Universidad de las Ciencias Informáticas

Facultad 3

Diseño e Implementación de la base de datos para el subsistema Planificación material y financiera, del paquete de aplicaciones ERP-Cuba, Cedrux.

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

Autor: Sisley Jiménez Martínez

Tutor: Ing. Drisis Silvia Díaz Rodríguez

Co-Tutor: Ing. Igniris Valdivia Báez

Ciudad de la Habana, junio 2011.
"Año 53 de la Revolución"

Declaración de autoría

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

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

_____ _____

Sisley Jiménez Martínez — Autor

Drisis Silvia Díaz Rodríguez — Tutor

"Aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de si mismos."

Ernesto Guevara de la Serna

Datos de contacto

Tutor: Ing. Drisis Silvia Díaz Rodríguez

Correo: dsdiaz@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Es instructor recién graduado, lleva dos años de graduado, y 2 años de experiencia en el tema.

Co-tutor: Ing. Igniris Valdivia Báez

Correo: ivaldivia@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2010.

Agradecimientos

Le doy gracias a las personas que han creído en mí, incluso, más de lo que deberían. Gracias por todos los sacrificios que han hecho, sin yo pedirlo y tal vez sin merecerlo. Esas personas son las que me impulsan a hacer, a tratar de hacerlo bien, de hacerlo mejor. Gracias por estar ahí siempre.

A todos los que me han ayudado a cumplir este sueño.

A la Revolución por darme la oportunidad.

A mis tutoras, por la preocupación y ocupación, sin ellas no lo hubiera logrado.

A mis amigos y amigas, las de antes, las de ahora, las de toda la vida, por ser mi paño de lágrimas, mis consejeras, mis defensoras, por no tener la obligación de estar y haber estado, por ser simplemente, mis amigas.

A aquellos que hoy forman parte importante de mi vida, con los que he compartido momentos únicos, a los que no quiero dejar de ver, de los que no quiero separarme.

A los que ya no están y sin embargo, estarán siempre, a mi abuelo Pine, por haberme querido tanto, por haberme dado tanto.

A mi familia que tantos sacrificios ha hecho, que tanto me ha impulsado, a mis tías, abuelas, abuelos, primos, por ser mi ejemplo, mi inspiración y mi fuerza.

A mi hermana por el amor que me inspira y por ser la razón por la que debo y me esfuerzo cada día.

A mis padres porque gracias a ellos estoy aquí, a mi papá por creer en mí, a mi mamá por todos los sacrificios, porque le debo lo soy.

Dedicatoria

A todos los que hacen suyos mis triunfos.

A todos los que puedan sentirse orgullosos.

A todos los que me quieren y a los que quiero, a mi hermana, a mis padres, a mi mamá.

Resumen

En la actualidad el avance que se produce en temas de informática y debido a sus enormes beneficios hace que las instituciones busquen alternativas para que su desempeño sea más eficiente y cómodo. Esto provoca que se revolucionen las soluciones informáticas en cualquier ámbito entre ellas las relacionadas con los Sistemas Integrales de Gestión, siendo en la Universidad de las Ciencias Informáticas (UCI) el proyecto Cedrux el encargado del desarrollo de una variante de ellos. El subsistema de Planificación pretende consolidar una solución con la que se pueda estructurar, organizar y controlar la planificación en cualquier entidad del país, por lo cual se hace necesario contar con una base de datos que permita la recopilación, control y organización de la información deseada para que pueda ser encontrada y utilizada fácilmente.

El presente trabajo de diploma se centra en el diseño de una base de datos para el subsistema de Planificación del proyecto Cedrux, incluyendo buenas prácticas para la obtención de un diseño exitoso. El desarrollo del trabajo fue guiado por las actividades establecidas y por la utilización de las herramientas especificadas por la dirección del proyecto, tales como: herramientas de diseño, gestor de base de datos y pruebas de concepto sobre las soluciones propuestas.

Se obtuvo como resultado un modelo de datos que responde a las necesidades planteadas, el cual ha sido validado teórica y funcionalmente teniendo en cuenta un conjunto de aspectos importantes para el área de las bases de datos relacionales como son la normalización, redundancia, integridad y seguridad.

Palabras claves: base de datos, Cedrux, subsistema.

Índice

Introducción

La revolución tecnológica en que se encuentra inmerso el mundo de hoy hace que las empresas, tratando de mantenerse en el mercado, busquen el perfeccionamiento mediante el correcto manejo de sus recursos, el uso racional de los mismos puede causar una disminución de sus debilidades y un aumento de sus fortalezas, mantener eficientemente grandes volúmenes de datos le permite administrar y supervisar los recursos accediendo a toda la información de forma confiable, precisa y oportuna, lo que contribuye a solucionar gran parte de sus dificultades optimizando los procesos. Esto se logra a través de los *sistemas de planificación empresariales* (en adelante ERP, según sus siglas en inglés), siendo sistemas de gestión de información que integran y automatizan muchas prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Estos sistemas integrales de gestión están compuestos por diferentes subsistemas o módulos integrados en una aplicación, entre ellas logística, contabilidad y planificación. Este último se refiere a las acciones llevadas a cabo para realizar planes y proyectos de diferentes índole. Planificar significa que los ejecutivos estudian anticipadamente sus objetivos y acciones, y sustentan sus actos no en corazonadas sino con algún método, plan o lógica.

Para los subsistemas integrados a un ERP, la gestión de la información es vital, lo que implica que necesiten aplicaciones como los Sistemas Gestores de Bases de Datos (en adelante SGBD), que les permitan manejar de manera clara, sencilla y ordenada un conjunto de datos.

Cuba, en particular la Universidad de las Ciencias Informáticas, se encuentra desarrollando un ERP, Cedrux, bajo la premisa que el sistema pueda implementarse sobre plataformas de Software Libre, lo que lo pondría un paso adelante respecto a otros ERP en explotación en el país. El mismo cuenta con varios subsistemas que necesitan gestionar grandes cúmulos de información, entre ellos el subsistema de Planificación material y financiera.

El proceso de planificación que tiene lugar hoy día en el país presenta dificultades, pues a la hora de manejar la información, los datos se encuentran en fuentes no seguras, ya sea en formato duro como hojas de papel o de manera digital como Microsoft Excel provocando la posibilidad de modificaciones no deseadas, lo cual atenta contra la integridad y confiabilidad de estos, se suma que los datos no se

encuentran centralizados lo que conlleva al descontrol y a la desorganización de los mismos. Además se invierte gran cantidad de horas hombres en la búsqueda y consulta de esta información.

Se hace necesario que el subsistema Planificación material y financiera cuente con una estructura de datos que permita la organización, control y almacenamiento de la información, situación que engloba el **problema a resolver**.

Para darle cumplimiento a lo anterior se plantea, como **objetivo general,** realizar el diseño y la implementación de una BD para el subsistema Planificación Material y Financiera que cubra los requerimientos definidos en la etapa de diseño de la solución.

Lo anterior da lugar a los siguientes **objetivos específicos**:

- Analizar tendencias actuales en cuanto al diseño de BD.
- Diseñar la BD para el subsistema Planificación Material y Financiera del paquete de aplicaciones ERP-Cuba, Cedrux.
- Implementar el acceso a datos.
- Realizar la validación teórica y funcional del diseño.

Como **objeto de estudio** se define el diseño e implementación de BD, delimitando el **campo de acción** a las estructuras de datos en el proceso de Planificación material y financiera.

El diseño y la implementación de una base de datos que permita la gestión de la información reduciendo la redundancia, garantizando la integridad y aumentando el rendimiento de los datos, contribuirá a administrar y controlar los recursos con mayor efectividad, constituyendo esta la **idea a defender**.

Métodos científicos: La investigación está sustentada en los métodos teóricos y los métodos empíricos, que facilitarán la investigación y servirán de guía para organizar mejor el trabajo y de esta forma posibilitar entender el problema, estudiarlo, analizarlo y llegar a conclusiones para la solución. A continuación se explica la utilización de los que han sido seleccionados:

Métodos empíricos:

- Entrevista: utilizado con el objetivo de obtener la información necesaria para diseñar e implementar la BD.

Métodos teóricos:

- Analítico – Sintético: Este método será utilizado para realizar un análisis de las tendencias actuales en cuanto al diseño de base de datos.
- Análisis Histórico – Lógico: Este método será utilizado en el estudio y análisis de las principales herramientas utilizadas para el manejo de datos.
- Modelación: Este método será utilizado en el momento de diseñar la BD, pues se debe definir un modelo de datos que cumpla con los requisitos necesarios.

Este trabajo está estructurado de la siguiente manera: Introducción, Desarrollo dividido en tres capítulos, Conclusiones, Recomendaciones, Bibliografía, Referencias bibliográficas, Anexos y Glosario de términos.

Capítulo #1: Fundamentación teórica.

- Estudio de sistemas de gestión de información.
- Definir herramientas y metodologías que se utilizarán en la propuesta de solución.

Capítulo #2: Análisis y descripción de la solución propuesta.

- Arquitectura de la propuesta de solución.
- Requisitos que se tienen en cuenta en el desarrollo de la propuesta de solución.
- Diagramas que muestran el diseño de la solución, además de la descripción de cada una de las tablas que los componen.

Capítulo #3: Validación del diseño.

- Se valida el diseño realizado de forma teórica teniendo en cuenta aspectos como la integridad, la normalización, la seguridad y cómo responde el sistema a las pruebas.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el siguiente capítulo se realizará un análisis de sistemas análogos con el objetivo de demostrar la necesidad de una nueva solución, se abordarán aspectos importantes de las BD y de los SGBD, así como elementos significativos relacionados con el diseño de BD, además se argumentará todo lo relacionado con las tecnologías y herramientas definidas por el proyecto para el desarrollo del sistema, con el objetivo que se tenga una visión general de lo que se usará para la obtención del producto.

1.2 Planificación en Cuba

Planificar consiste en establecer objetivos, estudiar la forma de conseguirlos, marcar un plan y controlar los resultados de forma sistemática. (1)

Se refiere a las actividades llevadas a cabo para realizar planes y proyectos de diferente índole. Opera la ejecución directa de los planes que serán realizados. Todo esto se realizará en las entidades que pueden ser: empresas, unidades presupuestadas u otro tipo de organización similar con una gestión económica, financiera, organizativa, técnica, productiva, comercial y laboral, con autonomía controlada, en cumplimiento de lo establecido por el Gobierno.

El proceso de planificación tiene sus diferencias entre los países capitalistas y los países con economía socialista, como es el caso de Cuba. En nuestro país la planificación socialista es un proceso técnico, económico y organizativo en el que se establecen los objetivos y estrategias de la organización a corto y mediano plazo y se definen las acciones y recursos para su cumplimiento de forma racional, constituyendo a la vez y sobre todo un proceso político-ideológico que expresa la voluntad de priorizar el aporte de las empresas estatales a la sociedad por encima de cualquier interés colectivo o individual y asegurar así el desarrollo de las empresas en correspondencia con los requerimientos de la economía nacional.(2)

1.3 ERP

El creciente desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) influye de manera directa en la forma en que las empresas dirigen sus estrategias de negocio y centran sus recursos en función de la mejora continua de sus procesos. Por tal motivo buscan mejorar los procesos utilizando sistemas que les permitan optimizar los recursos, como los Sistemas de Gestión de Recursos Empresariales.

La Planificación de Recursos Empresariales, o simplemente ERP (Enterprise Resourse Planning), es un conjunto de sistemas de información gerencial que permite la integración de ciertas operaciones de una empresa, especialmente las que tienen que ver con la producción, la logística, el inventario, los envíos y la contabilidad. (3)

Un ERP es un sistema o software administrativo que integra todas las áreas de una empresa (Como contabilidad, compras, o inventarios), mediante procesos transparentes y en tiempo real en bases de datos relacionales y centralizadas. (4)

Básicamente son aplicaciones que integran los aspectos funcionales de la empresa: gestión comercial, gestión financiera, gestión de producción, logrando así que se produzca un ahorro considerable de tiempo y que se minimicen los errores en gran medida.

A continuación se realiza un análisis atendiendo a las soluciones de diversos ERP.

1.3.1 Sistemas internacionales

La siguiente tabla muestra los sistemas de gestión a nivel internacional que serán analizados.

OpenBravo ERP		**España**
Open-ERP		**Argentina**
OpenXpertya		**España**

Tabla 1: Sistemas internacionales.

OpenBravo ERP

Es un sistema de gestión empresarial integrado (ERP) desarrollado en software libre y basado íntegramente en web, que ha sido desarrollado siguiendo el modelo MVC (Modelo Vista Controlador). Implementado en el lenguaje Java, presenta soporte para bases de datos PostgreSQL y Oracle, se ejecuta sobre Apache y Tomcat. La estructura de datos de la aplicación está basada originalmente en una versión antigua de Compiere.

Una de las principales ventajas con las que cuenta este ERP es que sigue un licenciamiento de software libre lo cual permite adquirirlo sin costo alguno, o la necesidad de depender de algún proveedor de servicio. Además la licencia del producto asegura el acceso público al código fuente y la posibilidad de modificar dicho código libremente adaptándolo a las necesidades de tu empresa.(5)

Como desventaja tiene que no presenta un módulo que garantice la planificación de forma centralizada, además se enfoca en empresas capitalistas con modelos económicos distintos al de empresas cubanas.

Open-ERP

Es un sistema de gestión de empresas de licencia libre, desarrollado en Argentina, que cubre las necesidades de las áreas de contabilidad, ventas, compras, almacén, inventario, proyectos, recursos

humanos y tiendas virtuales. Soporta múltiples monedas y múltiples entidades. Incorpora funcionalidades de gestión de documentos, conexiones con otras aplicaciones y permite trabajar remoto mediante una interfaz web o aplicación de escritorio multiplataforma.(6)

Incluye un entorno modular de programación/adaptación rápida de aplicaciones. Se basa en tecnología Python/XML trabajando sobre una base de datos PostgreSQL. Como ventaja principal se puede señalar su sencillez, pues brinda la posibilidad de comenzar con el módulo que necesites y crecer según las necesidades lo requieran, además de la escalabilidad ya que no depende del tamaño de la empresa.

A pesar de que cuenta con una amplia gama de módulos, ninguno comprende los procesos que tienen lugar dentro de la planificación material y financiera de nuestro país, como por ejemplo el cálculo de necesidades.

OpenXpertya

Es un software tipo ERP (Enterprise Resourse Planning) de características técnicas realmente sobresalientes: el código del servidor de aplicaciones, de los interfaces, de la aplicación B2B, B2C y B2E y del cliente fue desarrollado en J2EE, y es posible utilizar varios motores de base de datos; inicialmente, Oracle, y en siguientes versiones un amplio elenco de bases de datos de código libre. Desarrolla la arquitectura Cliente - Servidor en un modelo original de tres capas. (7)

OpenXpertya, a través del entorno de diseño en tres capas aporta una metodología de declaración de los conceptos de negocio, definición de la interacción con el sistema, procesos a realizar sobre los conceptos, y finalmente nos permite establecer restricciones a este modelo y validaciones. (7)

En la capa de datos tenemos el motor de base de datos relacional, independiente de la aplicación y escalable en función de las necesidades de la empresa final. Inicialmente se trabajó sobre Oracle, por su potencia y por ser un estándar del mercado, pero adicionalmente hay disponibilidad para la utilización de otros motores de base de datos, que pese a no disponer de tanta fiabilidad, tienen como baza a su favor la disponibilidad absoluta del software libre. (7)

Desde su inicio, openXpertya ha sido desarrollado como Código Abierto (*Open Source*). La principal y obvia ventaja monetaria es que no existen costos de licencia para el producto en sí mismo. El mayor

diferenciador de todos modos es que se puede, además, obtener el código fuente. Esto le brinda al usuario una independencia absoluta del proveedor. De este modo la empresa usuaria no depende de su existencia y prioridades, pudiendo cambiar de consultoría si lo deseara o asumir internamente el mantenimiento o desarrollo de la aplicación.(7)

Tiene como desventaja que no tiene implementada algunas funcionalidades como las relacionadas con la gestión de recursos humanos, otra de las desventajas es que para actualizarlo a nuevas versiones, es necesario descargar el producto completo y volverlo a instalar, en lugar de gestionar dichas actualizaciones desde el mismo producto, actualizando sólo aquellas partes del programa que hayas cambiado. (6)

1.3.2 Sistemas nacionales

Versat Sarasola		**Cuba**

Tabla 2: Sistema nacional.

Versat Sarasola

La filial en Villa Clara de TEICO, la empresa del MINAZ encargada de la Informática y las Comunicaciones, aglutinó y mejoró un grupo de programas aislados (Contabilidad; Finanzas; Control de Inventarios; Control de Medios Básicos; Nómina de Salarios; Planificación). En julio de 1999 comenzó la elaboración del VERSAT Sarasola. Dos años después, el central George Washington se convertía en la primera entidad que utilizaba el sistema.(8)

Es un sistema integrado de gestión económica diseñado para ser utilizado de acuerdo a las características de cada entidad, pues es configurable por cada una de ellas en el momento de su instalación y tiene como objetivo fundamental permitirle a los directivos analizar, controlar y evaluar los resultados de su negocio o actividad en tiempo real, al contar con un instrumento seguro, rápido, eficaz y de fácil manejo para la planificación, control y el análisis de la gestión económica y financiera.

Es una aplicación de escritorio implementada en Delphi, concebida sobre una plataforma de trabajo Cliente-Servidor lo que permite su instalación en red por las posibilidades que esta tecnología facilita para el trabajo en un entorno multiusuario. Trabaja sobre el sistema operativo Windows y es implementado sobre el SGBD SQL Server 2000.

Es una de las soluciones más utilizadas en el país la cual ha ahorrado importantes sumas de dinero y a pesar de incluir funcionalidades necesarias no contiene determinados procesos que hoy se hacen imprescindibles, no permite realizar el desglose al nivel de detalle que hoy se necesita, específicamente no permite gestionar las columnas de los modelos utilizados en un plan, ni realizar el cálculo de necesidades. Además de estar soportado sobre tecnología privativa.

1.3.3 Resultado

Los ERP analizados se han desarrollado teniendo en cuenta escenarios y necesidades referentes a la situación en la cual interactúan sus creadores, por lo mismo la mayoría no responden a los objetivos económicos y sociales que demanda el desarrollo de la sociedad cubana actual. De ahí que las soluciones de los sistemas presentados no cubren en su totalidad las necesidades inminentes hoy en el país, por un lado se dificulta el pago de las licencias de los basados en tecnologías propietarias, teniendo por otra parte que los basados en software libre no se ajustan a las necesidades o condiciones económicas que el país presenta. Se suma que algunos usan tecnologías que pueden ser riesgosas en la situación actual de bloqueo que sufre nuestro país, por ejemplo OpenXpertya está basado en la plataforma J2EE cuya máquina virtual es propiedad de SUN que es una empresa norteamericana y aunque haya comenzado a liberar el código de esta máquina virtual sigue estando bajo las leyes de su gobierno que bloquea por todos los medios el acceso a tecnología informática, además J2EE requiere un consumo de memoria elevado en comparación con otras plataformas como pudiera ser Python/Zope o PHP/Apache. (9)

Lo anterior desencadena la necesidad de aparición de un producto que minimice estas deficiencias tomando como base las ventajas de cada uno de ellos en pos de obtener un sistema de alta calidad que cubra en su mayor parte los requerimientos existentes.

1.4 Definiciones: bases de datos, modelo de datos y SGBD.

1.4.1 Base de datos

Una BD es una agrupación de información perteneciente a un mismo contexto y almacenados sistemáticamente para su posterior uso, constituyendo la base fundamental de cualquier sistema de información. Debido al constante desarrollo tecnológico la mayoría de las BD hoy en día se encuentran en formato digital. Permite guardar gran cantidad de información de forma organizada, funcionando como un enorme almacén para que en el momento deseado pueda ser utilizada con facilidad, permitiendo su recuperación rápida y segura.

Las BD son un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo.(10)

Es posible considerar a la base de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos. (11)

1.4.2 Modelo de datos

El modelo de datos se orienta a describir una BD, permite describir los elementos, características fundamentales de un problema determinado y cómo se relacionan estos elementos entre sí.

Algunas de las definiciones son:

- Es un lenguaje utilizado para la descripción de un BD. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base). (12)
- Un modelo de datos es básicamente una “descripción” de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. (11)

La clasificación de los modelo de datos se realiza de acuerdo al nivel de abstracción. Los modelos de datos físicos, son estructuras de datos a bajo nivel implementadas dentro del propio manejador. (12)

Modelo conceptual

Los modelos de datos conceptuales son aquellos que describen las estructuras de datos y restricciones de integridad. Se utilizan durante la etapa de análisis de un problema dado y están orientados a representar los elementos que intervienen y sus relaciones. Es independiente del SGBD que se pretende utilizar, es más cercano al usuario.

El resultado de la etapa del diseño conceptual se expresa mediante un modelo de datos de alto nivel. Uno de los más usados es el modelo Entidad–Relación.

Mediante este modelo se pretende visualizar los elementos que pertenecen a una BD, que reciben el nombre de entidades, las cuales se corresponden con el concepto de clase de la Programación Orientada a Objeto (POO) y donde cada tupla de una futura relación representaría un objeto de la POO.

Modelo lógico

Los modelos de datos lógicos se centran en las operaciones y se implementan en el algún manejador de BD. Representan abstractamente toda la información almacenada en la BD. El modelo lógico es más cercano al ordenador y depende del SGBD que se vaya a utilizar. El resultado del mismo puede ser el modelo relacional, el modelo de red o el modelo jerárquico. Es el momento en que se realiza el proceso de normalización.

Modelo físico

Como resultado tiene la implementación de la base de datos con todas sus restricciones. Considera las estructuras de almacenamiento y los métodos de acceso necesarios para proporcionar un acceso eficiente a BD.

Es necesario tener claro cuál es el SGBD a usar pues este modelo se debe adaptar a él. Es importante en esta etapa asegurar la BD estableciendo por ejemplo los permisos para los usuarios, así como la creación de vistas y la selección de índices para acelerar el acceso.

1.4.3 Sistema Gestor de Base de Datos (SGBD)

Es el responsable de tratar todas las peticiones de información de los usuarios. Es un software que sirve de interfaz entre la BD, el usuario y las aplicaciones que la utilizan, que pretende manejar de manera clara, sencilla y ordenada un conjunto de datos.

Un sistema de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones.La información en cuestión puede ser cualquier cosa que sea de importancia para el individuo u organización; en otras palabras, todo lo que sea necesario para auxiliarle en el proceso general de su administración. (11)

El software que permite la utilización y/o actualización de los datos en una o varias BD, desde diferentes puntos de vista a la vez, por uno o varios usuarios, se denomina Sistema Gestor de Base de Datos. (11)

Este sistema es un conjunto de programas de propósito general, que permite a los usuarios controlar el acceso y la utilización de la base de datos para incluir, modificar o recuperar información, incluyendo prestaciones con el fin de conseguir la independencia, integridad y seguridad de los datos y la concurrencia de los usuarios. (13)

Todo SGBD debe permitir crear y mantener una BD permitiendo especificar tipos, estructuras y restricciones, realizar consultas, actualizarlas y generar informes. Estos sistemas deben tener características u objetivos que deben cumplir, entre ellos:

- Control de la redundancia: la redundancia puede tener efectos negativos o no deseados, pues puede provocar inconsistencias en los datos o duplicar el trabajo al actualizar.
- Independencia: la independencia de los datos consiste en la capacidad de modificar el esquema lógico o físico de una BD sin tener que realizar cambios en las aplicaciones que sirven de ella.
- Seguridad: se debe garantizar que la información se encuentra segura, estableciendo los permisos de acceso y autorización a los usuarios y grupos correspondientes.

1.5 Clasificación de los SGBD

Se muestra en la siguiente tabla algunas de las clasificaciones de los SGBD.

Clasificación de los SGBD			
No. de usuarios a los que le da servicio	Monousuario	Multiusuario	
No. de sitios en los que se guarda la información	Distribuido	Centralizado	
Modelo de administración de datos	Jerárquico	En red	Relacional

Tabla 3: Clasificaciones de los SGBD.

1.5.1 Jerárquico

Éstas son bases de datos que almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol, en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*. Una de las principales limitaciones de este modelo, es su incapacidad de brindar eficientemente una solución a la redundancia de datos. (14)

El modelo jerárquico fue desarrollado para permitir la representación de aquellas situaciones de la vida real en las que predominan las relaciones de tipo 1:M, y también las relaciones de tipo 1:1.

Una base de datos de este tipo, no permite el acceso directo a las instancias de un hijo, si no es seleccionando previamente las instancias de los padres de los que depende. Siendo uno de sus problemas.

1.5.2 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). (14)

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.

1.5.3 Relacional

Uno de los puntos fuertes en el modelo relacional es la sencillez de su estructura lógica donde todos sus datos están estructurados en forma de tablas formadas por columnas y filas. Este modelo es el más utilizado debido a su rápido entendimiento por parte de los usuarios que no tienen conocimientos profundos sobre diseño de bases de datos.

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de entidades (tablas), compuestas por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla). (14)

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

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.

Edgar Frank Codd definió las bases del modelo relacional a finales de los 70. En las bases de Codd se definían los objetivos de este modelo:

- Independencia física. La forma de almacenar los datos, no debe influir en su manipulación lógica.

- Independencia lógica. Las aplicaciones que utilizan la BD no deben ser modificadas porque se modifiquen elementos de la BD.
- Flexibilidad. La BD ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- Uniformidad. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas).
- Sencillez.

➢ Será éste modelo el que se utilizará en el presente trabajo.

1.6 Diseño de base de datos

1.6.1 Objetivos

Los objetivos del diseño de la base de datos son: (15)

- Representar los datos que requieren las principales áreas de aplicación y los grupos de usuarios, y representar las relaciones entre dichos datos.
- Proporcionar un modelo de datos que soporte las transacciones que se vayan a realizar sobre los datos.
- Especificar un esquema que alcance las prestaciones requeridas para el sistema.

El diseño de BD debe proporcionar un acceso fácil y rápido a los datos evitando duplicar la información innecesariamente, ya que requiere malgastar el espacio e incrementar la posibilidad de que se produzcan errores e incoherencias. Se debe almacenar sólo la información necesaria y que la misma sea correcta y completa es un punto importante garantizando así la exactitud e integridad de ésta. Debe soportar las operaciones que los usuarios necesiten realizar con los datos ajustándose a las especificaciones de los mismos.

1.6.2 Estrategias de diseño

Hay varias estrategias a seguir para realizar el diseño: de abajo a arriba, de arriba a abajo, de dentro a fuera y la estrategia mixta. La estrategia *de abajo a arriba* parte de todos los atributos y los va agrupando

en entidades y relaciones. Es apropiada cuando la base de datos es simple, con pocos atributos. La estrategia *de arriba abajo* es más apropiada cuando se trata de bases de datos complejas. Se comienza con un esquema con entidades de alto nivel, que se van refinando para obtener entidades de bajo nivel, atributos y relaciones. La estrategia *de dentro a fuera* es similar a la estrategia de abajo a arriba, pero difiere en que se parte de los conceptos principales y se va extendiendo el esquema para considerar también otros conceptos, asociados con los que se han identificado en primer lugar. La estrategia *mixta* utiliza ambas estrategias, de abajo a arriba y de arriba a abajo, con un esquema de divide y vencerás. Se obtiene un esquema inicial de alto nivel, se divide en partes, y de cada parte se obtiene un sub-esquema. Estos sub-esquemas se integran después para obtener el modelo final. (15)

- La estrategia que se utilizará en la confección del diseño del presente trabajo de diploma es la de arriba a abajo debido a la complejidad presentada facilitando así el trabajo.

1.6.3 Inconvenientes del diseño

Los problemas que pueden ocasionar un incorrecto diseño de bases de datos son (16):

- Redundancia: está presente cuando se repiten innecesariamente datos en los archivos que conforman la base de datos.
- Inconsistencia: ocurre cuando existe información contradictoria o incongruente en la base de datos.
- Dificultad en el acceso a los datos: debido a que los sistemas de procesamiento de archivos generalmente se conforman en distintos tiempos o épocas y ocasionalmente por distintos programadores, el formato de la información no es uniforme y se requiere establecer métodos de enlace y conversión para combinar datos contenidos en distintos archivos.
- Anomalías en el acceso concurrente: ocurre cuando el sistema es multiusuario y no se establecen los controles adecuados para sincronizar los procesos que afectan a la base de datos. Comúnmente se refiere a la poca o nula efectividad de los procedimientos de bloqueo.
- Problemas de seguridad: se presenta cuando no es posible establecer claves de acceso y resguardo en forma uniforme para todo el sistema, facilitando así el acceso a intrusos.

- Problemas de integridad: ocurre cuando no existe a través de todo el sistema procedimientos uniformes de validación para los datos.

1.6.4 Fases del diseño de BD

Las fases del diseño de bases de datos (17):

- Fase inicial: análisis de requisitos. Descripción de la información a gestionar y sus procesos. Entrevistas con usuarios y expertos.

  - Captación y Análisis de requisitos. Especificación funcional.

  Caracterizar de forma completa las necesidades de los usuarios de la BD, tanto en los datos como en las operaciones a realizar con los datos.

  Entrevistar los futuros usuarios de la BD para captar las necesidades.

  Como resultado se obtiene:

  - Necesidades de datos. Especificación de la información que se quiere guardar.
  - Necesidades de manipulación de datos. Especificación de las operaciones a realizar con los datos.

- Diseño conceptual: traducción del análisis de requisitos al esquema conceptual. Representación generalmente gráfica de las entidades y sus relaciones.

  - Modelo Entidad Relación.

  Una vez encontrado el modelo abstracto que se quiere utilizar, el diseñador aplica los conceptos de este modelo para traducir los requisitos de datos del usuario al modelo abstracto, formando el esquema conceptual de la BD.

  Validar el esquema conceptual sobre las transacciones especificadas en los requisitos funcionales (consultas, actualizaciones, borrados, etc.).

- Implantación en el gestor:
  - Diseño lógico: traducción del modelo conceptual al lenguaje de definición de datos del gestor correspondiente. Modelo relacional.
  - Diseño físico: determina la organización de archivos y las estructuras de almacenamiento interno.

1.7 Arquitectura en Cedrux

El proyecto Cedrux tiene una arquitectura basada en componentes favoreciendo la reutilización de todos sus elementos siendo una arquitectura completamente modular. Una de las ventajas que provee la misma es la posibilidad de prueba de los componentes de manera independiente antes de probar el conjunto, lo cual facilita la detección de alguna anomalía permitiendo corregirlo en el componente afectado sin necesidad de afectar otra parte del sistema.

El sistema está formado por varios subsistemas los cuales cuentan con una serie de componentes que pueden ser mejorados de forma independiente y reutilizados por otro subsistema. En cada componente se utiliza como patrón arquitectónico el Modelo-Vista-Controlador (MVC).

El patrón MVC es un patrón de arquitectura de las aplicaciones software que separa la lógica de negocio de la interfaz de usuario. Facilita la evolución por separado de ambos aspectos la reutilización y la flexibilidad. (18)

Modelo-Vista-Controlador: patrón que separa la interfaz de usuario, la lógica de negocio y los datos de una aplicación en tres componentes distintos: la vista, el controlador y el modelo. El controlador recibe una petición y es quien decide quien la lleva a cabo en la capa del modelo. Luego que el modelo termina las operaciones devuelve al controlador el control de ejecución, y éste envía los resultados a la vista.

Modelo: administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Mantiene el conocimiento del sistema. No depende de ninguna vista y de ningún controlador.

Vista: maneja la visualización de la información.

Controlador: analiza los mensajes de eventos que recibe el sistema y modifica u obtiene datos del modelo en respuesta a las peticiones del usuario. Evita poner código indebido en la capa impropia, por ejemplo instrucciones de base de datos (modelo) en interfaz de usuario (vista).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. En aplicaciones web la separación entre la vista y el controlador está claramente muy bien definida.

1.7.1 Arquitectura de Datos

La arquitectura de datos es la encargada en principio de las definiciones de lo referente a datos como: patrones, estándares o la nomenclatura y de la integración de los modelos.

Siguiendo el enfoque orientado a componentes se tiene al menos un esquema por cada subsistema tratando de esta forma de reducir al máximo las integraciones a nivel de datos entre entidades de distintos subsistemas, las cuales se resolverían a nivel de sistema.

1.8 Lenguaje de modelado

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y las distintas combinaciones de la disposición para modelar un diseño de software.

Lenguaje Unificado de Modelado (UML)

- Se utilizara el lenguaje de modelado UML como soporte para el modelo de la BD.

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmos haya generado en muchos años.

UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (19)

1.9 Frameworks

Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Doctrine

Doctrine es un potente y completo sistema ORM (en inglés Object Relational Mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Funcionalidades:

1- Exporta una base de datos existente a sus clases correspondientes.
2- Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (20)

- Se estará haciendo uso de Doctrine en su versión 1.2.1.

1.10 Herramientas

El desarrollo de las TIC ha puesto al alcance de muchos, poderosas herramientas que facilitan el desarrollo de productos de software. Esto contribuye a controlar con mayor efectividad los procesos permitiendo realizar las tareas o actividades de forma sencilla y rápida.

La decisión de las herramientas utilizadas en la solución de sistema fueron definidas por el equipo de arquitectura del proyecto Cedrux, partiendo de aquí se referenciará cada una de ellas.

Visual Paradigm

Visual Paradigm para UML es una herramienta ampliamente utilizada en el mundo del software que permite a los profesionales modelar sus diseños. Posibilita la integración de aplicaciones empresariales a las bases de datos. Es capaz de generar código e ingeniería inversa para lenguajes como el PHP. Permite manejar grandes estructuras de manera eficiente, solo requiere una configuración de escritorio común. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues. (21)

Ayuda a la construcción de aplicaciones de calidad de manera más rápida y con un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, además de generar código desde diagramas y generar documentación.

Entre sus principales características se pueden encontrar:

- Soporta aplicaciones web.
- Genera código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Multiplataforma: Soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OSX
- Modelado de Base de Datos: Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

Otros aspectos de importancia: es capaz de mantener los objetos físicos en relación con el modelo lógico, permite el trabajo con PostgreSQL, la universidad tiene la licencia para su uso y funciona en las dos plataformas de desarrollo (Linux y Windows).

- Se estará haciendo uso Visual Paradigm en su versión 6.0 ó superior.

EMS PostgreSQL Manager

Es una herramienta de gran alcance para la administración y el desarrollo del servidor de la base de datos de PostgreSQL, ya que ofrece varias herramientas, que complementan diversas funcionalidades, que dan gran alcance a los usuarios. (22)

PostgreSQL

PostgreSQL es una fuente poderosa, abierta al sistema de la base de datos correlativo. Posee más de 15 años de desarrollo activo y una arquitectura probada, ha ganado una reputación fuerte para la fiabilidad, integridad de los datos y exactitud. Corre en todos los sistemas operativos incluidos Linux y Windows. También apoya el almacenamiento de grandes objetos binarios, incluso imágenes, sonidos y videos. Tiene interfaces nativas de la programación para los lenguajes C++, Java.net, Perl, Python, Ruby, entre otros y una documentación excepcional. Entre sus principales características pueden encontrarse (23):

- Disponible totalmente sin costo alguno.
- Disponible para los Sistemas Operativos UNIX y Windows.
- Soporte total del Modelo Relacional de Bases de datos.
- Extensiones propias a SQL para realizar consultas sobre la base de datos.
- Dependencias entre objetos, integridad referencial.
- Soporta valores no atómicos como dominio de un campo.

Sistema de gestión de BD relacional, incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Es un poderoso manejador de base de datos de código abierto diseñado para administrar grandes volúmenes de datos.

- Se estará haciendo uso de PostgreSQL en su versión 8.3 ó superior.

1.11 Metodología de desarrollo

La metodología de desarrollo a utilizar será Proceso de Desarrollo de Software (PRODESOFT) desarrollada por el UCID (Unidad de Compatibilización e Integración de Software para la Defensa), se logra con la combinación entre los modelos basado en Componentes, Iterativo y el Incremental. Se emplearán las técnicas para crear prototipos, si son requeridas, para los requerimientos del usuario de los que no existe una visión clara por parte de estos, con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

El ciclo de vida del proyecto se considera como parte del ciclo de vida del producto, el cual representa un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en la adquisición, el desarrollo, la explotación y el mantenimiento de un producto.

El ciclo de vida de un proyecto de software desarrollado se descompone en el tiempo en cinco fases secuenciales que son: Inicio, Modelación, Construcción, Explotación Experimental, Despliegue. Al final de cada fase los representantes de los grupos de roles presentes en el proyecto realizan una evaluación para determinar si los objetivos se cumplieron y así presentar a Consejo Técnico Formal para su evaluación y dar paso o no a la fase siguiente.

Ilustración 1: Etapas del ciclo de vida de PRODESOFT.

La Arquitectura de Datos identifica y precisa las mejores clases de datos que apoyan las funciones del negocio definidas en el modelo de negocio. Es la primera de las arquitecturas a ser concretadas porque la calidad de los datos es el producto básico de la función de la Ingeniería de Software. La arquitectura de datos tiene como objetivo puntualizar los principales tipos y fuentes de datos necesarios para dar soporte a las actividades de la entidad, de manera que sean:

- Entendibles por los participantes.
- Completas y consistentes.
- Estables.

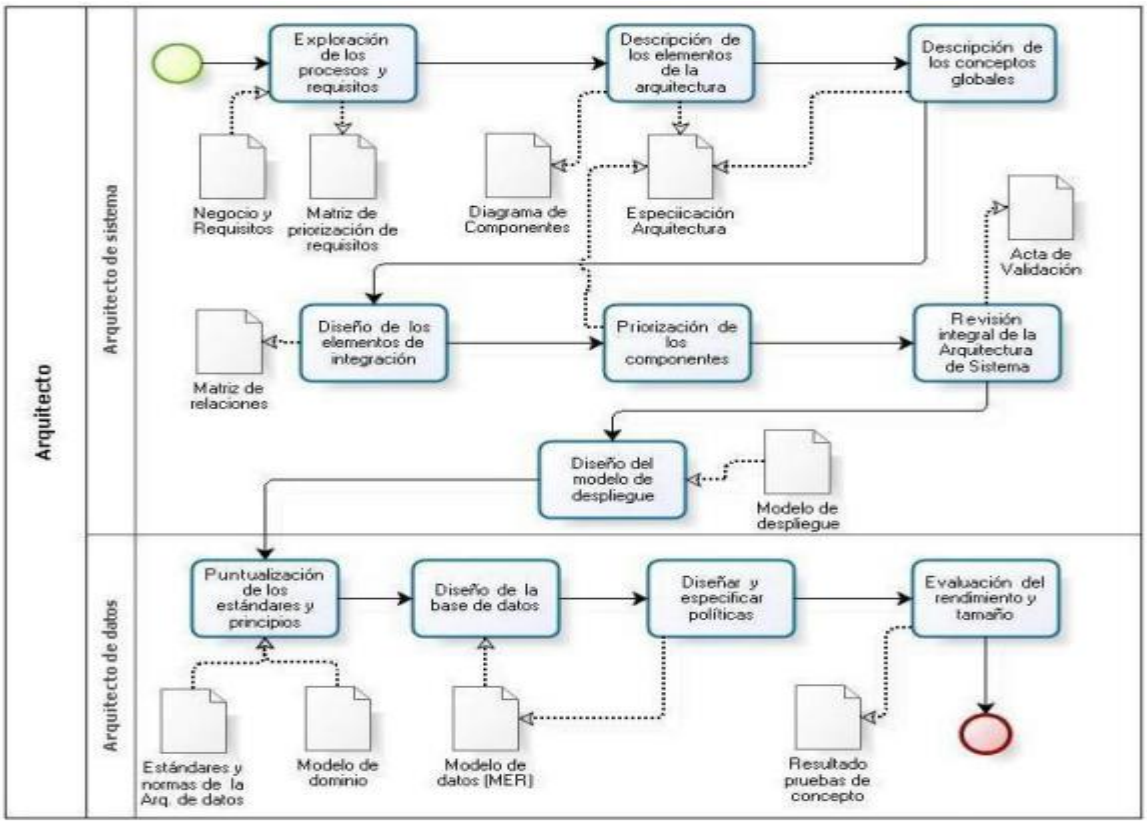


Ilustración 2: Actividades del diseño de la arquitectura.

1.12 Conclusiones parciales

Se abordaron en el capítulo elementos importantes relacionados con los aspectos fundamentales de los sistemas que han presentado soluciones similares en función de demostrar la necesidad de una nueva alternativa, se tocaron aspectos vinculados a BD así como la arquitectura en Cedrux y las herramientas utilizadas para alcanzar el resultado esperado, tratando de abarcar los elementos teóricos que sustentan la solución del problema permitiendo de esta forma sentar las bases teóricas para la correcta realización del trabajo que se pretende llevar a cabo.

Capítulo 2: Diseño de la solución

2.1 Introducción

En este capítulo se describen los procesos principales que tiene lugar para la construcción de la BD, entre los que se encuentran: la selección y argumentación de los requisitos, el desarrollo del modelo lógico y físico presentando el diagrama de clases persistentes y el diagrama Entidad-Relación, describiendo las clases y tablas principales obtenidas en ambos modelos.

2.2 Estándar de nomenclatura

Las reglas de nomenclatura para los objetos de la BD se realizó siguiendo el estándar definido por la UCID plasmadas en el documento: Políticas y Estándares del Rol Arquitecto de Datos.

2.3 Generación de llaves primarias

La creación de las pk únicas se hará de la siguiente forma:

Creación de una entidad nomencladora llamada nom_secuencias, en la misma se almacenan los campos que muestra la figura:

nom_secuencia		
+idsecuencia	**numeric(19, 0)**	**Nullable = false**
nombresecuencia	varchar(30)	Nullable = false
valorinicial	numeric(19, 0)	Nullable = false
valorincremental	numeric(1, 0)	Nullable = false
valorfinal	numeric(19, 0)	Nullable = false
nombreesquema	varchar(30)	Nullable = false

Ilustración 3: Tabla de nom_secuencias.

Nota: Este nomenclador es único para todos los esquemas y base de datos que intervienen, este nomenclador no se va a replicar, el nomenclador no debe tener el mismo nombre en todos los esquemas ya que puede dar error por tanto en cada esquema se va a llamar nom_secuenciasnombreesquema.

Ejemplo: nom_secuenciasplanificacion, es decir sin el prefijo mod (utilizado para establecer el nombre de los esquemas, por ejemplo: mod_planificacion).

idsecuencia: id tabla secuencial.

nombresecuencia: el nombre de la secuencia se insertará en dependencia a la tabla que corresponde esa secuencia. Este valor es único en la tabla, ejemplo.

Tabla persona - sec_persona_seq

Cuando PostgreSQL crea una secuencia nativa de ella le pone el sufijo _seq por defecto al final y para que Doctrine, al mapear, no presente problemas con las secuencias, también se le agrega este sufijo a las secuencias creadas.

valorinicial: valor que comenzará la secuencia.

valorincremental: valor en que va incrementado la secuencia, en este caso es en 1.

valorfinal: valor final que tomará ejemplo:

804999999999999999

nombreesquema: nombre del esquema donde se encontrará la secuencia, ejemplo**:**

mod_planificacion

Nota: el nomenclador se debe llenar manualmente o preparar un script que haga esta función en dependencia de las tablas a replicar en el esquema determinado.

idsecuencia	nombresecuencia	valorinicial	valorincremental	valorfinal	nombreesquema
1	sec_indicador_seq	8040000000000	1	80499999999999	mod_planificacion

Ilustración 4: Ejemplo de secuencia insertada.

Como se puede apreciar en el ejemplo anterior la semilla de la secuencia es formada según el valor inicial que se guarda en la tabla nom_secuencias.

Para generar las secuencias se tendrá una función la cual se ejecuta una sola vez cuando se está configurando la creación de las llaves primarias. El objetivo de esta función es concatenar las columnas guardadas en el nomenclador y formar la secuencia.

Secuencia en estructura de árboles

Cuando se inserta en una estructura de árbol un padre, el id padre que se le inserta es el mismo id de la llave primaria, esto presenta un problema porque ahora el id se genera solo y el programador no sabe cuál id insertar en idpadre, esto se solucionó de la siguiente forma.

El problema se resolvió creando una función disparadora, *"f_actualizacion_arbol",* la cual es llamada por un trigger, la misma tiene la función de buscar el nuevo valor de idpadre si es null es que no se insertó y sería padre de ella misma, por lo que se le asigna el valor de id de la tabla.

2.4 Soluciones para el diseño en Cedrux

2.4.1 Árboles

La siguiente figura muestra la representación de un árbol siguiendo el modelo de conjuntos anidados

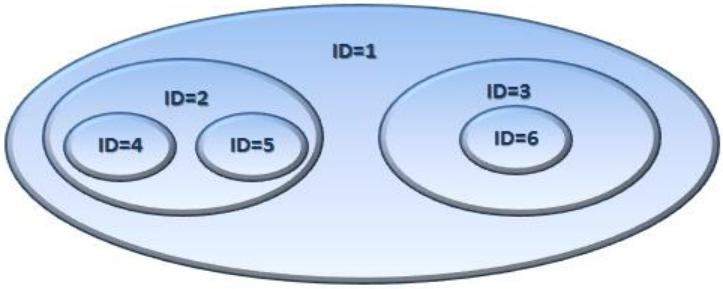


Ilustración 5: Representación de un árbol a través de conjuntos anidados.

En los modelos de conjunto anidados podemos buscar la jerarquía de una nueva forma, como contenedores anidados.

Se muestra en la siguiente figura los valores izquierdo y derecho correspondientes a los nodos anidados.

ID	Orden Izquierdo	Orden Derecho
1	1	12
2	2	7
3	8	11
4	3	4
5	5	6
6	9	10

Ilustración 6: Tabla de valores izquierdo y derecho.

El modelado de una estructura árbol en la base de datos, permite el conocimiento de todos los hijos pertenecientes a un determinado padre y viceversa.

Un árbol cuenta con 4 atributos básicos, id, id_padre, ordenizq y ordender.

Siguiendo el ejemplo anterior se representa a continuación cómo se organiza el árbol:

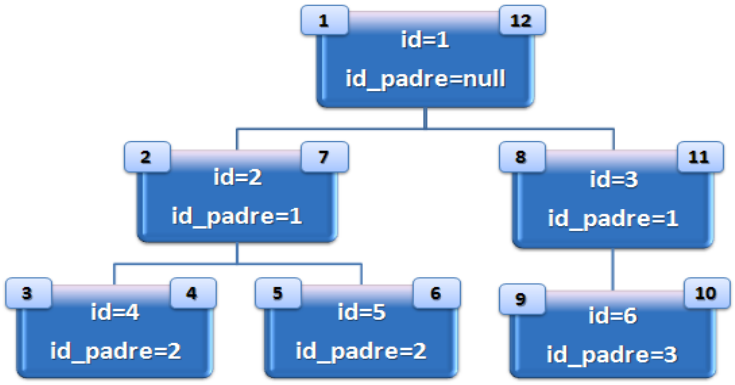


Ilustración 7: Estructura de un árbol.

Los valores izquierdo y derecho de los nodos se determinan mediante una función implementada siguiendo un recorrido en preorden.

Se muestra el correspondiente diagrama Entidad-Relación modelado en el Visual Paradigm.

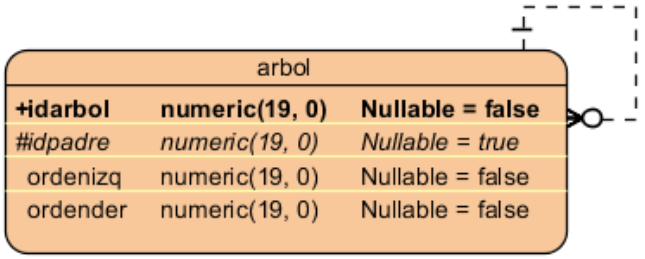


Ilustración 8: Diagrama Entidad-Relación de un árbol.

2.4.2 Multientidad

En diferentes tablas se registran datos comunes de diversos clientes o entidades, pero sucede que en el momento de obtener información en el sistema es necesario, en ocasiones, poder diferenciarla desde el punto de vista de una única empresa. En pos de evitar dificultades que pudieran atentar contra el correcto desenvolvimiento en el desarrollo del sistema por la situación anterior, se diseñó una solución capaz de enfrentar dicha problemática, atendiendo además a que el sistema sea genérico.

El subsistema de Estructura y Composición del proyecto es el encargado de realizar el diseño de las tablas que componen la multientidad. El resto de los subsistemas hace uso de la multientidad a través de un atributo que funciona como llave foránea y cuyo valor se obtiene de otro subsistema, que corresponde a otro esquema en BD. La siguiente figura muestra la manera en que se hace uso de la multientidad a través del diseño:

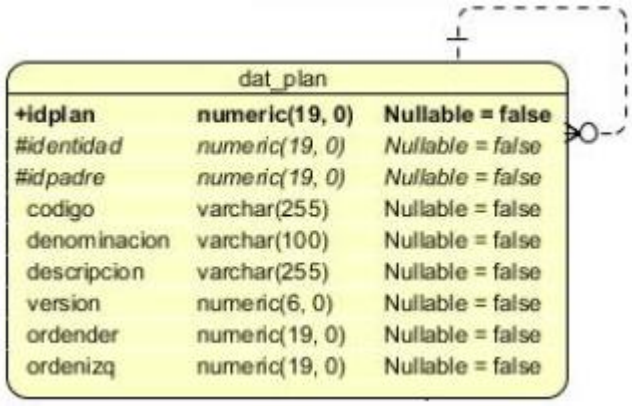


Ilustración 9: Utilización de la multientidad.

Para el uso de la multientidad se agrega a la tabla correspondiente el atributo identidad que hace referencia al idestructura de la tabla dat_estructura del esquema mod_estructuracomp, en este caso varias empresas trabajan con los datos guardados en esta tabla, puesla información es común para ellas. Así se asegura reconocer la entidad a la que corresponde la información.

2.4.3 Control de concurrencia

La concurrencia se evidencia, por ejemplo, cuando dos clientes acceden al unísono a un mismo campo con el objetivo de realizar determinada acción, sobre todo para realizar una modificación.

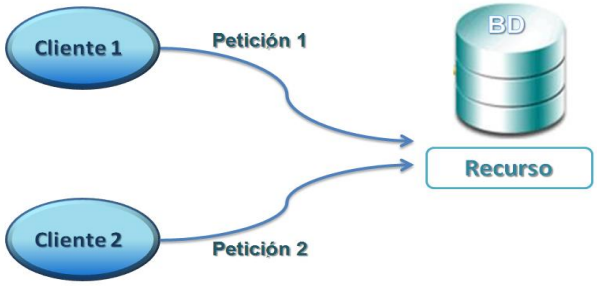


Ilustración 10: Ejemplo de concurrencia.

Es evidente que siempre llega una petición primero que la otra, ejecutándose en el orden de llegada. La primera petición realiza la acción y la BD hace una actualización eminente. Entonces la segunda petición puede estar sometida a dos opciones:

1. Que modifique el recurso al cual está accediendo, notificando una correcta operación e invalidando la acción de la primera petición.
2. Con un control de concurrencia que notifique un error operacional y sea notificado.

Los mecanismos de control de concurrencia deben resolver tres problemas: (11)

- Problema de actualización perdida.
- Problema de la dependencia no confirmada.
- Problema del análisis inconsistente.

En la BD como solución a la concurrencia se creará un campo denominado versión de tipo numérico, el cual es el responsable de almacenar la versión actual, con el objetivo de que cuando varios usuarios carguen un registro al unísono para modificar los datos, se le permitirá actualizar al primero que ejecute la acción, incrementando así la versión correspondiente y cuando los demás usuarios intenten ejecutar su acción se les enviará un mensaje indicando que el registro al cual ha accedido ha sido modificado, por lo que si desea realizar algún cambio debe actualizar.

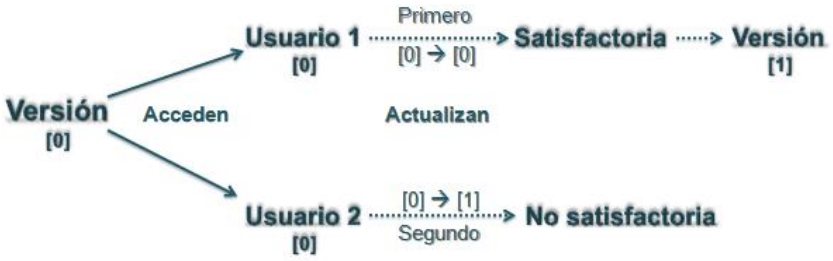


Ilustración 11: Solución para la concurrencia.

En este caso los usuarios 1 y 2 acceden al mismo tiempo al registro cargando la versión con valor 0, cuando el usuario 1 actualiza el registro se realiza la comprobación del campo versión evaluando su actual valor con el que tiene el usuario 1, siendo el mismo se actualiza el registro y cambia el valor del campo versión a 1, así cuando el usuario 2 actualiza realizándose el mismo proceso y no encontrar correspondencia entre los valores en cuestión se le envía una notificación al usuario indicándole que se produjo un cambio por otro usuario y que debe volver a cargar el registro si desea realizar la actualización. En caso de que sean más usuarios se procederá de la misma forma.

Lo anterior será realizado mediante un trigger que llamará a una función, *"chequear"* la cual comprobará que si la nueva versión que pone el cliente es igual a la antigua versión incrementará en uno y permitirá modificar la tupla accedida, (trigger ejecutado antes de realizar la modificación), de lo contrario se lanzaría una excepción.

2.5 Selección y argumentación de los requisitos del sistema propuesto

La captura de requisitos cumple un papel importante en la realización de lo que se desea producir, pues describe el comportamiento del sistema a desarrollar. Los requisitos especifican las condiciones o capacidades que el sistema debe cumplir y las restricciones sobre las cuales debe operar siguiendo la premisa de lograr una comunicación efectiva entre los usuarios y el equipo de trabajo del proyecto, elemento vital para alcanzar la calidad necesaria.

2.5.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, o sea, establecen el comportamiento del sistema.

Gestionar indicadores: el sistema debe permitir gestionar los datos referentes a los indicadores.

- Adicionar indicador: el sistema debe permitir crear nuevos indicadores.
- Modificar indicador: el sistema debe permitir modificar la información de los indicadores.
- Eliminar indicador: el sistema debe permitir eliminar un indicador.

Gestionar columnas: el sistema debe permitir gestionar los datos referentes a las columnas.

- Adicionar columna: el sistema debe permitir crear nuevas columnas.
- Modificar columna: el sistema debe permitir modificar la información de las columnas.
- Eliminar columna: el sistema debe permitir eliminar una columna.

Gestionar tipo de modelo: el sistema debe permitir gestionar los datos referentes a los tipos de modelos.

- Adicionar tipo de modelo: el sistema debe permitir crear nuevos tipos de modelos.
- Modificar tipo de modelo: el sistema debe permitir modificar la información de los tipos de modelos.

- Eliminar tipo de modelo: el sistema debe permitir eliminar un tipo de modelo.

Gestionar plan: el sistema debe permitir gestionar los datos referentes a los planes.

- Adicionar plan: el sistema debe permitir crear nuevos planes.
- Modificar plan: el sistema debe permitir modificar la información de los planes.
- Eliminar plan: el sistema debe permitir eliminar un plan.

Gestionar modelo: el sistema debe permitir gestionar los datos referentes a los modelos.

- Adicionar modelo: el sistema debe permitir crear nuevos modelos.
- Modificar modelo: el sistema debe permitir modificar la información de los modelos.
- Eliminar modelo: el sistema debe permitir eliminar un modelo.

Gestionar comentario: el sistema debe permitir gestionar los datos referentes a los comentarios.

- Adicionar comentario: el sistema debe permitir crear nuevos comentarios.
- Modificar comentario: el sistema debe permitir modificar la información de los comentarios.
- Eliminar comentario: el sistema debe permitir eliminar un comentario.

Gestionar tipo de comentario: el sistema debe permitir gestionar los datos referentes a los tipos de comentarios.

- Adicionar tipo de comentario: el sistema debe permitir crear nuevos tipos de comentarios.
- Modificar tipo de comentario: el sistema debe permitir modificar la información de los tipos de comentarios.
- Eliminar tipo de comentario: el sistema debe permitir eliminar un tipo de comentario.

Gestionar actividad: el sistema debe permitir gestionar los datos referentes a las actividades.

- Adicionar actividad: el sistema debe permitir crear nuevas actividades.
- Modificar actividad: el sistema debe permitir modificar los datos de las actividades.
- Eliminar actividad: el sistema debe permitir eliminar una actividad.

Gestionar normas: el sistema debe permitir gestionar los datos referentes a las normas.

- Adicionar normas a una actividad: el sistema debe permitir adicionar normas a una actividad.
- Modificar normas de una actividad: el sistema debe permitir modificar los datos de las normas de una actividad.
- Eliminar normas de una actividad: el sistema debe permitir eliminar normas de una actividad determinada.

Gestionar tipo de nomenclador: el sistema debe permitir gestionar los datos referentes a los tipos de nomencladores.

- Adicionar tipo de nomenclador: el sistema debe permitir crear nuevos tipos de nomencladores.
- Modificar tipo de nomenclador: el sistema debe permitir modificar la información de los tipos de nomencladores.
- Eliminar tipo de nomenclador: el sistema debe permitir eliminar un tipo de nomenclador.

Gestionar nivel de actividad: el sistema debe permitir gestionar los datos de los niveles de actividad.

- Adicionar nivel de actividad: el sistema debe permitir adicionar niveles de actividad.
- Modificar nivel de actividad: el sistema debe permitir modificar la información de los niveles de actividad.

- Eliminar nivel de actividad: el sistema debe permitir eliminar niveles de actividad.

Gestionar reglas: el sistema debe permitir gestionar las asociaciones de las reglas con las normas.

- Adicionar regla a una norma: el sistema debe permitir adicionar una regla que responda una norma.
- Modificar regla de una norma: el sistema debe permitir modificar la información de una regla de determinada norma.
- Eliminar regla de una norma: el sistema debe permitir eliminar una regla de determinada norma.

Gestionar elementos: el sistema debe permitir gestionar los elementos que corresponden a los tipos de nomencladores.

- Adicionar elemento a un tipo de nomenclador: el sistema debe permitir adicionar los elementos de un tipo de nomenclador.
- Modificar elemento a un tipo de nomenclador: el sistema debe permitir modificar los datos de un elemento de un tipo de nomenclador determinado.
- Eliminar elemento a un tipo de nomenclador: el sistema debe permitir eliminar un elemento correspondiente a un tipo de nomenclador.

Asociar actividades a una entidad: el sistema debe permitir asociar actividades a entidades.

Eliminar asociación de actividades con entidades: el sistema debe permitir eliminar asociaciones de actividades con entidades.

Adicionar columna a un tipo de modelo: el sistema debe permitir asociar columnas a un tipo de modelo.

Eliminar columna de un tipo de modelo: el sistema debe permitir eliminar asociaciones de las columnas con los tipos de modelos.

Adicionar indicador a un modelo: el sistema debe permitir adicionar un indicador a un modelo según la entidad correspondiente especificando la información deseada.

Eliminar indicador de un modelo: el sistema debe permitir eliminar indicadores de los modelos.

Adicionar un modelo de consolidación a un plan: el sistema debe permitir adicionar un modelo de consolidación a un plan.

Modificar un modelo de consolidación: el sistema debe permitir modificar los datos de los modelos de consolidación.

Eliminar un modelo de consolidación: el sistema debe permitir eliminar un modelo de consolidación.

2.5.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe cumplir, éstos ni describen información a guardar, ni funciones a realizar, a través de los mismos se debe obtener un producto atractivo, usable, rápido y confiable.

Se describen a continuación los requerimientos fundamentales relacionados con la BD definidos por el equipo de trabajo de la línea (módulo o subsistema).

Confiabilidad

- El SGBD escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.

Rendimiento

- Al intercambiar datos con el sistema, ya sea mediante acciones de inserción, búsqueda, eliminación o modificación de datos, en un servidor de 1 GB de memoria RAM, se debe recibir la respuesta de la acción realizada en un período de 0.1 a 0.8 segundos.

Seguridad

- Si se desea eliminar o modificar un elemento de la base de datos que está siendo utilizado por otro elemento que depende de él, el sistema no permite que este elemento sea eliminado, salvo las excepciones conciliadas previamente.
- Si dos usuarios acceden a un recurso e intentan modificar la información simultáneamente, el sistema solo debe dejar que modifique uno de ellos y notifica al otro usuario que actualice para obtener la última versión de los datos.
- Debe garantizarse la recuperación de los datos, en caso de algún fallo, a través de copias de respaldo realizadas, con frecuencia preferentemente diaria.
- La información almacenada en la BD deberá estar protegida de acceso no autorizado evitando modificaciones no deseadas en los datos.

Software

- El Sistema Gestor de Base de Datos a utilizar debe ser: PostgreSQL 8.3.8.

Hardware

Servidor de Base de datos:

- Tarjeta de Red: 1
- Procesador: 3.00 GHZ
- RAM: 1GB
- Disco duro: 160 GB
- UPS: 1

2.6 Diseño de la BD

2.6.1 Patrones utilizados

Un patrón describe un problema que ocurre varias veces, además de plantear la solución a este problema pudiendo utilizarse tantas veces sea necesario.

Los patrones de diseño son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Es una descripción de un problema y la solución a la que le da el nombre y que se puede aplicaren nuevos contextos. Muchos patrones ayudan a asignar responsabilidades a los objetos.(24)

Los patrones de diseño son soluciones simples a problemas específicos y comunes del diseño orientado a objetos, su principal objetivo es agrupar una colección de soluciones de diseño que sean válidas en distintos contextos. Es una solución a un problema de diseño no trivial que es efectiva, además facilitan el aprendizaje al programador inexperto, pudiendo establecer parejas problema-solución.(25)

Se utilizaron varios patrones de diseño dentro de los que se encuentran los patrones descritos por Kyle Brown y Bruce G. Whitenack, a continuación se describe la utilización de los mismos.

Representación de objetos como tablas

Problema: ¿Cómo representar un objeto en un esquema de base de datos relacionales?

Solución**:** Definir una tabla para cada clase de objetos persistentes. Los atributos de la clase que son tipos primitivos serán las columnas de las tablas.

Representación de relaciones como tablas

Problema: ¿Cómo representar una relación en un esquema de base de datos relacionales?

Solución:

- Para las relaciones de uno a uno ó uno a muchos:

  Colocar una clave ajena en la tabla de cardinalidad uno, para representar la relación de los objetos.

  O crear una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.

- Para las relaciones muchos a muchos:

  Crear una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.

Identificador de objetos

Problema: ¿Cómo mantener la identidad de un objeto en una base de datos relacional? Cada identidad de objetos individuales debe ser presentada en la base de datos.

Solución: Asignar un identificador independiente (OID) a cada objeto persistente. Se recomienda el uso de un generador de secuencias si hay alguno disponible en la base de datos ya que los identificadores son generalmente enteros largos que garantizan que sean únicos para una clase de objetos en particular.

Referencia de llaves foráneas

Problema: ¿Cómo representar objetos que referencian otros objetos que no son de tipos de datos base?

Solución: Asignar a cada objetos un identificador único. Luego añadir una columna por cada variable de instancia que no tenga un tipo de dato base o sea una colección. En esa columna almacenar el identificador del objeto referenciado y declarar la columna como llave foránea.

Representar una herencia en una base de datos relacional

Problema: ¿Cómo representar una jerarquía de herencia en una base de datos relacional?

Solución: Crear una tabla para cada clase en la herencia que tenga atributos. Cada atributo de la clase será una columna de la tabla, añadir además una columna adicional que represente la llave común entre todas las clases de la herencia.

Otro patrón utilizado es el de claves subrogadas. Estas claves no son más que un identificador único que se asigna a cada registro de una tabla. Son de tipo numérico secuencial sin significado especial y debe ser el único campo que sea clave principal de cada tabla.

2.6.2 Modelo físico

Luego de realizado el diseño lógico de la BD se obtuvo el diseño físico que se muestra:

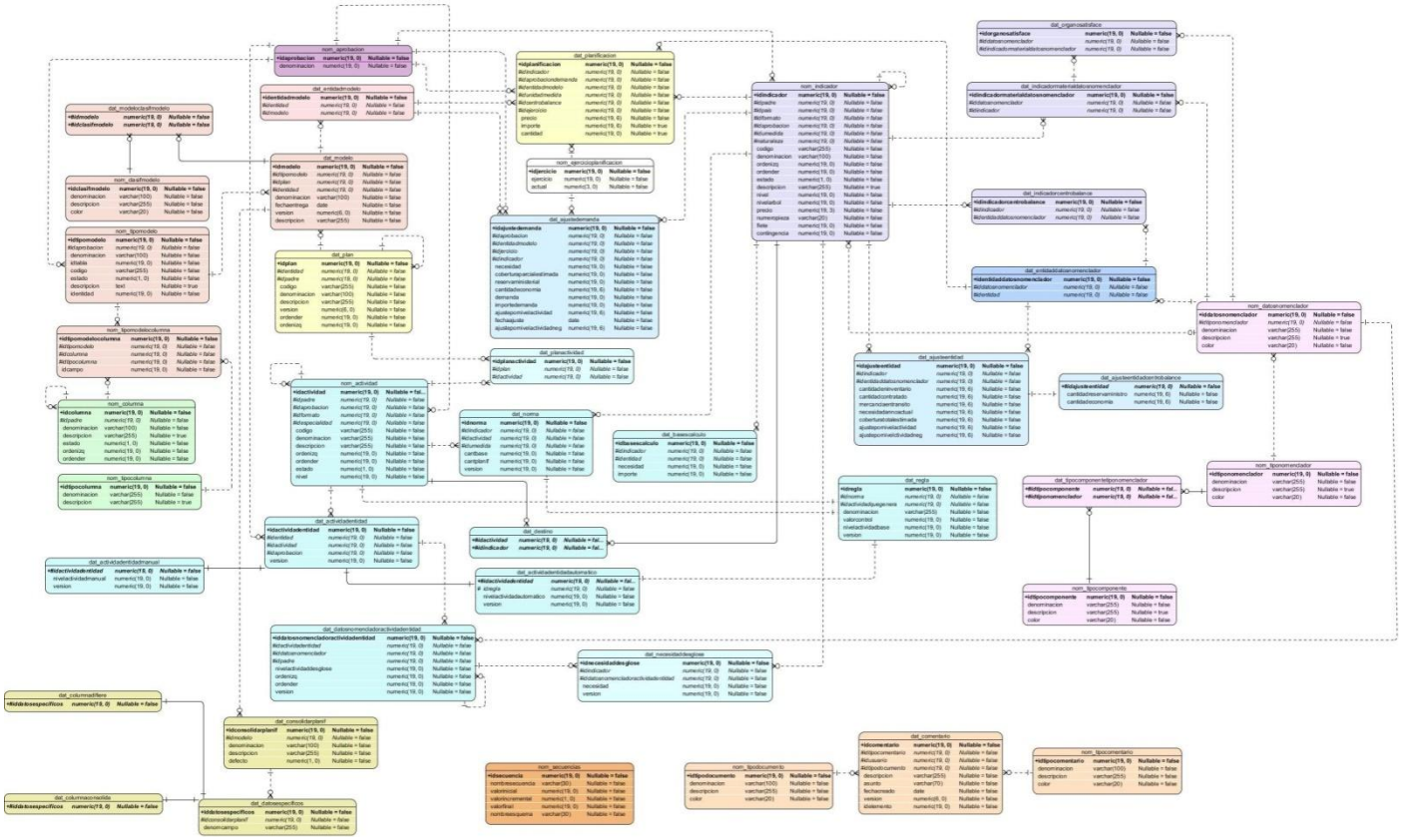


Ilustración 12: Diseño físico.

Descripción de las tablas

Se presenta a continuación la descripción de algunas de las tablas del diseño físico modeladas anteriormente, la descripción del resto de las tablas se encuentran en los anexos, para más detalles puede consultar el Diccionario de datos.

Nombre: nom_columna		
Descripción: Tabla que guarda los atributos, los cuales serán las columnas de los tipos de modelos.		
Atributos	**Tipos**	**Descripción**
idcolumna	numeric(19)	Atributo identificador de la columna.

denominacion	varchar(100)	Atributo que permite establecerle un nombre a la columna.
descripcion	varchar(255)	Atributo de la tabla con el cual se permite explicar en detalles cada columna.
estado	numeric(1)	Atributo con el cual se controla el estado de cada columna (1=activa y 0=inactiva).
idpadre	numeric(19)	Llave foránea que indica el id del padre en la estructura arbórea.
ordenizq	numeric(19)	Atributo que corresponde al valor izquierdo del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.
ordender	numeric(19)	Atributo que corresponde al valor derecho del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.

Tabla 4: Descripción de la tabla nom_columna.

Nombre: nom_indicador		
Descripción: Tabla nomencladora que guarda los indicadores, los cuales serán filas de los modelos.		
Atributos	**Tipos**	**Descripción**
idindicador	numeric(19)	Atributo que identifica a los indicadores.
denominacion	varchar(100)	Atributo que registra el nombre que se le asigna al indicador.
codigo	varchar(255)	Atributo que representa el código único del indicador.
estado	numeric(1)	Atributo que indica si el indicador se encuentra activado, en caso que tome valor 1 se encuentra activado, si toma valor 0 está desactivado, no permitiendo seleccionarlo.
descripcion	varchar(255)	Atributo de la tabla con el cual se permite explicar en detalles cada indicador.

nivel	numeric(2)	Campo que especifica si se va a planificar o no a partir de ese indicador en esa rama del árbol de indicadores, tomaría valor 0 en caso negativo, 1 en caso positivo.
nivelarbol	numeric(19)	Indica qué nivel del indicador corresponde: genérico, subgenerico, surtido o específico, a través de los valores de 0, 1, 2 y 3 respectivamente.
precio	numeric(19,3)	Atributo que guarda el precio del indicador.
numeropieza	varchar(20)	Atributo que almacena un número como identificador o número de serie de las piezas.
flete	numeric(19)	Atributo que almacena el valor del flete del indicador, siendo el valor financiero destinado, por ejemplo, a los servicios de transporte.
contingencia	numeric(19)	Atributo que almacena el valor de la contingencia del indicador, siendo el valor financiero destinado a situaciones de riesgo en caso de ocurrencia de un incidente.
naturaleza	numeric(19)	Almacena la naturaleza del indicador que sería un identificador foráneo proveniente de la tabla nom_datosnomenclador, el cual indicaría si es: equipo, medios materiales, servicios u otros gastos.
idpadre	numeric(19)	Llave foránea que indica el id del padre en la estructura arbórea.
idaprobacion	numeric(19)	Identificador del estado de aprobación en que se encuentre el indicador.
idumedida	numeric(19)	Llave foránea proveniente del esquema de datos maestros con el cual se identificará la unidad de medida correspondiente al indicador.
idpais	numeric(19)	Identificador del país del cual proviene el indicador.
idformato	numeric(19)	Atributo que establece el formato de un indicador.

ordenizq	numeric(19)	Atributo que corresponde al valor izquierdo del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.
ordender	numeric(19)	Atributo que corresponde al valor derecho del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.

Tabla 5: Descripción de la tabla nom_indicador.

Nombre: dat_plan		
Descripción: Tabla que persiste los planes y sus características.		
Atributos	**Tipos**	**Descripción**
idplan	numeric(19)	Atributo identificador del plan.
denominacion	varchar(100)	Atributo que persiste el nombre del plan.
codigo	varchar(255)	Atributo que permite establecerle un código único al plan, con el cual podrá ser identificado.
descripcion	varchar(255)	Atributo de la tabla con el cual se permite explicar en detalles cada plan.
version	numeric(6)	Atributo de la tabla que permite tener control de la concurrencia en la misma.
identidad	numeric(20)	Llave foránea que indica la entidad a la que corresponde el plan.
idpadre	numeric(19)	Llave foránea que indica el id del padre en la estructura arbórea.
ordenizq	numeric(19)	Atributo que corresponde al valor izquierdo del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.

ordender	numeric(19)	Atributo que corresponde al valor derecho del nodo, permitiendo ubicar el nodo en el árbol y agilizando de esta forma las búsquedas.

Tabla 6: Descripción de la tabla dat_plan.

2.7 Uso de índices

Los índices son utilizados para mejorar el rendimiento de la BD, incrementando la velocidad de las operaciones ya que permite al servidor de BD encontrar y recuperar tuplas de una tabla en menor tiempo que si no existiera. Se suelen usar sobre aquellos atributos que son consultados con frecuencia.

Importante tener claro que abusar del uso de índices más allá de mejorar el rendimiento podría atentar en su contra, los índices tienen un precio, pues siempre que se realice in INSERT sobre una tabla, PostgreSQL tiene que actualizar cada índice en la tabla para reflejar los cambios.

Se agilizará el acceso a los datos cuando en las búsquedas la consulta tenga condiciones asociadas a índices y cuando se exige ordenar la información pues el sistema comprueba si existe algún índice que le permite devolver la información ordenada.

En la BD propuesta se utilizaron varios índices con el objetivo de incrementar el rendimiento, además de los asociados a llaves primarias y campos únicos, se definieron índices para los campos ordenizq, ordender e idpadre de las tablas arbóreas nom_indicador y nom_actividad, además de índices para especificar que la combinación de dos atributos en una tabla debe ser único, como es el caso de idactividad e identidad de la tabla dat_actividadentidad.

2.8 Conclusiones parciales

El diseño de la BD que se obtuvo responde a las necesidades planteadas, con el mismo se logra reducir la redundancia evitando la existencia de datos repetidos innecesariamente aumentando la integridad de los datos, evitando así que se creen anomalías en las consultas realizadas para extraer e insertar datos en el sistema, además la utilización de índices logra mejorar considerablemente las operaciones con los datos.

Capítulo 3: Validación del diseño

3.1 Introducción

En el presente capítulo se realiza la validación teórica y funcional del diseño de la base de datos teniendo en cuenta aspectos de particular importancia entre los que destacan: la normalización, redundancia, integridad y seguridad de la BD. Se realizan además pruebas de concepto o de carga intensiva centradas en comprobar el correcto funcionamiento de la BD.

3.2 Validación teórica

El diseño de BD es un punto crucial en el desarrollo de un sistema, por lo cual se deben tener presente una serie de puntos elementales que garanticen un buen diseño. Estos aspectos son la normalización del diseño, la integridad de los datos, el análisis de la redundancia de la información y la seguridad de los datos, garantizado de esta manera el acceso a los datos por el personal autorizado y que la información no se altere como consecuencia de acciones no controladas.

3.2.1 Integridad de los datos

La integridad en una BD se refiere a la corrección y exactitud de la información contenida. Cualquier BD puede estar sujeta a una serie de restricciones de integridad de diversa complejidad. La integridad de los datos puede verse afectada de diferentes maneras al realizar modificaciones al contenido con sentencias INSERT, UPDATE o DELETE, a través de adiciones de datos inválidos y de modificaciones de datos existentes haciendo que tomen valores incorrectos o eliminándolos.

La integridad de los datos se contempla en diferentes niveles. Las relaciones de integridad referencial aseguran que se mantienen las asociaciones necesarias entre las relaciones. Las restricciones de integridad de la base de datos gobiernan la misma como un todo y las restricciones de integridad de transacciones controlan la forma en que se manipulan los datos, dentro de una o entre múltiples bases de datos.(26)

La integridad de los datos agrupaen varias categorías como:

Integridad de entidad

Relacionada con las llaves primarias de las tablas, estableciendo que ningún atributo que forme parte de la llave primaria puede aceptar valores nulos. Por definición, una llave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la llave primaria sirve para identificar las tuplas de modo único. En la BD propuesta se define una llave primaria para cada tabla, la cual no puede ser nula.

Integridad de clave

Establece que el atributo clave o llave primaria no puede tomar valores iguales en tuplas diferentes. En la BD propuesta se garantiza que las llaves primarias no pueden repetirse, las mismas son de tipo numeric (19,0), generándose incrementalmente.

Integridad de dominio

Una restricción de integridad de dominio es una regla que define los valores válidos de cada uno de los atributos de las tablas en la BD. En ocasiones es necesario definir más de una restricción de dominio para describir por completo un dominio.

Existe integridad de dominio básica, como no poder introducir letras en campos donde se va a almacenar números. Estas normas o reglas de integridad de dominio pueden indicar qué campos son necesarios tener obligatoriamente con valores, para que la base de datos no tenga datos sin conectar en el caso de tener relaciones o dependencias entre tablas.(27)

Se hace uso de los tipos de datos predefinidos por el gestor utilizado, específicamente: numeric, varchar, date, tratando de no poner campos nulos para evitar resultados inconsistentes, a pesar que en algunos casos es necesario.

Integridad referencial

Garantiza que las relaciones que se establezcan entre las tablas de la BD sean coherentes, manteniendo la consistencia entre las tuplas de las dos relaciones, aplicándose específicamente a las llaves foráneas. Si se tiene en una tabla una columna declarada como llave foránea, sus valores deben coincidir con los

valores presentes en la llave primaria de la tabla a la cual se hace referencia, de esta forma no se puede agregar un registro en la tabla que contiene la llave foránea si no existe uno que lo respalde, o sea, que exista en la tabla vinculada. Permite además realizar actualizaciones y eliminaciones en cascada, asegurando que las modificaciones que se realicen en una tabla se reflejen en la otra.

Un ejemplo de aplicación en el modelo de BD de este tipo de restricción se manifiesta en la tabla dat_modelo, la cual tiene los atributos idplan e idtipomodelo, que provienen y son llaves primarias de las tablas dat_plan y nom_tipomodelo respectivamente y cumplen con las siguientes restricciones:

- Estos atributos tienen los mismos dominios que en sus tablas origen.
- Al adicionar una tupla en la tabla dat_modelo, los valores correspondientes a idplan e idtipomodelo deben estar presentes en sus tablas origen.

El SGBD utilizado, asegura que las referencias de llaves foráneas sean tratadas correctamente, de esta manera se comprueba que en el modelo de datos se cumple con la regla de integridad referencial.

3.2.2 Normalización

El proceso de normalización se efectúa en el momento de realizar el modelo lógico de BD, con el objetivo de evitar anomalías de actualización en las estructuras de datos, asegurando así que las relaciones obtenidas no tengan datos redundantes.

La normalización es un proceso mediante el cual se dividen relaciones no convenientes, distribuyendo los atributos en relaciones más pequeñas que cumplan con una serie de requisitos o características deseables, siguiendo la premisa de eliminar redundancias y anomalías de inserción, eliminación y actualización.

El proceso de normalización es un estándar que consiste, básicamente, en un proceso de conversión de las relaciones entre entidades, evitando (28):

- La redundancia de los datos: repetición de los datos en un sistema.
- Anomalías de actualización: inconsistencias de los datos como resultado de datos redundantes y actualizaciones parciales.

- Anomalías de borrado: pérdida no intencionada de datos debido a que se han borrado otros datos.
- Anomalías de inserción: imposibilidad de adicionar datos en la BD debido a la ausencia de otros datos.

Existen 9 formas normales, de las cuales las tres primeras son las más usadas:

Primera forma normal (1FN)

Se dice que una tabla esta en primera forma normal si y sólo si cada uno de los campos contienen un único valor para un registro determinado.(28)

En esta forma normal están prohibidos los atributos multivaluados, compuestos y sus combinaciones, o sea que los atributos deben ser atómicos, que no puedan dividirse. En otras palabras, en esta fase no se permiten relaciones dentro de relaciones.

La BD propuesta cumple con la primera forma normal puesto que los atributos de cada una de las tablas son atómicos, no siendo ni multivaluados ni compuestos.

Segunda forma normal (2FN)

La primera condición que debe cumplir es que se encuentre en 1FN, luego que los atributos que no forman parte de la llave primaria, deben depender de manera total de los atributos que forman la llave primaria, lo cual establece que todas las dependencias parciales deben ser eliminadas. Esta forma normal se aplica a tablas que tienen una llave compuesta.

La segunda forma normal compara todos y cada unos de los campos de la tabla con la clave definida. Si todos los campos dependen directamente de la clave se dice que la tabla esta es segunda forma normal (2FN). (28)

La BD propuesta cumple con la segunda forma normal ya que además de estar en primera forma normal, los atributos de las tablas con llaves compuestas no presentan dependencias funcionales parciales.

Tercera forma normal (3FN)

Debe empezar cumpliendo que se encuentre en 2FN, luego que cada atributo de la relación que no está contenido dentro de la llave primaria dependa solo de la llave primaria y no de ningún otro atributo, o sea tiene como objetivo eliminar las dependencias transitivas. Entiéndase por dependencia transitiva aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Se dice que una tabla está en tercera forma normal si y solo si los campos de la tabla dependen únicamente de la clave, dicho en otras palabras los campos de las tablas no dependen unos de otros. (28)

La BD propuesta cumple con la tercera forma normal, pues todos los atributos de las tablas dependen de manera directa de la llave primaria.

3.2.3 Análisis de la redundancia de la información

La redundancia no es más que la repetición innecesaria de información en la BD. Esto puede significar un inconveniente al realizar modificaciones en los datos, siendo el elemento más frecuente de inconsistencias. Necesita además, mayor espacio de almacenamiento lo cual puede aumentar los costes de almacenamiento y acceso a los datos.

Un modelo de datos normalizado reduce en gran medida la redundancia. La presencia de redundancia implicaría la necesidad de utilizar triggers para asegurar que se mantenga la consistencia cuando los datos son insertados, modificados o eliminados.

En ocasiones resulta más costoso obtener la información a partir de los datos relacionados que el costo de su mantenimiento, teniendo en cuenta la periodicidad con que los datos son consultados y las operaciones a realizar, así como el tiempo en que se quiera obtener la respuesta. Siendo así, introducir redundancia puede proporcionar grandes beneficios. A continuación algunos ejemplos en los que la redundancia puede ser favorable:

Evitando uniones (Joins) para búsquedas.

Evitando uniones (Joins) para campos calculables.

La introducción de la redundancia por más mínima que sea, debe ser bien analizada y estudiada, donde los beneficios que se obtengan sean evidentes.

En la BD propuesta existe cierto nivel de redundancia ya que la tabla dat_planificacion, tiene los atributos precio e idunidadmedida que corresponden al indicador en el momento en que se realiza la planificación, valores que en un primer momento se encuentran en la tabla nom_indicador. Se decide persistir estos valores en dat_planificacion pues si en algún momento cambian los datos de estos atributos en nom_indicador existe necesidad de conocer el precio y la unidad de medida del indicador en el momento en que se planifica.

3.2.4 Análisis de la seguridad de la BD

La seguridad de la BD se basa en las acciones a tomar para garantizar que la información almacenada sea consultada por los usuarios autorizados. Los datos que guarda una BD tienen el riesgo de sufrir ataques que puedan provocar la pérdida o modificación no deseada de la información, por lo cual es sumamente importante establecer los mecanismos necesarios para asegurar la integridad de los mismos, dejando claro los privilegios de acceso de los usuarios. Es importante además garantizar la recuperación de la información en caso de que ocurra alguna anomalía en la cual se produzca algún fallo con la BD que provoque la pérdida de información.

Se define para garantizar la seguridad de la BD propuesta un usuario genérico para el trabajo de los arquitectos fuera de las líneas o subsistemas. El usuario tendrá permisos sobre toda la BD y sus objetos. Se tienen usuarios específicos por líneas, para uso de los responsables de la BD en la línea, este usuario es responsable del esquema y los objetos específicos de la línea. Se establece además un usuario genérico para uso de los desarrolladores en todas las líneas que tendrá permisos para insertar, actualizar y eliminar datos de las tablas así como ejecutar funciones y obtener los valores de las secuencias.

Se realizan copias de respaldo diariamente a la BD, en la mañana antes de empezar el trabajo y por la tarde luego de terminado el mismo, estas copias de respaldo se realizan de forma automática.

3.3 Validación Funcional

Para comprobar el rendimiento de la BD y su correcto funcionamiento es necesario realizar determinadas pruebas, asegurando así que la misma cumpla con los requisitos definidos. Las pruebas persiguen el objetivo de simular una carga de producción real y observar cómo se comporta la BD ante esta carga, comprobando que el sistema satisface las necesidades además de que se realice sin violar la integridad de los datos.

3.3.1 Pruebas de rendimiento

Las pruebas de rendimiento son parte fundamental en el proceso de desarrollo de una aplicación si se quiere conocer cómo responde el sistema ante determinadas situaciones para determinar fallas que puedan atentar contra la calidad.

Estas pruebas son factibles, ya sea para determinar qué parte del sistema provoca que el conjunto rinda mal como para comparar dos sistemas y determinar cuál de ellos funciona mejor, además de demostrar que el sistema cumple con los criterios de rendimiento.

Es importante que las condiciones en las que se desarrollan las pruebas sean similares a las esperadas en la realidad tratando de garantizar que los resultados sean los más precisos posibles, sin embargo, es muy difícil de conseguir.

Prueba de Volumen

Esta prueba se centra, a través del cúmulo de información en la BD, en analizar el comportamiento de esta verificando si la misma alcanza su límite de almacenamiento y pueda causar fallas. Se utilizó la herramienta Data Generator 2005 para PostgreSQL para el llenado voluminoso de la BD.

Las tablas fueron llenadas con un rango de datos equivalentes al que tendrá la BD en un período no menor a un año. Las tablas que mayor cantidad de tuplas acumularon fueron nom_indicador, nom_actividad, dat_planificacion y dat_basescalculo con 600000, 400000, 300000 y 100000 registros respectivamente, generando los datos de esta última tabla en un tiempo de 8 minutos y 22 segundos como se muestra en la figura.

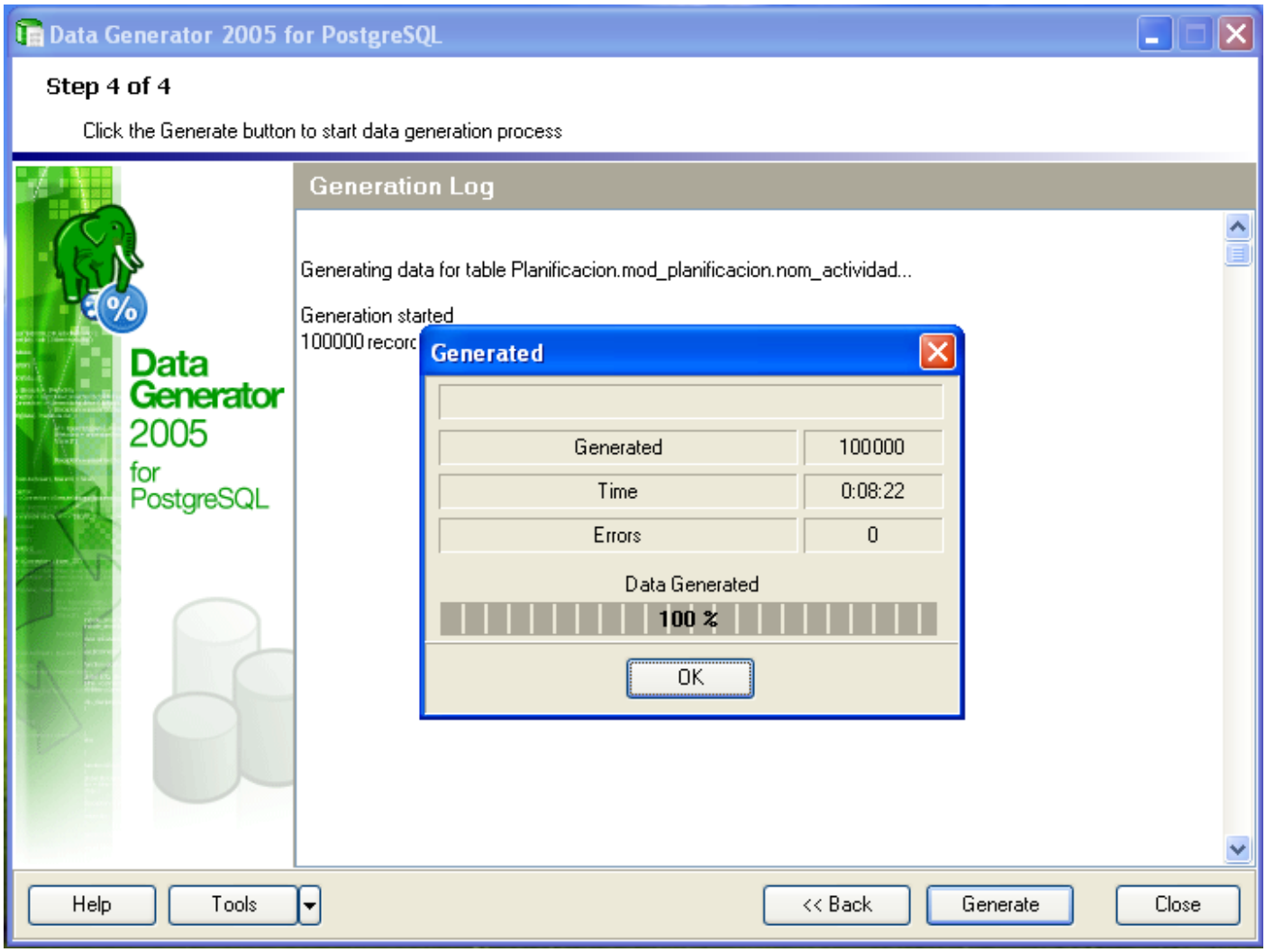


Ilustración 13: Resultados de la generación de datos en nom_actividad.

Al introducir los datos en las tablas no se presentaron problemas de límites de capacidad, desbordamiento de columnas, atributos o tipos de datos. La utilización de esta herramienta permitió verificar la integridad de los datos, además de garantizar que el diseño de las estructuras de la BD y el gestor utilizado para el desarrollo soportan el cúmulo de información requerida para el funcionamiento de la BD.

Prueba de Carga

Con estas pruebas se puede obtener como resultado el tiempo de ejecución de las consultas críticas, las que se realizarán con más frecuencia y las más complejas, o sea, las que más relaciones entre tablas presenten, con el objetivo de comprobar el comportamiento de la BD.

Las consultas fueron ejecutadas anteponiendo los comandos EXPLAIN ANALYZE, con lo que se observa el plan de ejecución, en este caso, la opción ANALYZE permite la evaluación de la consulta, entregando el tiempo empleado en su ejecución.

Consulta 1: listar indicadores

Consulta que comprueba el rendimiento y la velocidad a la hora de recuperar los indicadores.

Tablas relacionadas

nom_indicadores con 600000 registros.

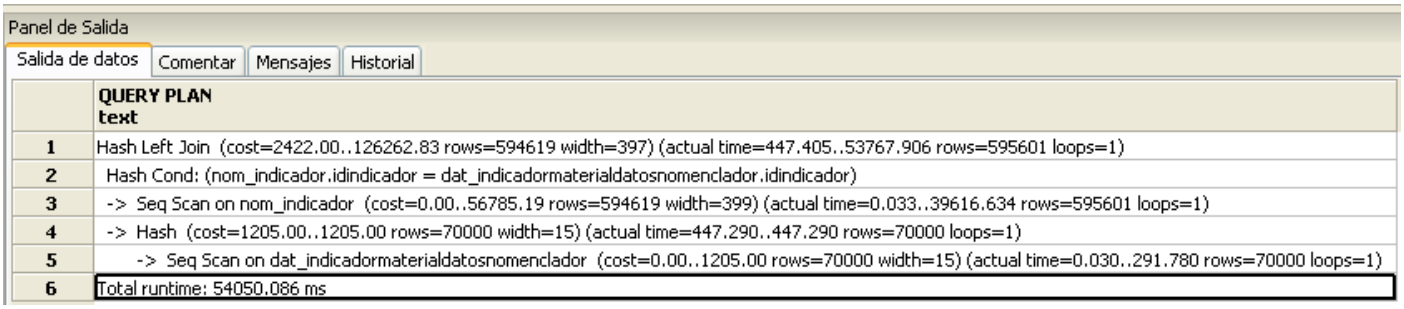
dat_indicadormaterialdatosnomenclador con 80000 registros.

Consulta

```
EXPLAIN ANALYZE SELECT mod_planificacion.nom_indicador.denominacion, mod_planificacion.nom_indicador.codigo,
        mod_planificacion.nom_indicador.descripcion,
        mod_planificacion.dat_indicadormaterialdatosnomenclador.iddatosnomenclador
        from mod_planificacion.nom_indicador
        left join mod_planificacion.dat_indicadormaterialdatosnomenclador
        on (mod_planificacion.nom_indicador.idindicador =
mod_planificacion.dat_indicadormaterialdatosnomenclador.idindicador)
```

Resultado

La siguiente imagen muestra el resultado de una ejecución de la consulta listar indicadores:


Panel de Salida

Salida de datos | Comentar | Mensajes | Historial

	QUERY PLAN text
1	Hash Left Join (cost=2422.00..126262.83 rows=594619 width=397) (actual time=447.405..53767.906 rows=595601 loops=1)
2	Hash Cond: (nom_indicador.idindicador = dat_indicadormaterialdatosnomenclador.idindicador)
3	-> Seq Scan on nom_indicador (cost=0.00..56785.19 rows=594619 width=399) (actual time=0.033..39616.634 rows=595601 loops=1)
4	-> Hash (cost=1205.00..1205.00 rows=70000 width=15) (actual time=447.290..447.290 rows=70000 loops=1)
5	-> Seq Scan on dat_indicadormaterialdatosnomenclador (cost=0.00..1205.00 rows=70000 width=15) (actual time=0.030..291.780 rows=70000 loops=1)
6	Total runtime: 54050.086 ms

Ilustración 14: Resultado de listar indicadores.

Consulta 2: listar indicadores hijos según órdenes

Consulta que comprueba el rendimiento y la velocidad a la hora de recuperar los indicadores hijos según los órdenes izquierdo y derecho.

Tablas Relacionadas

nom_indicador con 600000 registros.

Consulta

```
EXPLAIN ANALYZE  SELECT mod_planificacion.nom_indicador.denominacion from mod_planificacion.nom_indicador
                    where mod_planificacion.nom_indicador.ordenizq> 6399
                        and mod_planificacion.nom_indicador.ordender< 6940
```

Resultado

La siguiente imagen muestra el resultado de una ejecución de la consulta listar indicadores hijos según los órdenes izquierdo y derecho.

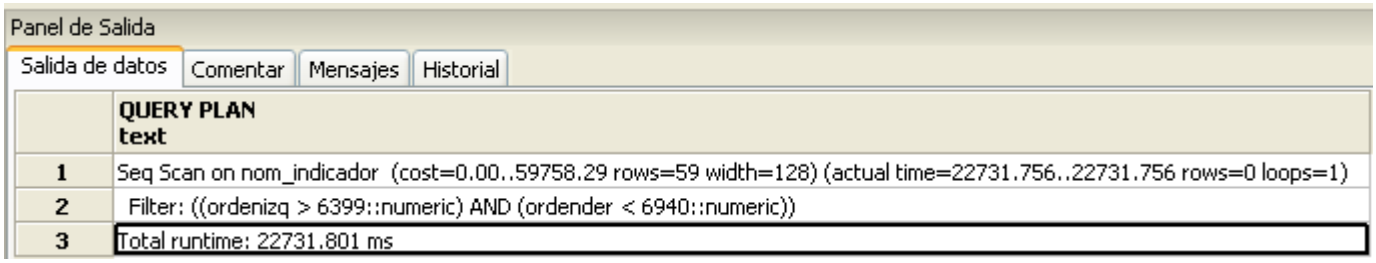


Ilustración 15: Resultado de listar indicadores hijos según los órdenes.

Consulta 3: buscar indicadores de un modelo

Consulta que comprueba el rendimiento y la velocidad a la hora de realizar una búsqueda en las tablas correspondientes, utilizando los INNER JOIN necesarios para ello.

Tablas Relacionadas

dat_planificacion con 80000 registros.

dat_entidadmodelo con 50000 registros.

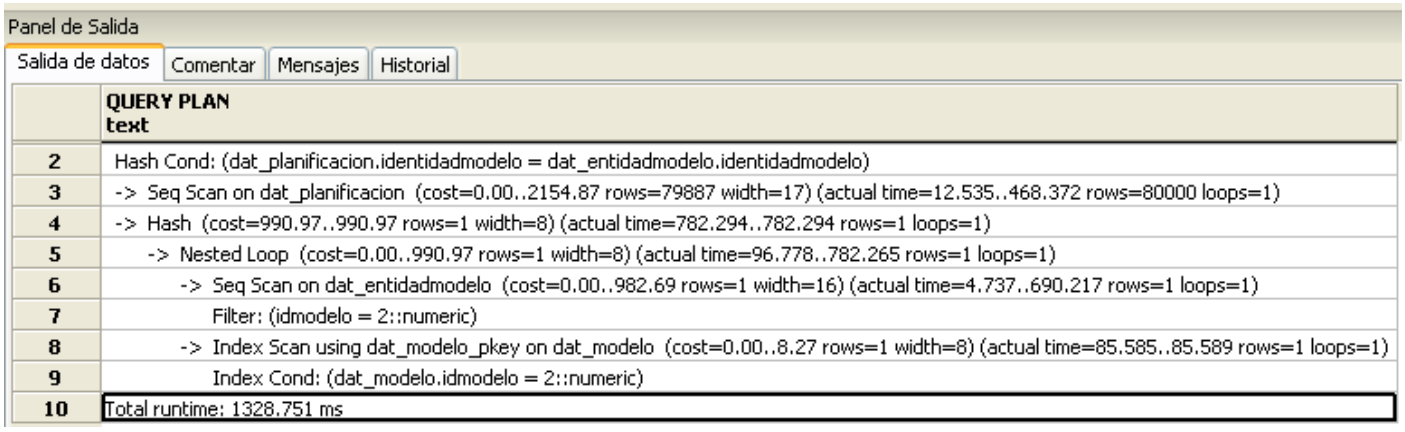
dat_modelo con 50000 registros.

Consulta

```
EXPLAIN ANALYZE select mod_planificacion.dat_planificacion.idindicador
                    from mod_planificacion.dat_planificacion
                    inner join mod_planificacion.dat_entidadmodelo
                    on mod_planificacion.dat_planificacion.identidadmodelo=
    mod_planificacion.dat_entidadmodelo.identidadmodelo
                    inner join mod_planificacion.dat_modelo
                    on mod_planificacion.dat_entidadmodelo.idmodelo= mod_planificacion.dat_modelo.idmodelo
                    where mod_planificacion.dat_modelo.idmodelo = 2
```

Resultado

La siguiente imagen muestra el resultado de una ejecución de la consulta buscar indicadores de un modelo.


Panel de Salida

Salida de datos | Comentar | Mensajes | Historial

	QUERY PLAN text
2	Hash Cond: (dat_planificacion.identidadmodelo = dat_entidadmodelo.identidadmodelo)
3	-> Seq Scan on dat_planificacion (cost=0.00..2154.87 rows=79887 width=17) (actual time=12.535..468.372 rows=80000 loops=1)
4	-> Hash (cost=990.97..990.97 rows=1 width=8) (actual time=782.294..782.294 rows=1 loops=1)
5	-> Nested Loop (cost=0.00..990.97 rows=1 width=8) (actual time=96.778..782.265 rows=1 loops=1)
6	-> Seq Scan on dat_entidadmodelo (cost=0.00..982.69 rows=1 width=16) (actual time=4.737..690.217 rows=1 loops=1)
7	Filter: (idmodelo = 2::numeric)
8	-> Index Scan using dat_modelo_pkey on dat_modelo (cost=0.00..8.27 rows=1 width=8) (actual time=85.585..85.589 rows=1 loops=1)
9	Index Cond: (dat_modelo.idmodelo = 2::numeric)
10	Total runtime: 1328.751 ms

Ilustración 16: Resultado de listar indicadores de un modelo.

Consulta 4: buscar reglas generadas por una actividad

Consulta que comprueba el rendimiento y la velocidad a la hora de realizar una búsqueda en las tablas correspondientes dado el identificador de la actividad que genera la regla, utilizando los INNER JOIN necesarios para ello.

Tablas Relacionadas

dat_regla con 50000 registros.

dat_norma con 50000 registros.

nom_indicador con 600000 registros.

nom_actividad con 100000 registros.

Consulta

EXPLAIN ANALYZE select mod_planificacion.dat_regla., mod_planificacion.nom_actividad.denominacion, mod_planificacion.nom_indicador.denominacion*

```
from mod_planificacion.dat_regla inner join mod_planificacion.dat_norma
        on mod_planificacion.dat_regla.idnorma =mod_planificacion.dat_norma.idnorma
        inner join mod_planificacion.nom_indicador
        on mod_planificacion.dat_norma.idindicador = mod_planificacion.nom_indicador.idindicador
        inner join mod_planificacion.nom_actividad
        on
mod_planificacion.dat_regla.idactividadquegenera=mod_planificacion.nom_actividad.idactividad
        where mod_planificacion.dat_norma.idactividad=1
        order by (mod_planificacion.dat_regla.denominacion)
```

Resultado

La siguiente imagen muestra el resultado de una ejecución de la consulta buscar reglas generadas por una actividad.

Panel de Salida

Salida de datos | Comentar | Mensajes | Historial

	QUERY PLAN text
1	Sort (cost=3451.52..3451.53 rows=1 width=452) (actual time=1965.561..1965.562 rows=1 loops=1)
2	Sort Key: dat_regla.denominacion
3	Sort Method: quicksort Memory: 17kB
4	-> Nested Loop (cost=1333.48..3451.51 rows=1 width=452) (actual time=1436.293..1955.685 rows=1 loops=1)
5	-> Nested Loop (cost=1333.48..3443.11 rows=1 width=333) (actual time=1330.445..1849.828 rows=1 loops=1)
6	-> Hash Join (cost=1333.48..3436.99 rows=1 width=203) (actual time=1321.771..1841.147 rows=1 loops=1)
7	Hash Cond: (dat_regla.idnorma = dat_norma.idnorma)
8	-> Seq Scan on dat_regla (cost=0.00..1916.00 rows=50000 width=194) (actual time=0.046..473.323 rows=50000 loops=1)
9	-> Hash (cost=1333.46..1333.46 rows=1 width=17) (actual time=1321.676..1321.676 rows=1 loops=1)
10	-> Seq Scan on dat_norma (cost=0.00..1333.46 rows=1 width=17) (actual time=11.514..1321.665 rows=1 loops=1)
11	Filter: (idactividad = 1::numeric)
12	-> Index Scan using idactividad on nom_actividad (cost=0.00..6.11 rows=1 width=138) (actual time=8.663..8.666 rows=1 loops=1)
13	Index Cond: (nom_actividad.idactividad = dat_regla.idactividadquegenera)
14	-> Index Scan using nom_indicador_pkey on nom_indicador (cost=0.00..8.40 rows=1 width=136) (actual time=105.831..105.837 rows=1 loops=1)
15	Index Cond: (nom_indicador.idindicador = dat_norma.idindicador)
16	Total runtime: 1965.810 ms

Ilustración 17: Resultado de buscar reglas generadas por una actividad.

Cada consulta fue ejecutada 4 veces buscando el tiempo promedio de ejecución (en milisegundos) de las mismas. La siguiente tabla muestra los resultados obtenidos en cada una de las ejecuciones, además del promedio por consulta de estas ejecuciones.

Consulta	Ejecución 1	Ejecución 2	Ejecución 3	Ejecución 4	Promedio
1	56510,546	62406,332	54050,086	52891,851	56464,7038
2	23078,481	24148,032	22731,801	24007.,342	23319,438
3	858,015	161,985	1328,751	150,375	624,7815
4	1267,134	115,701	1965,810	333,30	920,48625

Tabla 7: Resultados obtenidos de las consultas.

Las pruebas de carga también son utilizadas para observar el comportamiento de una aplicación frente a cierta cantidad de peticiones de usuarios y someter a la aplicación al límite de su funcionamiento. Estas pruebas tienen como objetivo determinar la robustez de la aplicación cuando la carga es extrema, ayudando así a determinar el límite real en cuanto a número de usuarios concurrentes.

Se utilizó el Apache JMeter para la realización de dichas pruebas, el cual está configurado directamente con el servidor de BD. Primeramente se crearon un grupo de hilos donde se definieron la cantidad de usuarios que realizarían peticiones al servidor como muestra la siguiente figura.

Ilustración 18: Configuración de Grupo de Hilos.

Luego se configuró el acceso a la BD a través del archivo Configuración de la conexión JDBC.

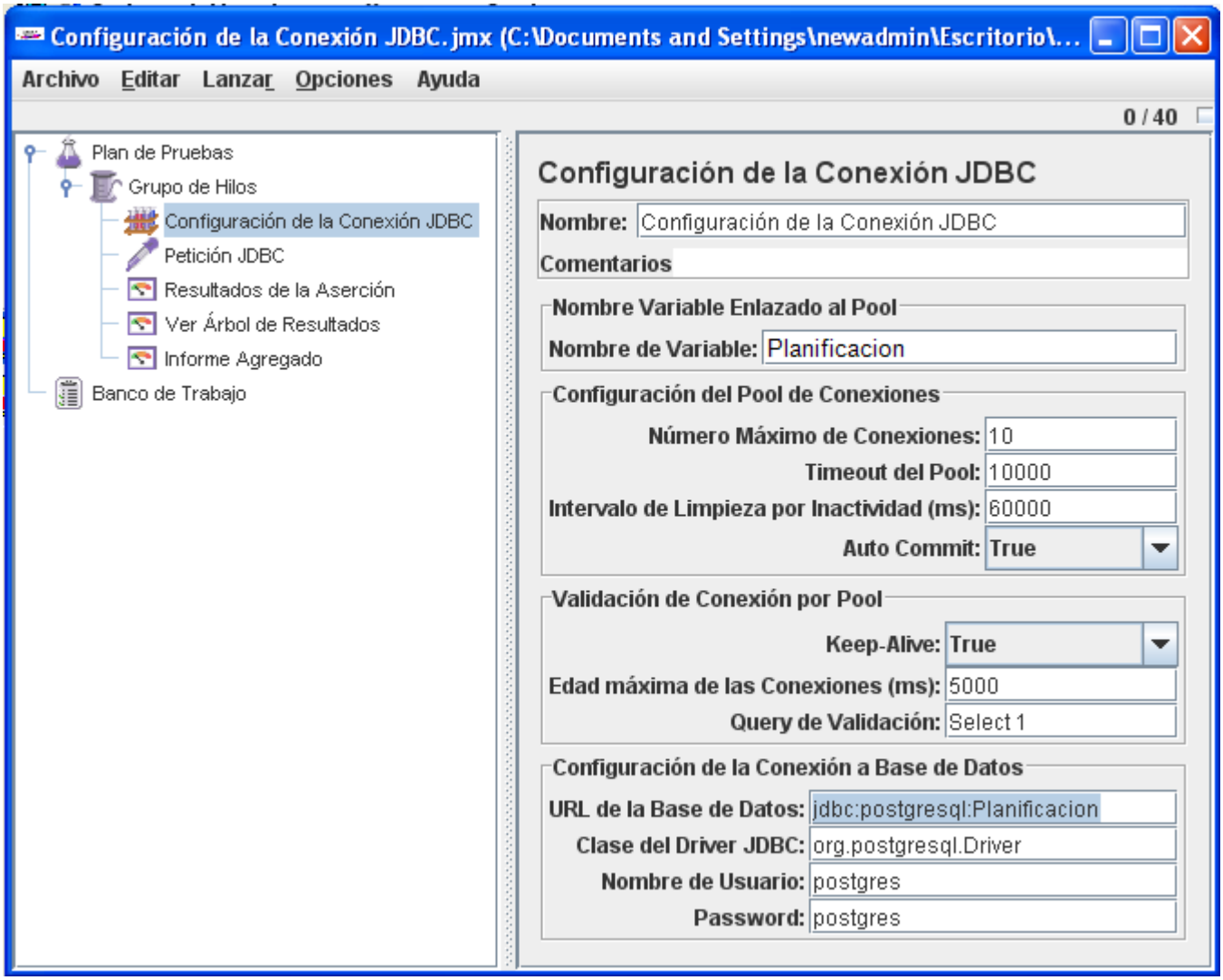


Ilustración 19: Configuración de la conexión JDBC.

Finalmente, se definió la petición que se realizaría a la BD en el archivo Petición JDBC.

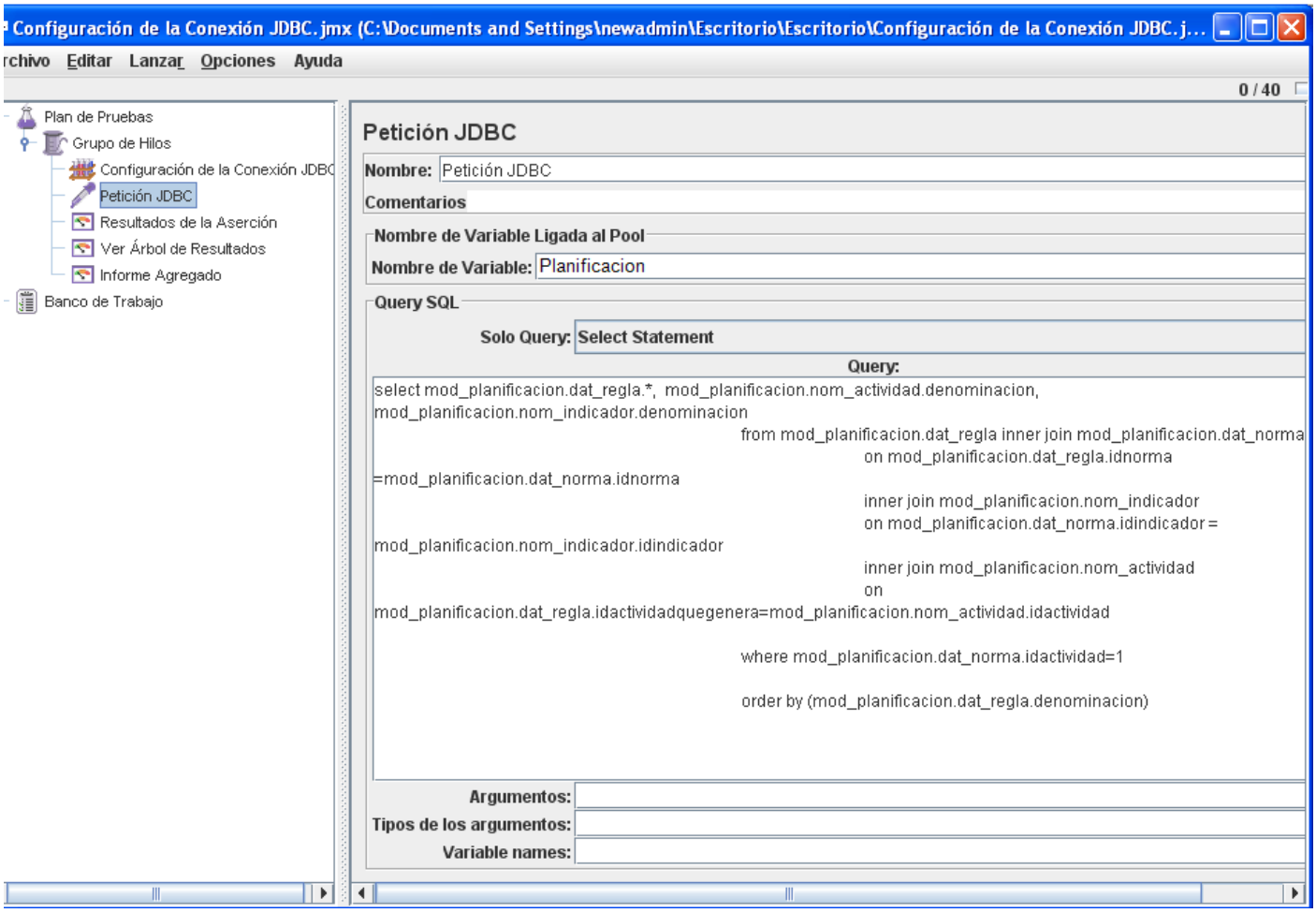


Ilustración 20: Petición JDBC.

Se tuvieron en cuenta las siguientes variables:

- Media: 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 tiempo de respuesta de una petición.
- Máximo: máximo tiempo de respuesta de una petición.

Las pruebas se realizaron sobre una misma consulta, pero con cantidad de usuarios conectados diferentes. La consulta escogida es buscar reglas generadas por una actividad, descrita anteriormente.

Usuarios concurrentes	Media(ms)	Mínimo(ms)	Máximo(ms)	Mediana(ms)
4	753	743	763	754
40	1748	896	2573	1831

Tabla 8: Resultado de la prueba.

3.4 Conclusiones parciales

Con la validación del diseño propuesto se chequearon un conjunto de parámetros importantes a tener en cuenta para realizar un correcto diseño de BD, especificando las restricciones de integridad y realizando el proceso de normalización en el cual se analizó cuidadosamente cuán factible sería desnormalizar el modelo e introducir redundancia en el mismo con el objetivo de ganar en rendimiento. Las pruebas realizadas permitieron conocer cómo respondería la BD ante determinadas situaciones. El comportamiento fue normal y esperado atendiendo a que el servidor escogido para las pruebas no cumple con las condiciones óptimas, en general se obtuvieron resultados satisfactorios.

Conclusiones

Concluido el presente trabajo de investigación se puede afirmar que se ha cumplido con el objetivo del mismo a través del diseño e implementación de una BD que permitió centralizar la información del subsistema planificación material y financiera. Terminada la investigación se tienen las siguientes conclusiones:

- Durante la investigación se realizó un estudio acerca de los elementos teóricos principales relacionados con el diseño de base de datos que favoreció el desarrollo del trabajo permitiendo sentar las bases para un correcto diseño de BD.
- Se realizó el diseño de la BD, el cual cumple con los requerimientos especificados, aplicando diversas soluciones que contribuyeron a mejorar el diseño.
- Se implementó el acceso a datos, lo cual permitió transformar el modelo entidad-relación al lenguaje de definición de datos del gestor correspondiente, en este caso el modelo relacional, estableciendo las estructuras de almacenamiento adecuadas.
- Se analizó el comportamiento de la BD ante situaciones determinadas a través de diferentes pruebas, a las cuales respondió satisfactoriamente.

De manera general se obtuvo una BD que permite el almacenamiento de la información requerida, permitiendo mayor organización y control de la misma.

Recomendaciones

Luego de terminado el presente trabajo se recomienda:

- Que se tenga presente el diseño propuesto para futuras versiones del subsistema Planificación material y financiera.
- Dar continuidad al trabajo realizado en pos de robustecer el diseño.

Referencias Bibliográficas

1. **Cortiñas, Jesús López.** ApuntesGestion. [En línea] 2009. http://www.apuntesgestion.com/2009/01/09/3-fundamentos-basicos-de-la-planificacion-empresarial/.

2. **Martínez, Lic. Y.** GestioPolis. [En línea] 2008. http://www.gestiopolis.com/economia/proceso-de-planificacion-empresail-en-cuba.htm.

3. **mastermagazine.** mastermagazine . [En línea] 2005. http://www.mastermagazine.info/termino/4908.php.

4. **publidirecta.** publidirecta. [En línea] 2010. http://www.publidirecta.com/dicc/diccionario-marketing_e.php.

5. **Borrego, Daniel.** [En línea] 2009. http://www.herramientasparapymes.com/erp-openbravo.

6. **Gil, Sergio Catala.***Informe de envaluación de ERP.* 2009.

7. **openXpertya.** openXpertya. [En línea] 2010. http://www.openxpertya.org/index.php?option=com_content&task=view&id=4&Itemid=5.

8. **EcuRed.** EcuRed. [En línea] 2011. http://www.ecured.cu/index.php/Versat_Sarasola.

9. **Gonzalez, Henry Raul.** Knol. [En línea] 2009. http://knol.google.com/k/henryraul/erp-cuba/3hor39g24xzh5/5#.

10. **García, Rosa María Mato.***Diseño de Base de Datos.* 1999.

11. **Date, J.C.***Introducción a los Sistemas de Base de Datos.* 2003.

12. **Definición.de.** Definición.de. [En línea] 2008. http://definicion.de/modelo-de-datos/.

13. **Camallea Núñez.***Gestión de Base de Datos con ADO.NET.* 2004.

14. **Hansen, Gary W.***Diseño y Administración de base de datos.* 1997.

15. **García Chávez, Carlos Alberto .** [En línea] 2005. http://www.mailxmail.com/curso-diseno-base-datos-relacionales/planificacion-diseno-administracion-bases-datos.

16. **RODAS, C. .** [En línea] 2009. http://www.slideshare.net/ChristianR/diseo-de-base-de-datos.

17. **Zorrilla Pantaleón, M. E.***Modelos de datos.* 2009.

18. **Pavón Mestras, Juan.***Estructura de las Aplicaciones Orientadas a Objetos.* 2009.

19. **Larman, C.***UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2004.

20. **Doctrine.** Doctrine. [En línea] 2008. http://www.doctrine-project.org.

21. **Paradigm, Visual.** Visual Paradigm. [En línea] 2006. http://www.visual- paradigm.com/product/vpuml/.

22. **Manager.net, SQL.** [En línea] 2011. http://www.sqlmanager.net/products/postgresql/.

23. **Gómez, L. B Oduardo.***Análisis y diseño del Componente del Sistema de Ingresos dentro del ERP Posta.* 2008.

24. **Olivares Rojas, Juan Carlos .***Patrones de Diseño.* 2010.

25. **Pérez Rodríguez, Maiby .***Módulo para la elaboración del Anteproyecto de Planificación del Sistema CEDRUX.* 2009.

26. **Riordan, Rebeca M.***Diseño de bases de datos relacionales con Acces y SqlServe.* 2000.

27. **Valdivia Báez, Igniris .** 2010 : s.n., Desarrollo del módulo de base de datos para un SCADA.

28. **CASELLES, J.** [En línea] 2008. http://www.coninteres.es/sql/material/Proceso_de_Normalizacion.pdf.

Glosario de términos

1. Compiere: aplicación para negocios de código abierto, ERP y CRM destinada para las empresas de pequeño y mediano tamaño.
2. MVC (en inglés Model-View-Controler): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.
3. Java: es un lenguaje de programación orientado a objetos.
4. PostgreSQL: es un sistema de gestión de base de datos relacional orientada a objetos y libre.
5. Oracle: es un sistema de gestión de base de datos objeto-relacional.
6. Apache: es un servidor web HTTP de código abierto para plataformas Unix.
7. Tomcat: también llamado Jakarta Tomcat o Apache Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation.
8. Python: es un lenguaje de programación de alto nivel.
9. XML: es el acrónimo en inglés de Lenguaje de Marcas Extensible (Extensible Markup Language), es un metalenguaje extensible de etiquetas. Permite definir la gramática de lenguajes específicos. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.
10. B2B: es el nombre con el que se conocen las transacciones entre empresas, que deciden comprar y vender sus productos a través de una red.
11. B2C: describe el comercio electrónico que surge entre compañías y consumidores.
12. B2E: es la relación comercial que se establece entre una empresa y sus propios empleados.
13. J2EE: es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java.

14. Cliente-Servidor: Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores.

15. Delphi: es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.

16. Windows: es el nombre de una serie de sistemas operativos.

17. Microsoft SQL Server: es un sistema para la gestión de bases de datos basado en el modelo relacional.

18. SGBD (Sistema de Gestor de Base de Datos): conjunto coordinado de programas, procedimientos, lenguajes. Que suministra tanto a usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos, manteniendo su integridad, confidencialidad y seguridad.

19. Notación Pascal Casing: los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

20. UCID: Unidad de Compatibilización e Integración de Software para la Defensa.

21. Tupla: es el nombre que recibe cada una de las filas de la tabla.

22. Trigger: un trigger (o disparador) en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

23. ORM: mapeador de Objetos Relacionales.