los demás pasos.

Buscar y organizar la información necesaria: reunir todos los tipos de información que desee registrar en la base de datos, como los nombres de productos o los números de pedidos.

Dividir la información en tablas: divida los elementos de información en entidades o temas principales, como Productos o Pedidos, cada tema pasará a ser una tabla.

Convertir los elementos de información en columna: decida qué información desea almacenar en cada tabla. Cada elemento se convertirá en un campo y se mostrará como una columna en la tabla. Por ejemplo, una tabla Empleados podría incluir campos como Apellido y Fecha de contratación.

Especificar claves principales: elegir la clave principal de cada tabla, es una columna que se utiliza para identificar inequívocamente cada fila, como Id_de_producto o Id_de_pedido.

Definir relaciones entre las tablas: examinar cada tabla y decidir cómo se relacionan los datos de una tabla con las demás tablas. Agregar campos a las tablas o crear nuevas tablas para clarificar las relaciones según sea necesario.

Ajustar el diseño: analizar el diseño para detectar errores, crear las tablas y agregar algunos registros con datos de ejemplo. Comprobar si se puede obtener los resultados previstos de las tablas, realizar los ajustes necesarios en el diseño.

Aplicar las reglas de normalización: aplicar reglas de normalización de los datos para comprobar si las

tablas están estructuradas correctamente y realizar los ajustes necesarios en las tablas del diseño de base de datos. [17]

2.3.1 Aplicación de patrones en el diseño de la base de datos

Módulo Gestión de la Información

En este módulo se observa la siguiente jerarquía (ver figura 1) para representar la misma en la base de datos se utiliza el patrón de árbol fuertemente codificado, el mismo es utilizado para representar jerarquías que son bien conocidas, recomendado en las estructuras en las que sean poco probables que ocurran cambios y admite tanto niveles como tenga la jerarquía.

Figura 1 Jerarquía de módulo

La entidad **artefacto**, no siempre tiene bien definidos sus atributos, para resolver este problema se utiliza el patrón entidad-atributo-valor, es la representación de un modelo flexible que permite adicionar tantas entidades como se deseen con sus atributos, es un acercamiento a la orientación a objeto representado en el modelo relacional.

Además se utiliza el patrón de llaves subrogadas que facilita la interacción con la base de datos en un futuro, plantea que se genere una llave primaria única para cada entidad, en vez de utilizar un atributo identificador en el contexto dado.

2.3.2 Conceptos identificados

Con el análisis de los requisitos se identificaron los conceptos fundamentales que se quieren almacenar en la base de datos, convirtiéndose estos en entidades del sistema, las mismas se describen a continuación:

Tabla: 2 Conceptos identificados

Concepto	Descripción	Atributo	Relación
Usuario	Ente que intercambia con el sistema.	• Nombre y Apellidos • Usuario • Proyecto/s de trabajo • Categoría (Estudiante/Profesor) • Contraseña	Proyecto, Rol
Rol	Es la clasificación que le son otorgados a los usuarios dentro de ellos existe el rol privilegiado admin que posee todos los permisos.	• Nombre • Descripción • Permisos • Usuarios asociados	Proyecto, Usuario
Proyecto	Conjunto de archivos sobre los cuales trabajan los usuarios asociados al mismo en dependencia del rol asignado.	• Nombre • Integrantes con roles asociados • Módulos a usar • Autor • Institución perteneciente	Rol, Usuario, Metodología, Producto
Metodología	Guía el proceso de desarrollo de software.	• Nombre • Descripción • Actividades • Roles • Fase	Fase, Proyecto
Fase	Diferentes etapas por los cuales transita el software para su desarrollo.	• Nombre • Descripción • Actividades	Metodología, Disciplina
Disciplina	Conjunto de actividades que tiene un objetivo específico dentro de un área del desarrollo de software.	• Nombre • Descripción • Actividades	Actividad, Artefacto, Fase, Ciclo de prueba
Actividad	Son aquellas acciones que se realizan por cada disciplina.	• Descripción • Artefacto de entrada	Disciplina, Rol

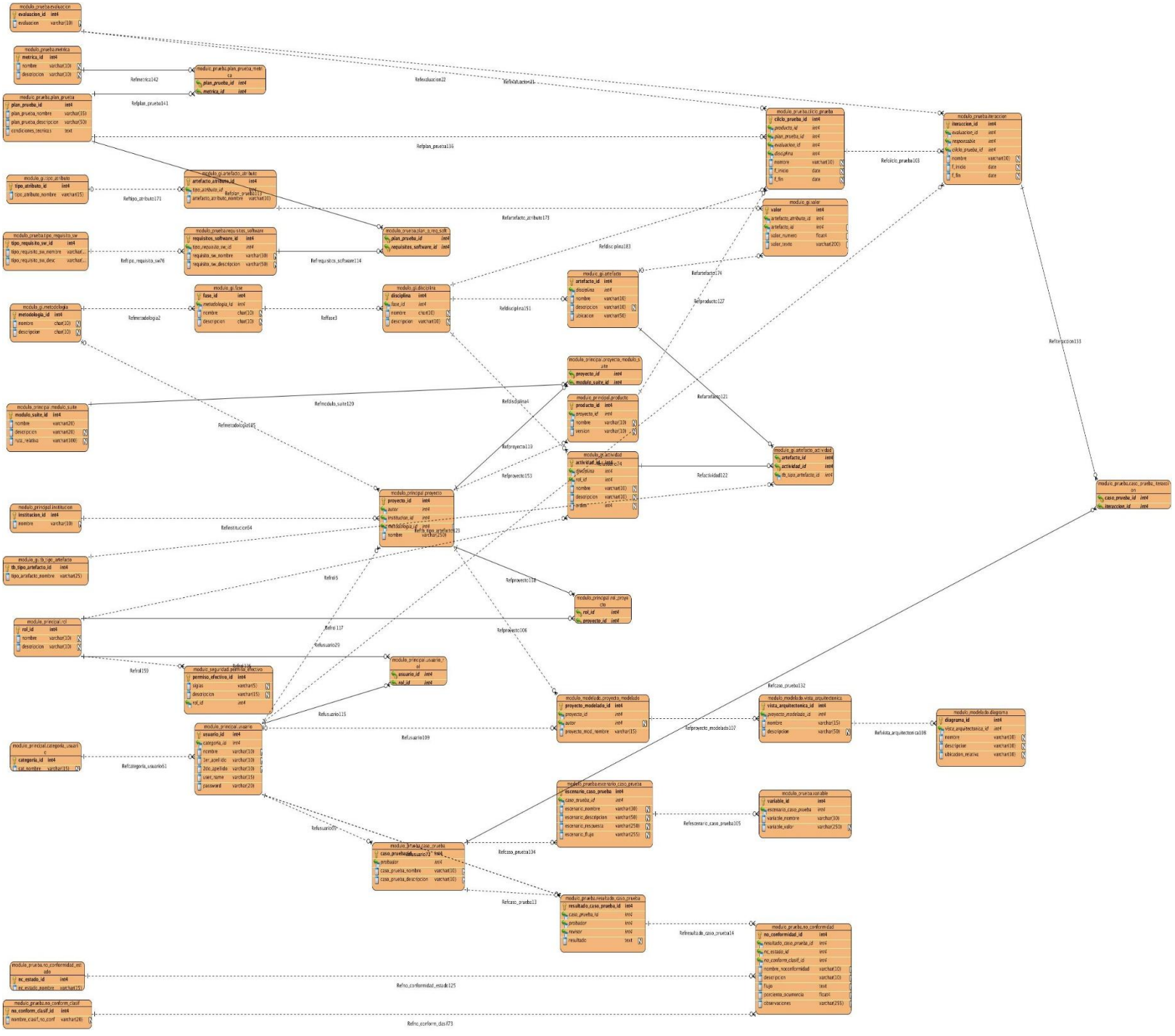
		• Artefacto de salida • Orden (Cada actividad tiene un orden que dice que actividad ocurre primero o de manera concurrente con otra)	
Artefacto	Documentos que se generan durante el proceso de desarrollo de software.	• Nombre • Descripción • Ubicación	Disciplina
Ciclo de prueba.	Fases de la ejecución de las pruebas para un hito del proyecto.	• Nombre • Fecha de inicio • Fecha de fin • Iteraciones • Responsable • Producto al que pertenece • Evaluación: B R M	Iteraciones, Plan de prueba, Producto, Disciplina
Iteración	Fases de ejecución de los casos de pruebas y resolución de las no conformidades.	• Nombre • Fecha de inicio • Fecha de fin • Responsable • Casos de pruebas • Ciclo de pruebas que pertenece • Evaluación: B R M	Caso de prueba, Ciclo de prueba

Caso de prueba	Es el artefacto que describe la prueba a ejecutar.	• Identificador • Nombre • Escenarios • Descripción • Respuesta • Flujos • Variables(nombre y valores)	Ciclo de prueba, Iteraciones, Resultado de caso de prueba
Resultado del caso de prueba	Es lo que se obtiene de la ejecución del caso de prueba.	• Identificador • Identificador del caso de prueba al que responde • Probador (Persona que registra el resultado de la prueba) • Resultado de la prueba • No conformidad (Opcional, sólo si se encuentran errores en el elemento revisado) • Revisado (Persona a la que se le notifica el resultado de la prueba)	No conformidad, Caso de prueba
No conformidad	Es el artefacto que describe el resultado no positivo de un caso de prueba.	• id_NoConf • NoConf • Descrip • Flujo: Secuencia de pasos que siguió el probador hasta esa no conformidad • Clasif: puede ser significativa, no significativa o una	Resultado del caso, Iteración

		recomendación • %Ocu: el por ciento de ocasión es en que se ha reportado del total de veces que se ha ejecutado el caso de prueba • Etapa de detección (la iteración en la que se generó como respuesta a un caso de prueba)	
Métrica	Es la fórmula para calcular la evaluación de las iteraciones.	• Nombre de la métrica. • Descripción.	Plan de prueba
Plan de pruebas	Artefacto que describe el proceso de pruebas a ejecutar.	• Roles y # de participantes por rol • Responsabilidades de los roles • Recursos del sistema -PC Clientes -Servidores • Requisitos a probar • Fecha inicio ciclo • Fecha fin ciclo • Fecha inicio iteración • Fecha fin iteración	Métrica, Ciclo de prueba
Requisito de software	Funcionalidad o cualidad que un software debe cumplir.	• Nombre • Descripción • Tipo: puede ser funcional o no funcional	Plan de prueba
Proyecto de modelado	Es el que engloba todo el desarrollo del modelado, es donde se crean los diagramas que necesita el proyecto y se realizan las modificaciones	• Identificador • Nombre • Nombre del producto al que pertenece	Proyecto, Usuario, Vistas arquitectónicas

	necesarias en dependencia del diagrama a realizar.		
Vista arquitectónica	Es donde se crean, modifican, eliminan los diagramas en dependencia de lo que necesite el usuario.	• Identificador • Nombre • Descripción	Proyecto de modelado, Diagrama
Diagrama	Son los diseños que se realizan para definir las vistas.	• Identificador • Nombre • Descripción	Vista arquitectónica
Producto	Es lo que se obtiene luego de realizar un ciclo de vida completo de un software.	• Nombre • Versión	Ciclo de prueba, Proyecto

Luego de definidas la entidad con sus atributos y relaciones se realiza el diseño del modelo relacional, representado mediante el diseño físico que se muestra a continuación:



2.3.3 Descripción de las tablas

El modelo de datos obtenido en la solución está conformado por 41 tablas, 35 son de datos y 6 nomencladores, como resultado de la integración de todos los módulos de la Suite de Ingeniería de Software. Estas tablas están definidas a partir de los conceptos previamente identificados y de las relaciones que entre ellos se establecen. Entre las más importantes figuran:

Tabla 3: Usuario

Nombre de la tabla: usuario		
Descripción: Almacena todos los usuarios del sistema.		
Atributo	**Tipo**	**Descripción**
usuario_id	integer	Identificador de la tabla.
categoria_id(FK)	integer	Identificador de la categoría del usuario.
nombre	varchar(15)	Almacena el nombre del usuario.
1er_apellido	varchar(15)	Almacena el primer apellido del usuario.
2do_apellido	varchar(15)	Almacena el segundo apellido del usuario.
user	varchar(10)	Almacena el nombre con el que se identifica el usuario en el sistema.
password	varchar(25)	Almacena la contraseña con la cual el usuario accede al sistema.

Tabla 4: Proyecto

Nombre de la tabla: proyecto		
Descripción: Almacena todos los proyectos que se crean en el sistema.		
Atributo	**Tipo**	**Descripción**
proyecto_id	integer	Identificador de la tabla.
autor(FK)	integer	Identificador del usuario que crea el proyecto.

institucion_id(FK)	integer	Identificación de la institución a la que pertenece el proyecto.
nombre	varchar(50)	Almacena el nombre con el que se registra el proyecto.

Tabla 5: Rol

Nombre de la tabla: rol		
Descripción: Almacena todos los roles del sistema.		
Atributo	**Tipo**	**Descripción**
rol_id	integer	Identificador de la tabla.
nombre	varchar(15)	Almacena el nombre del rol.
descripción	varchar(255)	Breve descripción del rol.

Tabla 6: Metodología

Nombre de la tabla: metodologia		
Descripción: Almacena las metodologías que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
metodologia_id	integer	Identificador de la tabla.
nombre	varchar(15)	Almacena el nombre de la metodología.
descripción	varchar(255)	Identificación de la institución a la que pertenece el proyecto.

Tabla 7: Fase

Nombre de la tabla: fase		
Descripción: Almacena todos las fases que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
fase_id	integer	Identificador de la tabla.
metodología_id(FK)	integer	Identificador de la metodología a la que pertenece.

institucion_id(FK)	integer	Identificación de la institución a la que pertenece el proyecto.
nombre	varchar(15)	Almacena el nombre de la fase.
descripcion	varchar(255)	Breve descripción de que es la fase y que hace.

Tabla 8: Disciplina

Nombre de la tabla: disciplina		
Descripción: Almacena todas las disciplinas que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
Disciplina	integer	Identificador de la tabla.
autor(FK)	integer	Identificador del usuario que crea el proyecto.
institucion_id(FK)	integer	Identificación de la institución a la que pertenece el proyecto.
nombre	varchar(50)	Almacena el nombre con el que se registra el proyecto.

Tabla 9: Actividad

Nombre de la tabla: actividad		
Descripción: Almacena todos las actividades que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
actividad_id	integer	Identificador de la tabla.
disciplina(FK)	integer	Identificador de la disciplina a la cual pertenece la actividad.
rol_id(FK)	integer	Identificación del rol encargado de realizar la actividad.
nombre	varchar(15)	Almacena el nombre con el

		que se registra la actividad.
descripcion	varchar(255)	Breve descripción que se hace en la actividad.
orden	integer	Especifica el orden en que se ejecutan las actividades, dice que actividad ocurre primero o de manera concurrente con otra.

Tabla 10: Artefacto

Nombre de la tabla: artefacto		
Descripción: Almacena todos los artefactos que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
artefacto_id	integer	Identificador de la tabla.
nombre	varchar(15)	Almacena el nombre del artefacto.
descripción	varchar(255)	Breve descripción de que es el artefacto y qué se registran en él.
ubicación	varchar(30)	Especifica el lugar donde se van a guardar los artefactos generados.

Tabla 11: Plan de prueba

Nombre de la tabla: plan_prueba		
Descripción: Almacena todos los proyectos que se crean en el sistema.		
Atributo	**Tipo**	**Descripción**
plan_prueba_id	integer	Identificador de la tabla.
plan_prueba_nombre	varchar(15)	Guarda el nombre del plan de prueba.
plan_prueba_descripción	varchar(255)	Breve descripción de lo que

		hace el plan de prueba y para qué es.
condiciones_tecnicas	text	Especifica las condiciones de los requerimientos del sistema.

Tabla 12: Ciclo de prueba

Nombre de la tabla: ciclo_prueba		
Descripción: Almacena todos los proyectos que se crean en el sistema.		
Atributo	**Tipo**	**Descripción**
ciclo_prueba_id	integer	Identificador de la tabla.
producto_id(FK)	integer	Identificador del producto al cual se le aplica el ciclo de prueba.
plan_prueba_id(FK)	integer	Identificador del plan de prueba al que pertenece.
evaluacion_id(FK)	integer	Identificación de la evaluación que se le dio al ciclo de prueba luego de finalizado.
nombre	varchar(15)	Almacena el nombre del ciclo de prueba.
f_inicio	date	Fecha en la que comienza el ciclo de prueba.
f_fin	date	Fecha en que culmina el ciclo de prueba.

Tabla 13: Iteración

Nombre de la tabla: iteracion		
Descripción: Almacena todos los proyectos que se crean en el sistema.		
Atributo	**Tipo**	**Descripción**
iteración_id	integer	Identificador de la tabla.

evaluación(FK)	integer	Identificador de la evaluación que se le da a la iteración luego de finalizada la misma.
responsable_id(FK)	integer	Identificación del usuario encargado de ejecutar la iteración.
ciclo_prueba_id(FK)	integer	Identificador del ciclo de prueba al que pertenece la iteración.
nombre	varchar(15)	Almacena el nombre con el que se registra la iteración.
f_inicio	date	Fecha de comienzo de la iteración.
f_fin	date	Fecha en que finaliza la iteración.

Tabla 14: Caso de prueba

Nombre de la tabla: caso_prueba		
Descripción: Almacena lo casos de prueba que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
caso_prueba_id	integer	Identificador de la tabla.
probador(FK)	integer	Identificador del usuario que ejecuta el caso de prueba.
caso_prueba_nombre	varchar(15)	Almacena el nombre de los casos de pruebas.
caso_prueba_descripción	varchar(255)	Breve descripción del caso de prueba.

Tabla 15: Resultado de Caso de Prueba

Nombre de la tabla: resultado_caso_prueba
Descripción: Almacena todos los resultados de los casos de prueba que se ejecutan.

Atributo	Tipo	Descripción
resultado_caso_prueba_id	integer	Identificador de la tabla.
caso_prueba_id(FK)	integer	Identificador caso de prueba al que pertenece.
probador(FK)	integer	Identificación del usuario que ejecutó el caso de prueba.
revisor(FK)	integer	Identificador del usuario encargado de revisar y analizar los resultados obtenidos.
resultado	varchar(255)	Guarda el resultado que se genera de ejecutar el caso de prueba que puede ser positivo o generar una nueva no conformidad.

Tabla 16: No Conformidad

Nombre de la tabla: no_conformidad		
Descripción: Almacena las no conformidades que se generan de ejecutar un caso de prueba caso de prueba.		
Atributo	**Tipo**	**Descripción**
no_conformidad_id	integer	Identificador de la tabla.
resultado_caso_prueba_id(FK)	integer	Identificador del resultado de caso de prueba del cual se genera la no conformidad.
nc_estado_id(FK)	integer	Identificador del estado en el cual se encuentra la no conformidad, que puede ser: Resuelta, No procede, Pendiente.
no_conform_clasif_id(FK)	integer	Identificador de la

		clasificación que presenta la no conformidad, puede ser: Significativa, no Significativa o Recomendación.
nombre_noconformidad	varchar(20)	Almacena los nombres de las no conformidades.
descripción	varchar(255)	Breve descripción de la no conformidad.
flujo	text	Secuencia de pasos que siguió el probador hasta esa no conformidad.
porciento_ocurrencia	float	Por ciento de ocasión en que se ha reportado la no conformidad.

Tabla 17: Métrica

Nombre de la tabla: metrica		
Descripción: Almacena las métricas que se utilizan para evaluar los ciclos de prueba e iteraciones.		
Atributo	**Tipo**	**Descripción**
metrica_id	integer	Identificador de la tabla.
nombre	varchar(15)	Almacena el nombre de la métrica.
descripcion	varchar(255)	Breve descripción de la métrica, que hace y como se ejecuta.

Tabla 18: Producto

Nombre de la tabla: producto		
Descripción: Almacena los productos que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
producto_id	integer	Identificador de la tabla.

nombre	varchar(25)	Nombre del producto que se registra.
version	varchar(10)	Guarda la versión en la que se encuentra el producto.

Tabla 20: Requisitos de software

Nombre de la tabla: requisitos_software		
Descripción: Almacena los requisitos de software que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
requisitos_software_id	integer	Identificador de la tabla.
tipo_requisito_software(FK)	integer	Identificador del tipo de requisito.
requisito_sw_nombre	varchar(15)	Almacena el nombre de los requisitos.
requisito_sw_descripcion	varchar(255)	Breve descripción de los requisitos.

Tabla 21: Proyecto de Modelado

Nombre de la tabla: proyecto_modelado		
Descripción: Almacena lo proyecto de modelados que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
proyecto_modelado_id	integer	Identificador de la tabla.
proyecto_id(FK)	integer	Identificador del proyecto al cual pertenece el proyecto de modelado.
autor(FK)	integer	Identificador del usuario que creo el proyecto de modelado.
proyecto_modelado_nombre	varchar(10)	Almacena los nombres de los proyectos modelados.

Tabla 22: Vista Arquitectónica

Nombre de la tabla: vista_arquitectonica

Descripción: Almacena las vistas arquitectónicas que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
vista_arquitectonica_id	integer	Identificador de la tabla.
proyecto_modelado_id(FK)	integer	Identificador del proyecto de modelado al cual pertenece la vista.
nombre	varchar(15)	Almacena el nombre de las vistas arquitectónicas.
descripción	varchar(255)	Breve descripción de la vista arquitectónica.

Tabla 23: Diagrama

Nombre de la tabla: diagrama		
Descripción: Almacena los diagramas que se registran en el sistema.		
Atributo	**Tipo**	**Descripción**
diagrama_id	integer	Identificador de la tabla.
vista_arquitectonica_id(FK)	integer	Identificador de la vista arquitectónica a la cual pertenece el diagrama.
nombre	varchar(15)	Almacena el nombre de los diagramas.
descripción	varchar(255)	Breve descripción de los diagramas.

2.4 Esquemas definidos

Los esquemas en una base de datos describen la estructura de la misma, en un lenguaje formal soportado por un sistema administrador de base de datos (DBMS), son usados para separarla de manera lógica. En una base de datos relacional, el esquema define sus tablas, sus campos en cada tabla y sus relaciones. Los esquemas definidos para la solución propuesta se describen a continuación:

Se define un esquema por cada uno de los módulos de la Suite de Ingeniería de Software y uno para la seguridad, por lo que se cuenta con 4 esquemas denominados: modulo_gi, modulo_prueba,

modulo_principal, modulo_modelado, modulo_seguridad

2.5 Estrategia de Copias de Respaldo

Para enfrentar los riesgos de pérdida de información ante posibles fallos o caídas inesperadas del sistema con la consecuente corrupción de los mismos se establece como medida de seguridad realizar una copia total del sistema con criterio semanal y una incremental con carácter diario, conservando solamente las últimas 4 de mayor período (Semanal) y todas las subdivisiones de la misma lo que posibilita una cobertura total de un mes de trabajo.

2.6 Estrategia inicial de indexado

Un índice es una estructura de datos definida sobre una o varias columnas de la tabla, permite localizar de forma rápida las filas de la tabla en base a su contenido en la columna indexada, además de permitir recuperar las filas de la tabla ordenadas por esa misma columna. Estos tienen diferentes clasificaciones como se muestra a continuación:

Índice simple y compuesto

Un índice simple está definido sobre una sola columna de la tabla mientras que un índice compuesto está formado por varias columnas de la misma tabla (tabla sobre la cual está definido el índice). Cuando se define un índice sobre una columna, los registros que se recuperen utilizando el índice aparecerán ordenados por el campo indexado. Si se define un índice compuesto por las columnas col1 y col2, las filas que se recuperen utilizando dicho índice aparecerán ordenadas por los valores de col1 y todas las filas que tengan el mismo valor de col1 se ordenarán a su vez por los valores contenidos en col2. Por ejemplo si definimos un índice compuesto basado en las columnas (provincia, localidad), las filas que se recuperen utilizando este índice aparecerán ordenadas por provincia y dentro de la misma provincia por localidad.

Índice agrupado y no agrupado

El término índice agrupado no se debe confundir con índice compuesto, el significado es totalmente diferente. Un índice agrupado es un índice en el que el orden lógico de los valores de clave determina el orden físico de las filas correspondientes de la tabla. El nivel inferior, u hoja, de un índice agrupado contiene las filas de datos en sí de la tabla. Una tabla o vista permite un solo índice agrupado al mismo tiempo. Los índices no agrupados que existentes en las tablas se vuelven a generar al crear un índice agrupado, por lo que es conveniente crear el índice agrupado antes de crear los índices no agrupados.

Estos especifican la ordenación lógica de la tabla. Con un índice no agrupado, el orden físico de las filas de datos es independiente del orden indexado.

Índice único

Índice único es aquel en el que no se permite que dos filas tengan el mismo valor en la columna de clave del índice. Es decir que no permite valores duplicados.

Todos los tipos de índices generalmente están enfocados en hacer eficientes las consultas, pero teniendo en cuenta los datos almacenados, su cantidad y variabilidad, es que se toma la decisión de qué índices definir.

La solución propuesta utiliza el indexado que trae por defecto el gestor PostgreSQL, para la búsqueda de datos utilizando las llaves primarias, foráneas y campos únicos. Todas las llaves primarias, que son llaves subrogadas además, poseen índices de tipo "*b-tree*" (Árboles-B), de esta manera, cualquier búsqueda que se realice utilizando las llaves se optimiza mediante este método.

Algunas instrucciones para decidir cuáles índices implicarán una mejora significativa en el rendimiento del sistema ante consultas son:

- Evitar crear demasiados índices en tablas que se actualizan con mucha frecuencia y procurar definirlos con el menor número de columnas.
- Es conveniente utilizar un número mayor de índices para mejorar el rendimiento de consulta en tabla con pocas necesidades de actualización pero con grandes volúmenes de datos. Un gran número de índices contribuyen a mejorar el rendimiento de las consultas que no modifican datos, tales como *Select*, ya que el optimizador de consulta dispone de más índices entre los que elegir para determinar el método de acceso más rápido.
- Se recomienda utilizar una longitud corta en la clave los índices agrupados, los no agrupados también mejoran si se crean en columnas únicas o que no admiten valores Nulo.
- Un índice único en lugar de un índice no único con la misma combinación de columnas proporciona información adicional al optimizador de consultas y por tanto, resulta más útil. [18]

2.7 Consideraciones finales del capítulo

En el presente capítulo se hace una descripción de la propuesta solución donde se muestra el modelo físico de la base de datos listo para montarlo en el SGBD definido, se hace una descripción de las principales tablas, se definen una serie de aspectos a tener en cuenta en el correcto diseño de la base de

datos de la Suite de Ingeniería de Software, especificándose también la utilización de patrones que permiten obtener una base de datos robusta, escalable y segura. Con el diseño obtenido se va a lograr almacenar toda la información que se genera en la Suite, logrando así una exitosa implementación de la misma.

CAPÍTULO 3: VALIDACIÓN DEL DISEÑO PROPUESTO PARA LA SUITE DE INGENIERÍA DE SOFTWARE.

3.1 Introducción

El presente capítulo está enfocado hacia el análisis y validación del diseño propuesto a través de pruebas teóricas y funcionales como son la normalización, pruebas de carga y estrés.

3.2 Normalización

La normalización es una técnica importante en el diseño de las bases de datos, permite obtener estructuras de datos eficientes, es un proceso a través del cual se aplican reglas a las relaciones obtenidas de la transformación del modelo lógico al relacional. Evitando así redundancia de datos, problemas de actualización en las tablas y permitiendo proteger la integridad de los datos.

La normalización involucra varias fases que se realizan en orden, es decir para pasar a la siguiente fase debe haberse concluido la anterior.

Tras completar cada fase se dice que la relación está en:
Primera Forma Normal (1FN)
Segunda Forma Normal (2FN)
Tercera Forma Normal (3FN)
Forma Normal de Boyce-Codd (FNBC)
Existen, además, 4FN y 5FN
Las bases de datos relacionales solo llegan hasta la 3F.

Primera Forma Normal (1FN)

Una relación está en (1FN) si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos.

- Toda relación normalizada, o sea, con valores atómicos de los atributos.
- La relación no incluye ningún grupo repetitivo.

Segunda Forma Normal (2FN)

Además de estar en 1FN, asegura que los atributos no llaves dependen completamente de la llave primaria, entonces si la tabla está en primera forma normal y además el atributo de la llave primaria no es compuesto está en 2FN.

Tercera Forma Normal (3FN)

Si está en 2FN y si, y sólo si, los atributos no llaves son independientes de cualquier otro atributo no llave primaria es decir que se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria.

Se puede decir que el diseño de la base de datos propuesto se encuentra en 3FN ya que todos los atributos son atómicos, los no llave dependen completamente de la llave primaria y no existen dependencias transitivas. [19]

3.3 Integridad de los Datos

La integridad de los datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando la información se modifica mediante sentencias SQL como INSERT, UPDATE, o DELETE, la integridad de los datos almacenados puede perderse de varias formas ejemplo: puede añadirse datos no válidos al sistema.

Para enfrentar estos riegos se utilizan las restricciones que brinda el SGBD PostgreSQL, para proteger la integridad de los datos, estas son:

Restricción de dominio:

Cada atributo está acompañado de su dominio lo cual obliga a que esos campos contengan ese tipo de dato.

Restricción NOT NULL

Se específica mediante la restricción NOT NULL si el atributo no puedo ser nulo.

Restricción de FOREIGN KEY (llave foránea)

Expresa que un determinado atributo es una referencia a un atributo llave primaria de otra entidad. Con esto se logra que se mantenga la integridad y la consistencia de los datos. [20]

3.4 Prueba de Carga y Estrés

El rendimiento es una característica a fundamental a tener en cuenta cuando se desarrolla un sistema, para analizar el rendimiento de la base de datos de Suite de Ingeniería de Software se le realizan las pruebas de carga y estrés:

Estrés: El testeo de estrés intenta romper el sistema bajo testeo desbordando sus recursos o reduciendo

la cantidad de estos. Los objetivos del testeo de estrés son encontrar los límites del sistema y asegurar que tras un fallo en el sistema, se recupera sin causar graves problemas.

Carga: generalmente se refiere a la práctica de testear el comportamiento de una aplicación mediante cargas o entradas pesadas para verificar si el sistema satisface los requisitos de rendimiento para situaciones críticas como, por ejemplo, la cantidad límite de usuarios accediendo de forma concurrente a los servicios del programa, documentos extremadamente grandes, cantidad de transacciones que se pueden procesar de forma concurrente cada minuto, tiempo de respuesta, etc. [21]

Para la realización de estas pruebas se emplea la herramienta JMeter la cual se configuró para conectarse a la base de datos, para poblar la misma se utilizó EMS Date Generator para PostgreSQL. Dichas pruebas se realizaron con 1 consulta, para las cuales el número de usuarios fue diferente por lo que arrojaron resultados distintos que se presentan a continuación a través de las siguientes variables:

- Media: es el tiempo promedio de respuesta de todas las peticiones.
- Mediana: tiempo promedio de la mitad de los resultados, la otra mitad tomará un tiempo entre la mediana y el valor máximo.
- Mínimo: mínimo de tiempo de respuesta de un petición a la base de datos.
- Máximo: máximo de tiempo de respuesta de una petición.

Consulta a probar: Obtener el nombre del proyecto que tenga asignado la metodología ´x´. (Ver **anexo 1**)

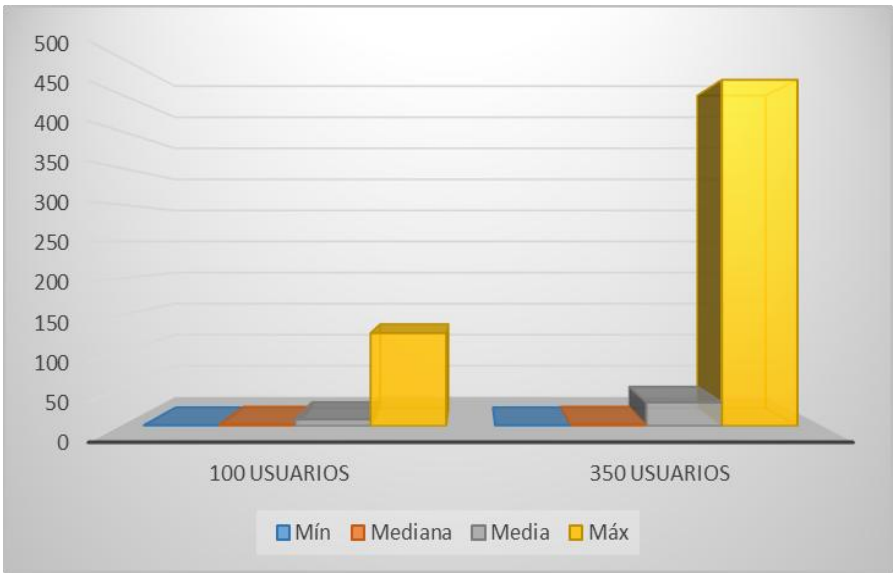
Select proyecto.nombre from modulo_principal.proyecto, modulo_gi.metodologia where modulo_principal.proyecto.metodologia_id=modulo_gi.metodologia.metodologia_id and modulo_gi.metodologia.nombre='DJTM'

Después de validar la consulta se procede a los resultados en la herramienta JMeter:
Resultados de las pruebas para 100 hilos.

Cantidad de usuarios concurrentes	Media(ms)	Mediana(ms)	Mínimo(ms)	Máximo(ms)
100	9	2	1	126

Resultados de las pruebas para 350 hilos.

Cantidad de usuarios concurrentes	Media(ms)	Mediana(ms)	Mínimo(ms)	Máximo(ms)
350	32	2	1	469



Conclusiones finales del capítulo

En el presente capítulo se describe la validación de la base de datos de la Suite de Ingeniería de Software por lo que se puede plantear que se han cumplido los objetivos propuestos para el mismo, donde se concluye que la base de datos queda normalizada hasta la tercera forma normal con lo que se garantiza evitar redundancia en los datos y evitar anomalías de actualización. Además se pobló las tablas de la base de datos para realizarle pruebas de carga y estrés donde respondió satisfactoriamente, estas acciones validan la propuesta de solución cumpliendo de esta forma los objetivos trazados en la presente investigación

CONCLUSIONES FINALES

Durante el desarrollo de este trabajo se expuso la necesidad de diseñar una base de datos para la Suite de Ingeniería de Software que permita integrar todo el flujo de información que se genera en la Suite.

- Del estudio del arte realizado sobre los sistemas de bases de datos existentes se identificó como mejor solución el diseño de una base de datos relacional para dar respuesta al problema identificado.
- El modelo de datos definido garantiza la gestión integrada dela información que generan los subsistemas de la Suite de Ingeniería de Software.
- Los resultados obtenidos en las pruebas realizadas permitieron validar el diseño propuesto para la solución.

Luego de estos procesos de trabajo realizados se considera que se han cumplidos los objetivos planteados y se puede concluir que la base de datos para la Suite de Ingeniería de Software dará solución a la situación problemática planteada.

RECOMENDACIONES

Después de haber diseñado la base de datos para la Suite de Ingeniería de Software en la presente investigación se recomienda:

- El refinamiento del modelo físico propuesto en próximas iteraciones del desarrollo del software.
- Utilización de tablespaces, para separar las tablas más utilizadas de las de menos gestión o de las que no varían en contenido, con el objetivo de agilizar la gestión de las mismas.
- Garantizar un proceso de reingeniería sobre el diseño en base a los indicadores de desempeño y funcionalidad generados sobre la explotación del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. Alcalde Alejandro. Introducción a base de datos. [En línea] 2010. Disponible en:http://elbauldelprogramador.com/programacion/basededatos/introduccion-base-de-datos/. Consultado: 2013-01-29.
2. Aulaclic. Unidad 8. El DDL, Lenguaje de Definición de Datos, 8.10. Definición de índice [En línea] febrero 2010.Disponible en: http ://www.aulaclic.es/sqlserver/t_8_15.htm.
3. Blaha Michael. Patterns of data modeling. 2010.
4. Camps Paré, Casillas Santillán Luis Alberto, Costal Costa Dolors, Gibert Ginestá Marc, Martín Escofet Carme, Pérez Mora Oscar. Bases de datos,[En línea]mayo 2005, Disponible en:www.sw-computacion.f2s.com/Linux/007-Bases de datos.pdf.
5. Editorial McGraw-Hill. SGBASE DE DATOS. Sistemas gestores de bases de datos (primera parte).Capítulo 6: SGBASE DE DATOS. Componentes y lenguajes. [En línea] 2008. Disponible en: http://www.emagister.com/curso-sistemas-bases-datos/sgbase de datos-componentes-lenguajes. Consultado: 2013-01-22.
6. Estudio Comparativo de Bases de Datos .Disponible en: dspace.ups.edu.ec/bitstream/123456789/522/7/CAPITULO5.pdf.
7. García Chávez Carlos Alberto. Diseño de base de datos relacionales. Capítulo 3: Sistemas de bases de datos. [en línea] 2005 Disponible en:http://www.emagister.com/curso-diseno-base-datos-relacionales/sistemas-bases-datos. Consultado: 2013-01-28.
8. Gimson Loraine. Metodologías ágiles y desarrollo basado en conocimiento. 2012.
9. Herramientas Case Disponible en: www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf. Consultado: 2013-02-05.
10. Instituto Tecnológico de Informática. Testeo de estrés y carga. [En línea] diciembre 2010. Disponible en: http://www.iti.es/servicios/servicio/resource/7240/index.html.
11. Lamarca Lapuente María Jesús. Bases de datos. [En Línea] 2011. Disponible en: http://www.hipertexto.info/documentos/b_datos.htm. Consultado: 2012-12-11.
12. Sergi Blanco Cuaresma. Metodologías de desarrollo [En línea] febrero 2008.Disponible en:

http://www.marblestation.com/?p=644. Consultado: 2013-03-15.

13. Larrea Vivar. Base de Datos y Análisis del Sistema de Monitoreo. [En Línea] 2007. Disponible en: http:// dspace.ups.edu.ec/bitstream/123456789/178/4/Capitulo%203.pdf.
14. Lic. Rosa María Mato García. Diseño de Bases de Datos. [En línea] octubre 1999.Disponible en: http. Consultado: 2013-03-22.
15. Fernandez Del Monte Yusleydi. Planilla de Especificación de Requisitos. Habana 2012.Disponible en: Expediente de proyecto de Nova Qalit.
16. Tramullas Jesús. Diseño de Bases de Datos [En línea] marzo 2013.Disponible en: http://www.slideshare.net/tramullas/diseo-de-bases-de-datos-17102226. Consultado: 2013-02-18.
17. Office. Conceptos básicos del diseño de una base de datos [En línea] 2007. Disponible en:http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx.
18. Portal en español de PostgreSQL. Integridad referencial con PostgreSQL [En línea] 2009. Disponible en:http://www.postgresql.org.es/node/249. Consultado: 2013-03-26.
19. RODRÍGUEZ YUNTA, Luis. Bases de datos documentales: estructura y uso [En Línea] 2001.Disponible en:www.unav.es/dpp/documentacion/proteger/lryunta.pdf.
20. Romero Vargas Francisco.Tema1 SISTEMAS DE ALMACENAMIENTO DE LA INFORMACIÓN. Disponible en:www.iesromerovargas.com/moodle/pluginfile.-php/327/.../T1.pd. Consultado: 2013-01-30.
21. Silva Viera Diego. Diferencias entre los SGBASE DE DATOS´s [En línea] 2012.Disponible en:http://www.slideshare.net/diegosilvaviera1/diferencias-entre-los-sgbase de datoss. Consultado: 2013-02-01.

BIBLIOGRAFÍA

Sitio oficial de PostgreSQL. Disponible en: http://www.postgresql.org. Consultado: 2013-02-03.

Larrea Vivar. Base de Datos y Análisis del Sistema de Monitoreo. [En Línea] 2007. Disponible en: http:// dspace.ups.edu.ec/bitstream/123456789/178/4/Capitulo%203.pdf.

RODRÍGUEZ YUNTA, Luis. Bases de datos documentales: estructura y uso [En Línea] 2001.Disponible en:www.unav.es/dpp/documentacion/proteger/lryunta.pdf.

Estudio Comparativo de Bases de Datos .Disponible en: dspace.ups.edu.ec/bitstream/123456789/522/7/CAPITULO5.pdf.

Alcalde Alejandro. Introducción a base de datos. [En línea] 2010. Disponible en:http://elbauldelprogramador.com/programacion/basededatos/introduccion-base-de-datos/. Consultado: 2013-01-29.

Lamarca Lapuente María Jesús. Bases de datos. [En Línea] 2011. Disponible en: http://www.hipertexto.info/documentos/b_datos.htm. Consultado: 2012-12-11.

Camps Paré, Casillas Santillán Luis Alberto, Costal Costa Dolors, Gilbert Ginestá Marc, Martín Escofet Carme, Pérez Mora Oscar. Bases de datos, [En línea] mayo 2005, Disponible en:www.sw-computacion.f2s.com/Linux/007-Bases de datos.pdf.

Romero Vargas Francisco.Tema1 SISTEMAS DE ALMACENAMIENTO DE LA INFORMACIÓN. Disponible en:www.iesromerovargas.com/moodle/pluginfile.-php/327/.../T1.pd. Consultado: 2013- 01-30.

Editorial McGraw-Hill. SGBD. Sistemas gestores de bases de datos (primera parte).Capítulo 6: SGBD. Componentes y lenguajes. [En línea] 2008. Disponible en: http://www.emagister.com/curso-sistemas-bases-datos/sgbase de datos-componentes-lenguajes. Consultado: 2013-01-22.

García Chávez Carlos Alberto. Diseño de base de datos relacionales. Capítulo 3: Sistemas de bases de datos. [En línea] 2005 Disponible en:http://www.emagister.com/curso-diseno-base-datos-relacionales/sistemas-bases-datos. Consultado: 2013-01-28.

Silva Viera Diego. Diferencias entre los SGBDS´s [En línea] 2012.Disponible en:http://www.slideshare.net/diegosilvaviera1/diferencias-entre-los-sgbase de datos. Consultado: 2013-

02-01.

Blaha Michael. Patterns of data modeling. 2010.

Gimson Loraine. Metodologías ágiles y desarrollo basado en conocimiento. 2012.

Herramientas Case Disponible en: www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf. Consultado: 2013-02-05.

Instituto Tecnológico de Informática. Testeo de estrés y carga. [En línea] diciembre 2010. Disponible en: http://www.iti.es/servicios/servicio/resource/7240/index.html.

Minay Carlos. Modelos de Bases de Datos. [En Línea] 2009. Disponible en http://es.scribase de datos.com/doc/17170125/Modelos-de-Bases-de-Datos. [Consultado] 9-12-2012.

Lic. Rosa María Mato García. Diseño de Bases de Datos. [En línea] octubre 1999.Disponible en:http://eva.uci.cu/mod/resource/view.php?id=11576&subase de datosir=/Bibliografia_Basica. Consultado: 2013-03-22.

Portal en español de PostgreSQL. Integridad referencial con PostgreSQL [En línea] 2009. Disponible en: http. Consultado: 2013-03-26.

Anna Salazar Grau. Estructura de las bases de datos. [En línea] 2010.disponible en:http://www.slideshare.net/142918/estructura-de-las-bases-de-datos-5670614. Consultado: 9-12-2012.

Salazar. Sistema Gestor De Base De Datos. [En línea] 2009. Disponible en:http://www.slideshare.net/beatriz1019/sistema-gestor-de-base-de-datos-beatriz Consultado: 11/12/2012.

Jorge Sánchez Asenjo. Sistemas gestores de Bases de Datos[En línea] 2009 disponible en:ubuntuone.com/p/sqt/ [En línea].consultado 11/12/2012.

SGBASE DE DATOS. Sistemas gestores de bases de datos (primera parte)**.**Capítulo 6**:** SGBASE DE DATOS. Componentes y lenguajes. Editorial McGraw-Hill [En línea] 2008. Disponible en: http://www.emagister.com/curso-sistemas-bases-datos/sgbase de datos-componentes-lenguajes. Consultado: 22/01/2013.

BIBLIOGRAFÍA

Fidel Gil, Javier Albrigo, Javier Do Rosario SISTEMAS DE GESTIÓN DE BASE DE DATOS / DBMS [En línea] 2005 Disponible en:www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r61632.PDF.Consultado: 28/01/2013.

Carlos Alberto García Chávez. Diseño de base de datos relacionales. Capítulo 3: Sistemas de bases de datos [en línea] 2005 Disponible en:http://www.emagister.com/curso-diseno-base-datos-relacionales/sistemas-bases-datos . Consultado: 28/01/2013.

Introducción a base de datos. Alejandro Alcalde [En línea] 2010. Disponible en:http://elbauldelprogramador.com/programacion/basededatos/introduccion-base-de-datos/. Consultado: 29-01-2013.

Vázquez Lucía. Ventajas y desventajas de PostgreSQL [En línea] 2012.Disponible en: http://www.tecnologiaslibres.com/portal/content/view/19395/52/. Consultado: 03-02-2013.

Silva Viera Diego. Diferencias entre los SGBASE DE DATOS´s [En línea] 2012.Disponible en:http://www.slideshare.net/diegosilvaviera1/diferencias-entre-los-sgbase de datoss. Consultado: 2013-02-01.

Tecnologías Información. Integridad de Datos [En línea] 2009. Disponible:http://www.tecnologias-informacion.com/integridaddatos.html. Consultado: 2013-03-27

Hansen, Gary W; Hansen James V. Diseño y Administración de Bases de Datos. 2da Edición.
Lic. Rosa María Mato García. Diseño de Bases de Datos. [En línea] octubre 1999.Disponible en: http. Consultado: 2013-03-22.

Fernandez Del Monte Yusleydi. Planilla de Especificación de Requisitos. Habana 2012.Disponible en: Expediente de proyecto de Nova Qalit.

Office. Conceptos básicos del diseño de una base de datos [En línea] 2007 disponible en:http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx.

GLOSARIO DE TÉRMINOS

SQL: Structured Query Language (entendida en español como **Lenguaje de Consulta Estructurado**), el cual identifica a un tipo de lenguaje vinculado con la gestión de **bases de datos de carácter relacional** que permite la especificación de distintas clases de operaciones entre éstas.

Tuplas: se trata de cada una de las filas de la tabla. Es importante señalar que no se pueden tener tuplas duplicadas en una tabla.

Artefacto: es una pieza de información que es producida o utilizada por procesos. Los artefactos son los elementos tangibles de un proyecto, elementos que el proyecto produce o usa mientras se trabaja en busca del producto final. Éstos, pueden tomar varias formas y formatos, como por ejemplo

- Un documento, tal como la visión o la lista de riesgos.
- Un modelo, por ejemplo un diagrama de casos de uso o el modelo de diseño.
- Un elemento dentro de un modelo, tal como una clase, un caso de uso o un subsistema
- Ejecutables, por ejemplo el ejecutable del prototipo
- Código fuente.

Arquitectura de Software (AS): se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos, idealmente, se crea en etapas tempranas del desarrollo.

Redundancia: Repetición de una información previamente existente**.**
Rol: Papel desempeñado por las personas en la sociedad.
Atributo: se trata de cada una de las columnas de la tabla. Vienen definidas por un nombre y pueden contener un conjunto de valores.

Entidad: es la representación de un objeto o concepto del mundo real que se describe en una base de datos, se describe en la estructura de la base de datos empleando un modelo de datos. Por ejemplo, nombres de entidades pueden ser: Alumno, Empleado, etc. Cada entidad está constituida por uno o más atributos. Por ejemplo, la entidad "Alumno" podría tener los atributos: nombre, apellido, año de nacimiento.

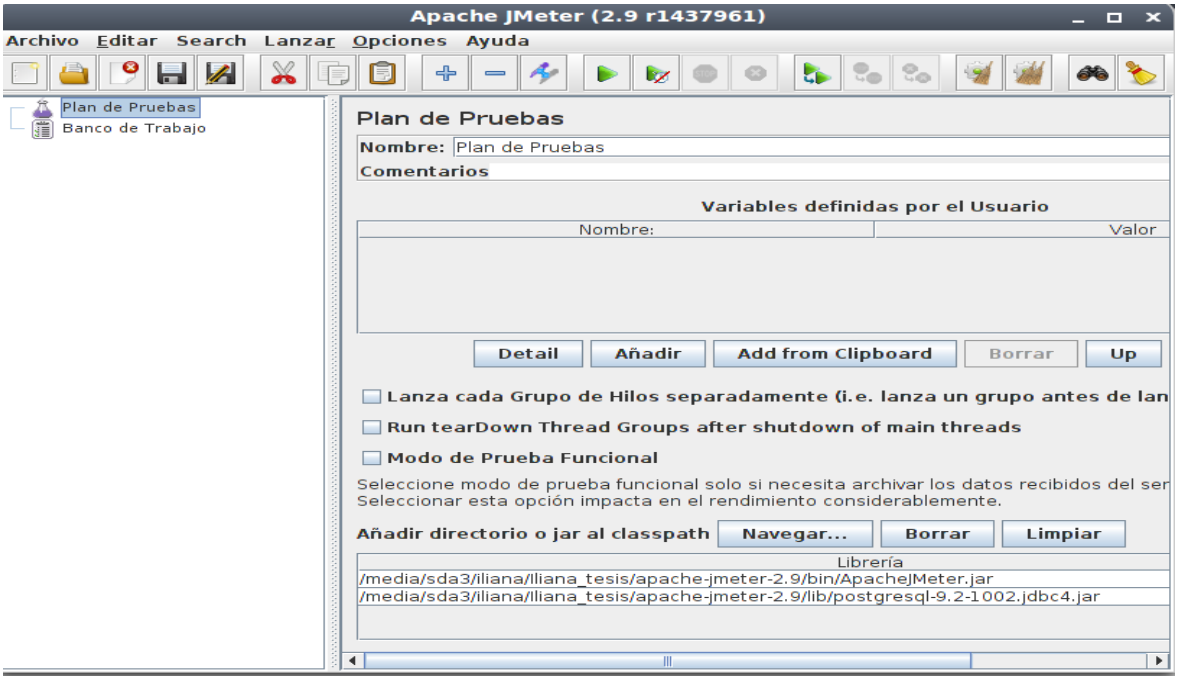
GLOSARIO DE TÉRMINOS

Nomencladores: son aquellas tablas que permiten realizar enumeraciones de un concepto dentro del universo de información, estas deben ser finitas y conocidas, ejemplo: se tiene una tabla persona la misma cuenta con una categoría (estudiante/ profesor), para representar la categoría de la persona se utiliza un tabla nomenclador (categoría_persona) va almacenar los dos tipo de categoría.

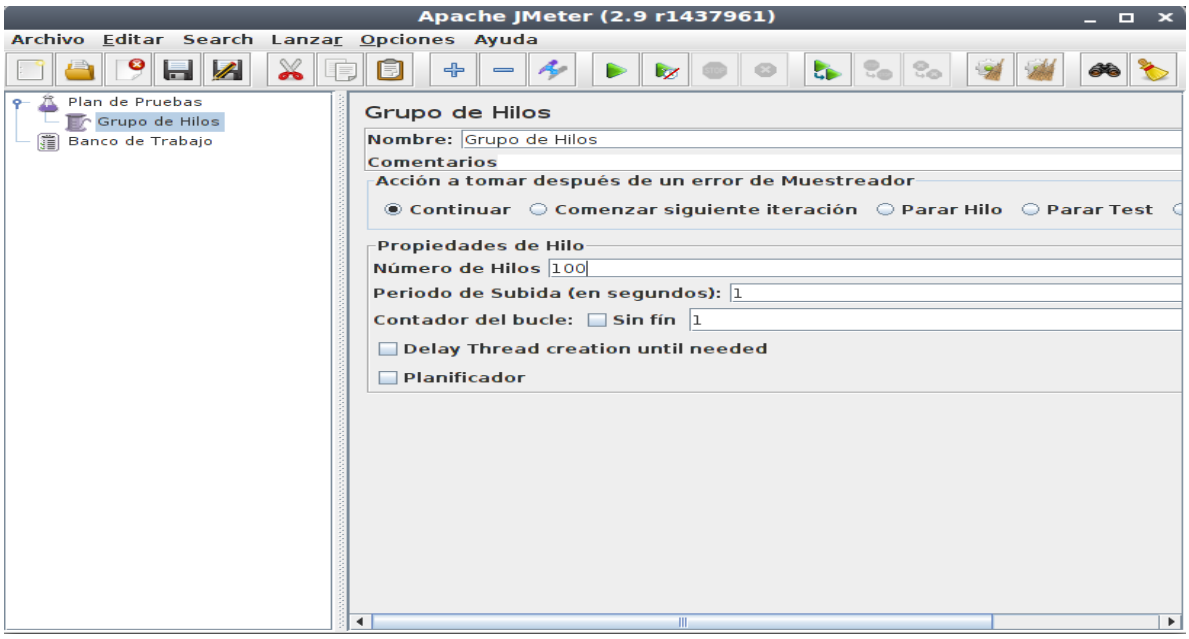
ANEXOS

Pasos para realizar pruebas de carga y estrés con la herramienta JMeter

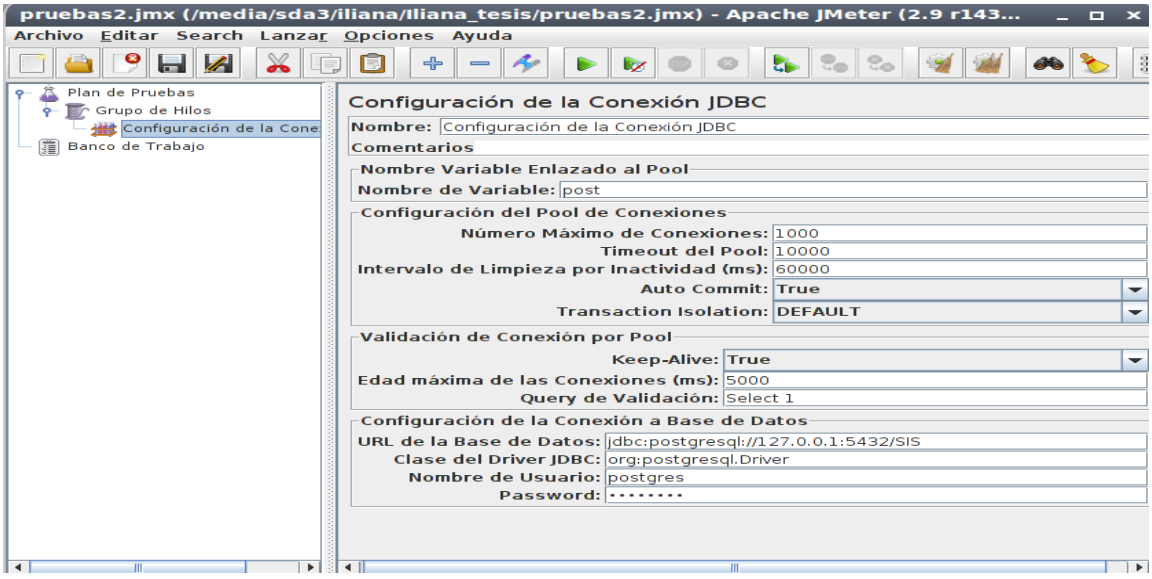
1. Contar con la herramienta JMeter en alguna de sus versiones, en este caso se utilizó 2.9
2. Se adicionan al Plan de Prueba los archivos .jar requeridos como lo muestra la siguiente imagen



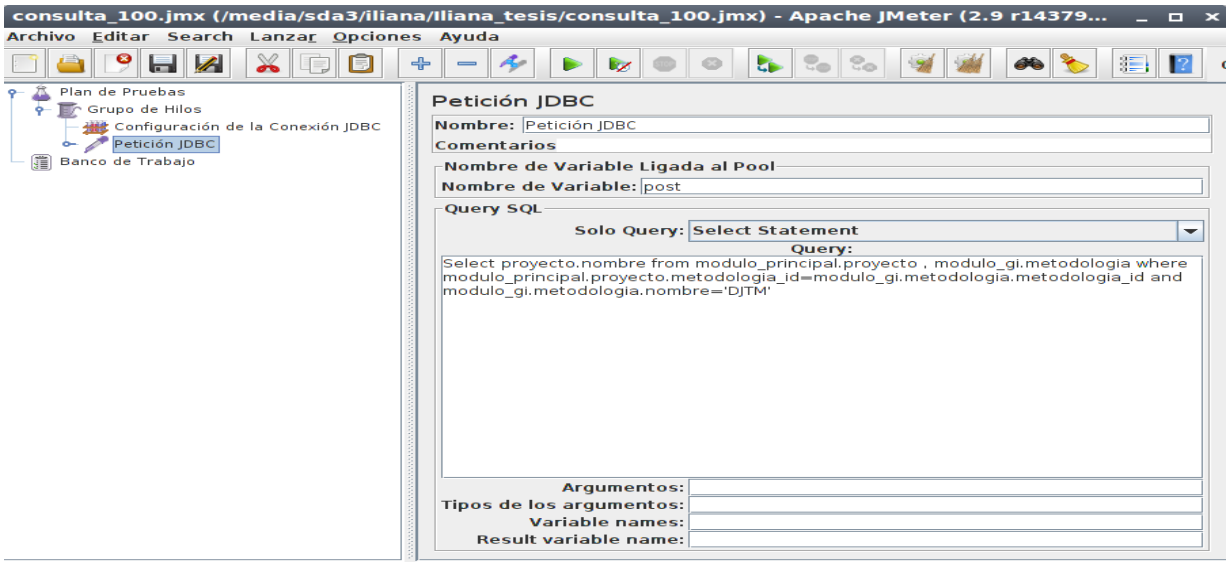
3. Siguiendo la ruta Plan de Prueba/Añadir/Grupo de Hilos, se define la cantidad de usuarios concurrentes o número de hilos, además del período de subida(en segundos), que es el tiempo en segundo que se va a demorar el JMeter en lanzar todos los hilos (en este caso se seleccionan 100 hilos y el período de subida es de 1 segundo, entonces cada hilo se ejecutará 0,1 segundo después de que el hilo anterior haya sido lanzado); se le dio valor 1 al controlador de bucles, encargado de realizar tantas veces como se le especifica con el número.



4. Trazar la ruta Grupo de Hilos/Añadir/Elementos de Configuración/Configuración de la Conexión JDBC.



5. Trazar la ruta Grupo de Hilos/Añadir/Muestreador/Petición JDBC. En el campo nombre, se elige el nombre de la petición, y en nombre de variable se le asigna el mismo nombre que se definió en la configuración de la conexión jdbc. Posteriormente se especifica el tipo de consulta realizar.



6. Elegir la ruta Grupo de Hilos/Añadir/Receptor/ [Informe Agregado, Grafico de Resultados, Ver Árbol de Resultados]. Estas opciones permiten graficar de forma fácil los resultados mediante las etiquetas definidas a la hora de realizar las pruebas de carga y estrés.
7. Finalmente se ejecuta la herramienta y se valoran los resultados expuestos.