Universidad de las Ciencias Informáticas

Facultad 6

Título: Plugin para la integración del proceso de normalización de bases de datos relacionales en la herramienta de administración de bases de datos HABD.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es):

Osmil Guillén Tamayo

Grether Méndez Gómez

Tutor(es):

Msc. Yanet Espinal Martín

Ing. Mikel Díaz Hernández

La Habana, 2012, ¨Año 54 de la Revolución¨.

Frase

¨La enseñanza que deja huella no es la que se hace de cabeza a cabeza, sino de corazón a corazón¨...

Howard G. Hendricks

Dedicatoria

Grether

A mi mamá, papá, hermano, cuñada y a mi tío Omar por su amor y apoyo incondicional. A mi familia, que de una u otra forma me ha ayudado a convertirme en quién soy.

A a mi sobrino, para que se sienta orgulloso de tener una tía que lo ama.

En especial a mis abuelos que aunque ya no estén aquí, se que desde ese cielo azul me están cuidando.

A mi padrino Nápoles que aunque es un poco gruñón, siempre a sabido extenderme la mano.

A Osmil por ser un compañero de tesis incondicional y excelente novio.

A mis tutores Yanet Espinal Martín y a Mikel Díaz Hernández, por habernos guiado en esta última travesía en la universidad.

A mis amigos de Artemisa y en especial a Dayana, Maybis y Sara que aunque no están presente en este momento tan especial, están en nuestros corazones y a todos lo que convivieron conmigo este año tan crucial: a mi compañera de cuarto y valerosa amiga Leysi, Sadis, a la compañerita Ivet, gracias por darnos café para poder conciliar el sueño, en fin a todos.

Dedicatoria

Osmil

A mis queridos padres y mi linda hermanita por molestarse y apoyarme en el momento preciso, por tener siempre un consejo útil y una sonrisa incondicional cuando todo parecía difícil.

A mis tutores Yanet y Mikel por saber olvidarse de sus problemas y ayudarnos a resolver los nuestros, por dedicarnos tiempo, espacio y enseñarnos a ser mejores cada día.

A mi querida novia Grether, por brindarme su apoyo incondicional.

A mis amigos de siempre, quienes en menos de cinco años se han ganado un lugar para toda la vida, por comprender lo inexplicable, por hacerme reir de los problemas, por ayudar en lo difícil y por ser una mano amiga en todo momento.

Agradecimientos

Grether y Osmil

A nuestros padres y hermanos, por el apoyo incondicional que nos han brindado, y a quienes les debemos todo lo que hemos logrado en la vida.

A nuestros tutores Yanet y Mikel, quienes con paciencia y dedicación nos dieron la clave para ser mejores cada día y nos ayudaron a cumplir este sueño de 5 años.

A los amigos con los que hemos compartido la mejor etapa de nuestras vidas, por hacer de cada día juntos un recuerdo inolvidable.

A Nela, Glennys, Adnan y David Silva por extendernos la mano cuando más falta nos hacía.

A Edgar por ser un ejemplo a seguir.

A todos los que han contribuido de una forma u otra, para hacer realidad este sueño.

Declaración de autoría

Declaramos ser autores del presente trabajo “Plugin para la integración del proceso de normalización de bases de datos relacionales en la herramienta de administración de bases de datos HABD” y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2012.

_____	_____
Grether Méndez Gómez	Osmil Guillén Tamayo
Firma del Autor(a)	Firma del Autor(a)
_____	_____
Msc. Yanet Espinal Martín	Ing. Mikel Díaz Hernández
Firma del Tutor(a)	Firma del Tutor(a)

RESUMEN

Dado el crecimiento vertiginoso de los volúmenes de información en la sociedad actual y por consiguiente, el aumento de la información manipulada por los sistemas informáticos, se hace necesario optimizar el tratamiento de los grandes lotes de información que genera la sociedad, así como, integrar dicha optimización con herramientas de administración de bases de datos existentes. En la Universidad de las Ciencias Informática, específicamente en el departamento PostgreSQL, perteneciente al centro de desarrollo DATEC, se creó la herramienta de administración de bases de datos HABD, la cual no posee funcionalidades que garanticen el proceso de normalización de bases de datos relacionales.

Por esta razón se decide desarrollar un plugin que permita integrar el proceso de normalización de bases de datos relacionales a la herramienta de administración de bases de datos HABD. Para el desarrollo de la misma se seleccionaron las tecnologías y herramientas a utilizar que más se adecuaron a las características del proyecto y a las políticas de uso de software libre de la universidad. Se identificaron y documentaron las funcionalidades a cumplir por la aplicación y se generaron los artefactos correspondientes a cada fase del proceso de desarrollo de software guiado por la metodología Extreme Programing. La realización del plugin denominado “NBR” constituirá una alternativa de gran ayuda en la utilización de las bases de datos relacionales, evitando anomalías que pueden existir en las mismas, tanto en los ambientes productivos, como no productivos.

Palabras Claves: Bases de Datos Relacionales; Normalización; Diseño de Bases de Datos; Sistemas Informáticos.

Índice

Índice

Índice de ilustraciones

Índices de Ilustraciones

Índice de tablas

Índice de Tablas

INTRODUCCIÓN

Hablar de computación, es hablar de un tema apasionante en todos los sentidos, hace soñar sobre el futuro, hace discutir sobre las tecnologías apropiadas y sus costos, las políticas para desarrollar una industria, institución y un país. Pero fundamentalmente hablar de computación o informática es hablar de la necesidad de recursos humanos capacitados, de los cambios en la forma de trabajar y los nuevos empleos, de las nuevas posibilidades de desarrollo individual y hasta de aprendizaje con la inserción de la computadora.

El ser humano siempre ha necesitado encontrar métodos rápidos y efectivos para resolver sus cálculos, y con ayuda de su gran inventiva ha conseguido a través de los siglos desarrollar las computadoras. Hoy día las personas están acostumbradas a vivir con ellas, y no se percatan de que su aparición ha tenido gran influencia en diversos aspectos de la vida diaria, mejorándola y abriendo puertas que antes eran desconocidas para la humanidad.

La información en su sentido más amplio, es comunicación del conocimiento y ha sido fundamental en todas las sociedades. El incremental avance tecnológico y los grandes volúmenes de información que se manejan en estos tiempos han proporcionado un elevado desarrollo y utilización de los Sistemas de Gestión de Información (SGI), los cuales se han convertido en un factor indispensable en la evolución de la sociedad. Con el incremento de la información en la sociedad, se hace cada vez más engorroso el trabajo con la información de forma manual en los SGI, lo que de una forma u otra provoca ciertas dificultades en la manipulación y gestión de la información, este fue uno de los inconvenientes principales que dieron paso al surgimiento y desarrollo de las bases de datos.

En el proceso y construcción de toda aplicación informática, el diseño de las bases de datos (BD) ocupa un lugar importante dentro de su desarrollo; a tal punto que esta puede verse como un proceso relativamente independiente dentro de la confección de un sistema.

En el diseño de una Base de Datos Relacional (BDR) pueden encontrarse anomalías de inserción, eliminación, actualización de los datos, así como restricciones artificiales en la estructura de los mismos y dependencia entre ellos, implicando grandes problemas en la gestión y obtención de su información.

En la actualidad existen varias herramientas de administración de bases de datos, donde el estudio de estas arrojó la no existencia de funcionalidades que permiten comprobar y dar solución a estas anomalías que se encuentran en el diseño de las BD.

Introducción

La Universidad de las Ciencias Informáticas (UCI) es una institución que se caracteriza por formar profesionales comprometidos con la Revolución y altamente calificados en la rama de la informática. En la misma se han creado varios Centros de Desarrollo, uno de estos es el Centro de Tecnologías de Gestión de Datos (DATEC), en el cual se encuentra el departamento PostgreSQL, encargado de investigar directamente los temas referentes al Sistema Gestor de Bases de Datos (SGBD) PostgreSQL. Actualmente en dicho departamento se encuentra en desarrollo una herramienta de administración de bases de datos llamada HABD, surgida por los inconvenientes encontrados en las distintas herramientas de administración de bases de datos, tomando como punto de partida el PgAdmin3[1].

HABD provee una interfaz amigable y una arquitectura basada en plugin, lo que permite a los desarrolladores incorporar nuevos servicios de manera sencilla. De esta forma se le pueden agregar gran número de funcionalidades, evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema.

Dicha herramienta, no posee funcionalidades que permitan la normalización de BDR, lo que induce pérdida en la calidad de los modelos de datos e insuficiente integración para etapas futuras de la implementación de estos modelos físico en la HABD.

Por todo lo antes expuesto se propone como **problema de la investigación:** ¿Cómo integrar el proceso de normalización de bases de datos relacionales en la herramienta de administración de bases de datos HABD?

Para darle solución al problema de la investigación formulado anteriormente, se plantea como **Objetivo General:** Desarrollar un plugin para la herramienta de administración de bases de datos HABD que permita normalizar bases de datos relacionales.

Se propone como **Objeto de estudio**: *Diseño de bases de datos relacionales.*
Enmarcado en el **Campo de acción:** *Tecnologías para la integración del proceso de normalización de bases de datos relacionales en la herramienta de administración de bases de datos HABD.*

Objetivos específicos:

- Analizar el proceso de normalización de bases de datos relacionales y su integración con la herramienta HABD.
- Diseñar el plugin de normalización de bases de datos relacionales para HABD.
- Implementar el plugin de normalización de bases de datos relacionales para HABD.

[1] *Es una herramienta de código abierto para la administración de bases de datos PostgreSQL.*

- Validar el plugin de normalización de bases de datos relacionales para HABD.

Tareas de la Investigación:

1. Identificación de la teoría y empleo de la normalización de bases de datos relacionales dentro del diseño de las bases de datos.
2. Identificación de las tendencias y tecnologías actuales más apropiadas para la construcción de plugin.
3. Definición del modelo de dominio.
4. Elaboración de las historias de usuarios.
5. Elaboración del plan de iteraciones.
6. Realización del modelo de diseño.
7. Implementación de las funcionalidades identificadas para la realización del proceso de normalización hasta la forma normal de Boyce-Codd.
8. Implementación de las funcionalidades identificadas para la integración del componente con la herramienta de administración de bases de datos HABD.
9. Definición de los métodos de pruebas a utilizar en la validación del plugin de normalización de bases de datos relacionales.
10. Realización de los diseños de casos de pruebas para el plugin de normalización de bases de datos relacionales.

Posibles resultados:

- Artefactos resultantes del proceso de desarrollo según la metodología empleada.
- Plugin para la normalización de bases de datos relacionales en la herramienta de administración de bases de datos HABD.

El presente trabajo de diploma, está compuesto por Introducción, Tres Capítulos, Conclusiones, Recomendaciones, Referencias Bibliográficas, Bibliografías, Anexos y Glosario de Términos.

El Primer Capítulo “Fundamentos Teóricos”, expone una evaluación del estado del arte del surgimiento de las BD y los diferentes modelos que han existido hasta la actualidad, definiciones y características de los mismos, haciendo énfasis fundamentalmente en las BDR, para entrar en el estudio de su etapa de diseño, principalmente en el proceso de normalización, y la integración de dicho proceso con la herramienta de administración de bases de datos HABD.

El Segundo Capítulo “Características del sistema propuesto”, se expone los diferentes elementos que fundamentan el desarrollo de la propuesta. Donde se describe la herramienta que se propone a través de las historias de usuario y los requerimientos tanto funcionales y no funcionales. De esta forma se generan los artefactos que propone la metodología XP.

El Tercer Capítulo “Validación del sistema propuesto”, describe las técnicas de validación y pruebas aplicadas al sistema, para validar que las tareas realizadas en el diseño y los requisitos cumplen con las normativas planteadas por los clientes.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS

INTRODUCCIÓN

Este capítulo brindará una descripción detallada del estado del arte del proceso de normalización de las BDR, así como también abordará la fundamentación de cómo integrar este proceso con HABD a través de un plugin; apoyándose en características, definiciones y ventajas para la selección de las herramientas y medios necesarios que garantizarán el éxito de esta investigación.

1.1 CONCEPTOS ASOCIADOS A LA INTRODUCCIÓN

Al adentrarnos en la investigación es necesario conocer algunos conceptos fundamentales que proporcionarán un mejor entendimiento de los términos utilizados a lo largo de este capítulo.

1.1.2 INTRODUCCIÓN A LAS BASES DE DATOS

A lo largo de los años el hombre estudió la manera más eficiente de almacenar grandes volúmenes de información y que esta estuviera disponible para ser consultada y gestionada en cualquier momento. Con el surgimiento de las nuevas tecnologías y el despegue de aplicaciones informáticas, aparecieron programas para almacenar los datos en forma de archivos, lo cual era más rápido, sencillo y fiable para manipular la información, pero aun así tenían grandes dificultades a la hora de querer modificar registros, estructuras o simplemente buscar información. A raíz de estas necesidades nacen las Bases de Datos, en estas se guardan los datos utilizados por los usuarios y con el paso del tiempo se han convertido en una de las herramientas más importantes de la sociedad, utilizadas como vía de almacenamiento y recuperación de la información a nivel científico, social, económico, político y cultural.

Base de Datos

Entre las bibliografías más consultadas en la actualidad referente al tema de las BD, destacala del Dr. Christopher J. Date. Este define a las BD como: un conjunto de datos persistentes[1] que es utilizado por los sistemas de aplicación de alguna empresa dada. (1)

Se concluye que las BD no son más, que grandes volúmenes de informaciones relacionadas y organizadas, que se pueden almacenar y recuperar.

[1] *El tipo de datos de la BD difiere de otros datos más efímeros, como los datos de entrada, los datos de salida, las instrucciones de control, los bloques de control de software, los resultados intermedios.*

1.1.3 Introducción a los Sistemas Gestores de Bases de Datos

Los SGBD se diseñan para manejar grandes cantidades de información. Esta gestión de los datos incluye tanto la definición de las estructuras para el almacenamiento como los mecanismos de acceso a los datos. Además debe cuidar la seguridad de la información almacenada en la BD, tanto contra las caídas del sistema como contra los intentos de acceso no autorizado.

Sistema de Gestión de Bases de Datos

Según la Lic. Rosa María, en su libro "Diseño de Bases de Datos": un SGBD es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez. (2)

De igual manera, un SGBD es un sistema que permite a los usuarios manipularlas BD, es decir: definir, crear y eliminar los datos que se manipulan, proporcionándole a este un acceso controlado.

Objetivos

El objetivo de los SGBD es servir de interfaz entre la BD, el usuario y las aplicaciones, definiendo los datos a distintos niveles de abstracción y garantizando la seguridad e integridad de los mismos. Entre los más utilizados se encuentran: Oracle, DB2, PostgreSQL, MySQL, MS SQL Server y otros más.

1.1.4 Sistemas de Administración de Bases de Datos

Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. El Sistema de Administración de Bases de Datos (DBMS) es un conjunto de servicios (aplicaciones de software) para administrar bases de datos. (3)

Los Sistemas de Administración de Bases de Datos permiten:

- Acceso a la información por parte de múltiples usuarios.
- Manipulación de objetos en las bases de datos (insertar, eliminar, editar).

Las necesidades actuales de empresas productoras de software y los problemas de pago de licencias a precios elevados han inclinado la balanza hacia la utilización de software libre. Entre los DBMS más usados en la actualidad, creados bajo software libre se encuentran:

- **PgAdmin3**

PgAdmin3 es la herramienta Open Source de administración por excelencia para BD PostgreSQL. Algunas de sus características son: el soporte completo para UNICODE[1], edición rápida de consultas y datos multihilo y soporte para todos los tipos de objetos de PostgreSQL. Incluye una interfaz gráfica de administración, una herramienta para el trabajo con Lenguaje de Consulta Estructurado (SQL), un editor de código de procedimientos y funciones. Está diseñado para darle respuesta a las necesidades de la mayoría de los usuarios, desde la escritura de consultas simples en SQL hasta el desarrollo de BD complejas. La interfaz gráfica soporta todas las características presentes de PostgreSQL y se puede hacer la administración fácilmente. (4)

- **phpPgAdmin**

Herramienta web muy parecida al popular phpMyAdmin, mejorado para PostgreSQL, brinda soporte para la navegación y modificación de la mayoría de los tipos de objetos de las BD PostgreSQL, además de la ejecución de consultas ad-hoc[2]. Mantenido por el equipo de phpPgAdmin. (5)

- **PostgreSQL PHP Generator**

PostgreSQL PHP Generator es una herramienta GUI de PostgreSQL de tipo freeware pero muy poderosa, que permite generar scripts PHP de gran calidad para las tablas seleccionadas, vistas y consultas para el trabajo adicional con estos objetos a través de la web. (6)

- **PGAccess**

Es la herramienta gráfica original de PostgreSQL, con un buscador de BD al estilo de MS-Access, desarrollado en Tcl/Tk. Permite la edición, adición y navegación de BD, tablas, vistas, funciones, secuencias y usuarios, así como también el desarrollo gráfico de consultas. Un formulario y un diseñador de reportes están bajo desarrollo por el equipo de PGAccess.(7)

Todas estas herramientas potentes y muy utilizadas por la mayoría de las empresas, garantizan un adecuado trabajo con las BD.

---

[1] *Proporciona un número único para cada carácter, sin importar la plataforma, sin importar el programa, sin importar el idioma.*

[2] *Es un pedido o solicitud de información que se ha de extraer de una Base de Datos. La consulta es iniciada directamente por el Usuario, en vez de ser iniciada indirectamente por medio de un Programa de Aplicación que brinde menú u otras estructuras para formular la consulta.*

1.1.5 Modelos de Bases de Datos, Bases de Datos Relacionales

Emitido por el Dr. Ian Graham, en el libro “Métodos Orientados a Objetos”, el cual pronuncia que es un formalismo matemático que consta de una notación para describir los datos y las estructuras de datos (información), y de un conjunto de operaciones válidas que se pueden utilizar para manipular estos datos, o al menos, los símbolos que los representan. (8)

Se concluye que los Modelos de Bases de Datos, no son más que simples algoritmos o funciones matemáticas que permiten la implementación de un sistema eficiente de BD.

Los modelos de datos pueden ser divididos en tres grandes tipos (9)**:**

1. **Modelos lógicos basados en registros:** Modelos utilizados para describir los datos en los modelos conceptual y físico. A diferencia de los modelos lógicos basados en objetos, se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.

- Modelo jerárquico.
- Modelo red o reticular.
- Modelo relacional.

2. **Modelos lógicos basados en objetos:** Modelos utilizados para describir datos en el nivel conceptual y el externo. Se caracterizan por proporcionar capacidad de estructuración bastante flexible y permiten especificar restricciones de datos.

- **Entidad-Relación:** Se basa en una percepción del mundo compuesta por objetos, llamados entidades, y relaciones entre ellos. Las entidades se diferencian unas de otras a través de atributos
- **Semántico:** Modelos dedicados específicamente a representar la realidad sobre la cual versa la BD.
- **Orientado a Objetos:** Se basa en objetos, los cuales contienen valores y métodos, entendidos como órdenes que actúan sobre los valores, en niveles de anidamiento. Los objetos se agrupan en clases, relacionándose mediante el envío de mensajes.

Dentro de los modelos lógicos basados en objetos, el más extendido es el modelo entidad-relación, seguido por el modelo orientado a objetos.

3. **Modelos físicos de datos:** Describen como se almacenan los datos en la computadora, representando informaciones tales como: las estructuras de los registros, el ordenamiento de los registros y las rutas de acceso. No abundan tantos modelos físicos como lógicos, los más usuales son: modelo unificador y la memoria de marco.

Bases de Datos Relacionales

Introducir como nuevo paradigma el concepto de Bases de Datos Relacionales (BDR) fue un hecho trascendental en la época de los setenta, por el Dr. Edgar Frank Codd[1], tras ser postuladas como: **"Modelo Relacional"**. Por lo que podemos afirmar que una BD que cumple con el modelo relacional, se le llama **BDR**, estas permiten establecer relaciones entre los datos que una vez fueron guardados en tablas, y a través de estas relacionar los datos de ambas tablas. La estructura fundamental del modelo relacional es la **"relación"**, es decir una tabla bidimensional constituida por filas (tuplas)[2]y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la BD. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representan las propiedades de la entidad. (10)

Modelo Entidad-Relación

En 1976, el Dr. Peter Chen presentó el modelo entidad-relación, que es la técnica más utilizada en el diseño de BD, con el objetivo de erradicar las deficiencias del modelo relacional para representar los datos de una manera más real. (11)

A pesar de la claridad y facilidad que aportaba el modelo E-R para modelar problemas reales, el modelo E-R era exclusivamente estático, expresaba las entidades de la realidad y sus relaciones. El diseño conceptual basado en los Diagramas Entidad Relación (DER) adolecía, por tanto, de dinámica. Para paliar esta carencia, inicialmente se solía combinar el E-R con los populares Diagramas de Flujo de Datos (DFD). (12)

[1] *Brillante matemático y científico de la computación, creador de modelo de datos relacional y toda su teoría, trabajó toda su vida en la IBM, donde realizaba sus labores de investigación en el área de la computación, considerado uno de los más prestigiosos en el campo de la informática.*

[2]*Las tuplas en una relación son un conjunto en el sentido matemático del término, es decir una colección no ordenada de elementos diferentes.*

En 1979, Codd intentó subsanar algunas de las deficiencias de su modelo relacional con una versión extendida denominada RM/T en 1979 y posteriormente RM/V2 en 1990 (13). Alrededor de la mitad de los ochenta, algunas aplicaciones exigían mayor expresividad en los datos con los que trabajaban. Como respuesta a la creciente complejidad de dichas aplicaciones surgió el nuevo modelo de datos: orientado a objetos, esta evolución representa la "tercera generación de los SGBD".

Modelo orientado a objeto, bases de datos orientados a objetos

Las bases de datos orientadas a objetos (BDOO) se crearon para tratar de satisfacer las necesidades de estas nuevas aplicaciones. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de BD tradicionales. Una característica clave de las BDOO es la potencia que proporcionan al diseñador al permitirle especificar tanto la estructura de objetos complejos, como las operaciones que se pueden aplicar sobre dichos objetos.

El modelo orientado a objetos también soporta relaciones de muchos a muchos, siendo el primer modelo que lo permite. Aun así se debe ser muy cuidadoso cuando se diseñan estas relaciones para evitar pérdidas de información. (13)

Diseño de Bases de Datos Relacionales

El objetivo del diseño de una BDR es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. Este se ha convertido en una actividad popular, desarrollada no solo por profesionales sino también por no especialistas. Véase en la Ilustración 1, el diseño de BDR, por la Dra. Letticia M. Seijas. (14)

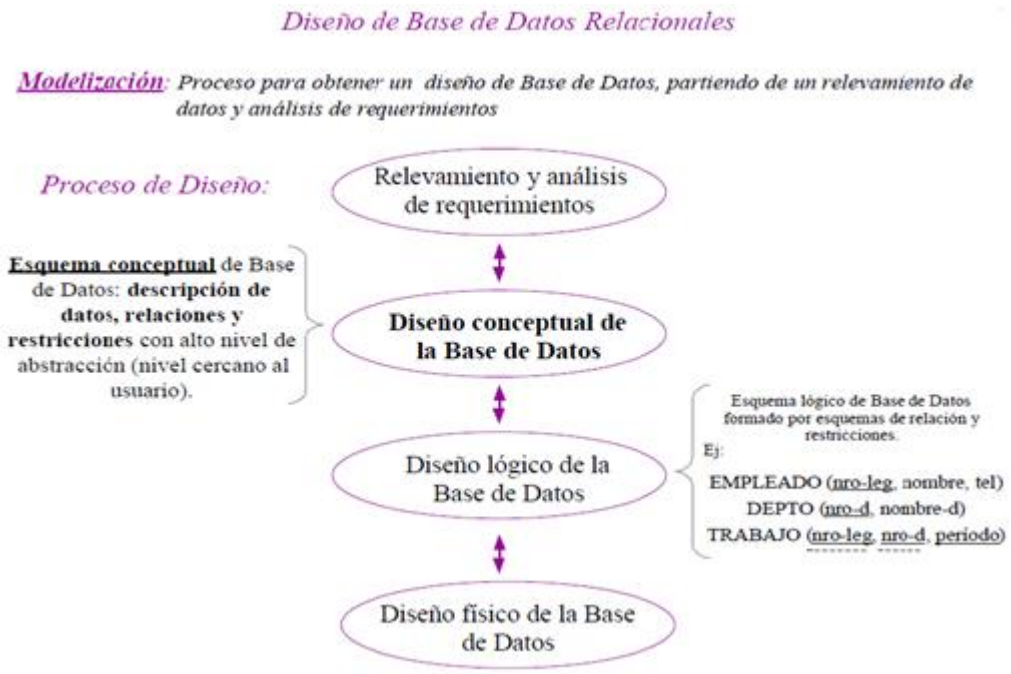


ILUSTRACIÓN 1: DISEÑO DE BASE DE DATOS RELACIONALES.

La ilustración muestra el modelamiento del diseño de BDR, a través del levantamiento de los datos y el análisis de los requerimientos.

1.1.6 Introducción a la Normalización

Hoy en día la mayoría de los diseñadores no confían en las metodologías de diseño de BD, esta es una de las causa principales que dan paso a que fracase el desarrollo de los sistemas de información. Dado la no utilización de estas metodologías provocan regularmente que la información que se gestiona sea limitada y con el riesgo de no poder extender el diseño sin tener que reestructurar la BD, al surgir nueva información relacionada con lo modelado. Muchos de estos problemas se deben a la falta de claridad que permita entender la naturaleza exacta de los datos.

Surgimiento de la Normalización

La normalización de BDR fue introducida por el *Dr. Edgar F. Codd* a principios de los 70, poco después de crear el modelo relacional (12), es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional, es una etapa posterior a la correspondencia entre el esquema conceptual y el esquema lógico. La normalización pretendía añadir calidad a los diseños de BDR evitando la aparición de datos redundantes y los problemas de actualización que conllevan. Normalizar *esquemas de relación* de una BDR supone, en primer lugar, comprobar una serie de normas, llamadas "*formas normales"*, basadas en dos conceptos fundamentales: "*Dependencias Funcionales y Claves"*, y en segundo lugar, descomponer los esquemas que no las cumplen para obtener esquemas normalizados.

1.1.7 Tipos de Dependencias

Dependencias funcionales (DF)

En la actualidad existen muchas definiciones relacionadas con las dependencias funcionales en BDR. Las dependencias funcionales son reglas independientes del tiempo que verifican los atributos de un contexto determinado (15). Véase Ilustración 2.

Sea $R(\{A_i : Di\})$

$t,s \in R$ tuplas de R

X, Y subconjuntos de atributos $X \subseteq R(\{A_i : Di\})$, $Y \subseteq R(\{A_i : Di\})$

Se define una dependencia funcional (DF)

$X \rightarrow Y$ (X determina funcionalmente a Y o Y depende funcionalmente de X)

como una restricción sobre las tuplas de R que garantiza que dado un valor único de X, el valor de los atributos que componen Y ha de ser también único.

$X \rightarrow Y \models (t.X = s.X \Rightarrow t.Y = s.Y)$

ILUSTRACIÓN 2: CONCEPTO DEPENDENCIA FUNCIONAL.

Un esquema de relación **R** se ve enriquecido por la semántica asociada a las dependencias funcionales inherentes al contexto de los atributos **A**: **R (A, DF)** contiene las dependencias funcionales **DF** como restricciones semánticas.

Intuitivamente, dado un conjunto de dependencias funcionales, hay otras que se deducen como consecuencia de ellas. Al conjunto inicial de DF, completado con todas las dependencias que de él se deducen, se le denomina **cierre** y se representa por DF+ (obviamente: DF es subconjunto de DF+). El

cálculo de DF+ se fundamenta en los denominados "**Axiomas de Armstrong".** Véase Ilustración 3 (12)

Reflexividad: Si $\forall X$, entonces $X \rightarrow X$.

Aumentatividad: Si $X \rightarrow Y$ entonces $X' \rightarrow Y \quad X \subseteq \forall X'$

Proyectividad: Si $X \rightarrow Y$ entonces $X \rightarrow Y' \quad \forall Y' \subseteq Y$

Aditividad: Si $X \rightarrow Y$ y si $U \rightarrow W$ entonces $X \cup Y \rightarrow U \cup W$

Transitividad: Si $X \rightarrow Y$ y $Y \rightarrow Z$ entonces $X \rightarrow Z$.

ILUSTRACIÓN 3: AXIOMAS DE ARMSTRONG.

Dependencia funcional plena o completa

Definición: Se dice que el atributo **Y** de **R** depende funcionalmente por completo o totalmente del atributo **X** de **R**, si depende funcionalmente de **X** y no depende funcionalmente de ningún subconjunto de **X.** (16)

Dependencias funcionales transitivas

Definición: Entre dos descriptores **X** e **Y** tiene lugar una dependencia transitiva, cuando se cumplen las siguientes condiciones. Véase Ilustración 4 (16):

$X \cap Y = \emptyset$

$\exists Z : X \cap Z = \emptyset \; Y \cap Z = \emptyset$

$\exists Z : X \rightarrow Z, Z \nrightarrow X, Z \rightarrow Y$

ILUSTRACIÓN 4: DEPENDENCIA FUNCIONAL TRANSITIVA.

Claves o Llaves

Definición: Sea **R** un esquema de relación, **F** un conjunto de dependencias funcionales y **X⊆R**. Se dice que **X** es clave de **R** si y solo si **X** es un conjunto minimal de atributos que determina funcionalmente todos los atributos en **R**. Véase Ilustración 5 (16):

(a) $X \rightarrow R \in F^+$

(b) No existe $Y \subseteq X$, tal que $Y \rightarrow R \in F^+$ (X es minimal)

ILUSTRACIÓN 5: CLAVES O LLAVES.

Puede suceder que exista más de una clave para un esquema **R**. En casos como estos, suele designarse a una de ellas como **clave primaria**. El término **clave candidata** lo usaremos para aquellas claves que no han sido designadas como primarias. Cuando un conjunto de atributos **X** satisface la condición (a) pero no la condición de minimalidad, diremos que **X** es una **superclave**. Llamaremos **atributos primos o principales** a todos aquellos que participan en alguna clave, y **atributos no primos o no principales** a aquellos que no participan en ninguna clave.

1.1.8 Fases de la Normalización. Formas Normales

El término formas normales está dado por un conjunto de técnicas investigadas a partir de la década de 1970, por teóricos relacionales, como: Dr. Codd, Dr. Boyce, Dr. Ronald Fagin, para combatir las anomalías existentes en las BDR, pero no fue antes de 1970 que CODD y otros definieron la 1era, 2da y 3ra Forma Normal. Un poco después, en el año 1974, Boyce y Codd redefinieron la 3era forma normal, llamándola Forma Normal de Boyce-Codd (FNBC), debido a que esta no contemplaba algunos casos particulares. Posteriormente en 1977 y 1979 se definieron la Cuarta Forma Normal (4FN) y la Quinta Forma Normal (5FN), estas dos últimas creadas por el Dr. Ronald Fagin. (1)

El proceso de la normalización involucra varias fases que se realizan en orden lógico, como se mostrará a continuación:

- **Primera forma normal (1FN)**

  Definición: El término 1FN describe una relación en la cual (17):

  - ✓ Todos los atributos claves están definidos.
  - ✓ No existen grupos repetidos en la tabla. En otras palabras, cada intersección de fila/columna puede contener uno y solo un valor, no un conjunto de valores.
  - ✓ Todos los atributos son dependientes de la clave primaria.

- **Segunda forma normal (2FN)**

  Definición: El término 2FN describe una relación en la cual (18):

  - ✓ Está en 1FN.
  - ✓ Todos sus atributos no primos tienen dependencias funcionales total respecto a cada una de las claves.

Aún podemos observar que es posible que una relación que cumpla con 2FN exhiba dependencias transitivas; es decir uno o varios atributos pueden ser funcionalmente dependientes de atributos no primos.

- **Tercera forma normal (3FN)**

  Definición: El término 3FN describe una relación en la cual (18):

  - ✓ Está en 2FN.
  - ✓ Ningún atributo no primo depende transitivamente de ninguna clave.

- **Forma normal de Boyce-Codd (FNBC)**

  Definición: Una relación está en FNBC, si todo determinante[1] en ella es una clave candidata. (17)

Evidentemente, si una tabla contiene solamente una clave candidata, la 3FN y la de FNBC son equivalentes. Si se plantea esta proposición de otra manera, la FNBC puede ser violada solo si la relación contiene más de una clave candidata.

- **Cuarta forma normal (4FN)**

  Definición: El término 4FN describe una relación en la cual

  - ✓ Una relación está en 4FN si está en la FNBC.
  - ✓ No contiene dependencias multivaluadas[2].

- **Quinta Forma Normal (5FN)**

  Definición: El término 4FN describe una relación en la cual

  - ✓ Permite hacer frente a un tipo de dependencia denominada dependencia de unión
  - ✓ Suele presentarse cuando resolvemos tres (o más) entidades, todas relacionadas con una relación muchos-a-muchos a las otras.

Ventajas de la normalización:

- El sistema de BD no sufra de anomalías de almacenamiento.
  - ✓ **Redundancia de datos:** Repetición de datos en un sistema.
  - ✓ **Anomalías de actualización:** Inconsistencias de los datos como resultado de redundancias y actualizaciones parciales.
  - ✓ **Anomalías de inserción:** Imposibilidad de incluir datos en la BD debido a la ausencia de otros.

[1] *Es cualquier atributo cuyo valor determina otros valores de atributos.*
[2] *Existe una dependencia multivaluada cuando hay tres atributos (A, B y C) en una relación, tal que por cada valor de A existe un bien definido conjunto de valores de B y un bien definido conjunto de valores de C, sin embargo el conjunto de valores de B es independiente del conjunto C y viceversa.*

- ✓ **Anomalías de borrado:** Pérdida de información no intencionada debido a que se han borrado otros datos.

- El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una BD implantada sobre un modelo bien diseñado tiene mayor esperanza de vida, aunque aumente su tamaño o su ambiente sea dinámico, que una BD con un diseño pobre, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales. Se dice que un modelo que se encuentre en 3ra FN, se considera un modelo adecuado, aunque muchas bibliografías recomiendan normalizar hasta la FNBC. Dado que la 4ta y 5ta FN son para relaciones poco usadas en la práctica.

- La normalización de BDR se lleva a cabo por cuatro razones fundamentales:

  - ✓ Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
  - ✓ Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
  - ✓ Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
  - ✓ Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

En términos más sencillos la normalización trata de simplificar el diseño de una BDR, a través de la búsqueda de la mejor estructuración que pueda utilizarse con las relaciones involucradas en ella.

1.2 Herramientas existentes para la Normalización de Bases de Datos Relacionales

En la actualidad se cuenta con una serie de herramientas de normalización de BDR, a continuación se mencionan los nombres de algunas utilizadas en la sociedad actual: JMathNorm, NORMIT, WEB-BASED TOOL, DB Design Tools, LDBN–Learning Database Normalization. Estas herramientas garantizan el proceso de normalización pero no se integran a los SGBD, por tal motivo se crea la herramienta SINORGES en la UCI, diseñada por la tutora del presente trabajo de investigación, la cual permite eliminar o mejorar un grupo de deficiencias encontradas en las anteriores herramientas citadas. Para mayor información puede consultar el artículo ¨*Sistema para la Integración del Proceso de Normalización de Bases de Datos Relacionales con Gestores de Bases de Datos (SINORGES)* ¨. (19)

Herramienta de Administración de Bases de Datos HABD

En la actualidad se estudia la forma de introducir el proceso de normalización en los SGBD, elemento que es apoyado por algunos especialistas del tema, pero también es criticado por otro grupo considerado de expertos, pues se afirma que el surgimiento y la función de los SGBD no es la de verificar el modelo de datos, sino gestionar y administrar la representación y almacenamiento de los datos, así como la seguridad, control de acceso de usuario y su lenguaje de acceso a la BD. Esto trajo consigo que se analizará la integración del proceso de normalización ya no desde el propio SGBD PostgreSQL, que es el que se está potenciando en la universidad, sino desde la propia herramienta de administración HABD, que se está desarrollando por parte del equipo de proyecto, con el objetivo que brinde la mayoría de las prestaciones posibles para manipular, administrar y gestionar BD en PostgreSQL, sin tener que acceder o interactuar con otras herramientas adicionales, como las mencionadas anteriormente para fines específicos.

HABD provee una interfaz amigable y una arquitectura basada en plugin, que permite a los desarrolladores incorporar nuevos servicios de manera sencilla por la flexibilidad que posee. De esta forma se le pueden agregar gran número de funcionalidades en las que actualmente se encuentra trabajando el grupo de desarrolladores de la línea PostgreSQL tales como el CRUD-PG, Mask-PG, PgDataMerge, Particionado de BDR, entre otras, evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema. La aplicación contribuye al intercambio entre el usuario y el gestor PostgreSQL y facilita el trabajo de estos. Para más información dirigirse a la tesis de investigación “Exploración y diseño de la herramienta de administración de bases de datos para PostgreSQL (HABD)”.

1.2.1 ¿Cómo integrar la teoría de Normalización a HABD?

Existen diferentes formas de integrar el proceso de normalización a la HABD. Después de un profundo estudio realizado se consideró que como parte de la solución se deberían analizar tres propuestas esenciales: los módulos, las librerías y los plugin informáticos.

Capítulo 1. Fundamentos teóricos

Definición de Módulo Informático

Un módulo es un segmento, rutina, subrutina, sub-algoritmo o procedimiento, que puede definirse dentro de un algoritmo con el fin de ejecutar una tarea específica, y puede ser llamada o invocada desde el algoritmo principal cuando sea necesario. Los módulos son independientes en el sentido de que ningún módulo puede tener acceso directo a cualquier otro módulo, con excepción del módulo al que llama y sus propios sub-módulos. Sin embargo, los resultados producidos por un módulo pueden ser utilizados por cualquier otro módulo cuando se transfiera a ellos el control. (20)

- **¿Qué ventajas proporcionan los módulos?**
  - ✓ Los **módulos son independientes,** pues varios programadores podrán trabajar simultáneamente en la confección de un algoritmo, repartiéndose las distintas partes del mismo. Logrando efectuar con mayor facilidad del desarrollo de un programa.
  - ✓ El uso de **módulos facilita la proyección y la comprensión de la lógica,** subyacente para el programador y el usuario.
  - ✓ **Aumenta la facilidad de depuración y búsqueda de errores** en un programa, ya que estos se pueden aislar fácilmente.

Definición de Librería Informática

Conjunto de módulos objeto (resultados de compilación) agrupados en un solo fichero que suele tener las extensiones **.lib**, **.bpl**, **.a**, **.dll**, etc. Estos ficheros permiten tratar las colecciones de módulos como una sola unidad, y representan una forma muy conveniente para el manejo y desarrollo de aplicaciones grandes. (21)

- **¿Qué ventajas proporcionan las librerías?**
  - ✓ Las librerías son una forma sencilla y versátil de modularizar y reutilizar código.
  - ✓ Contienen el código y los datos que proporcionan servicios a programas independientes.
  - ✓ Facilita la distribución del código y datos.

Definición de Plugin Informático

Los plugin no son parches ni actualizaciones, sino propiedades añadidas a los programas originales, aparecidas por primera vez a mediados de los años 70 y conocidas también como complementos, extensiones y agregado. Un plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

- **¿Qué ventajas proporcionan los plugin?**

Los plugin permiten que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones, permiten reducir el tamaño de la aplicación, así como separar el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software.

El análisis de los puntos de vista anteriormente planteados arroja como propuesta de solución para la integración de la teoría de normalización de BDR a la herramienta HABD el desarrollo de un plugin, dado el hecho que la arquitectura de HABD está basada en plugin y que estos brindan gran estabilidad para cualquier aplicación que se desarrolle. Además después del estudio se valoró que es la técnica más apropiada para brindar un servicio o un grupo de funcionalidades específicas por las características antes descritas.

Ventajas de integrar el plugin de normalización a la HABD:

- Agiliza el trabajo de los usuarios.
- Hace independiente la integración de sus funcionalidades.
- No carga la aplicación a la cual se integra, con esto se garantiza eficiencia a la hora de trabajar.

1.3 Metodología y Lenguajes de Programación

Metodología de Desarrollo

“La metodología de desarrollo de software en Ingeniería de Software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (22)

Existen grandes variedades de propuestas de metodología para desarrollar software. Estas metodologías por lo general se enfocan en el control del proceso, estableciendo reglas rigurosas a la hora de realizar diferentes actividades, utilización de las herramientas y las diferentes notaciones al respecto, dado estas reglas, estas metodologías se identifican por ser rígidas y dirigidas por el número excesivo de documentación que se genera en cada una de las actividades desarrolladas.
Esta dirección no resulta ser la más adecuada para muchos de los proyectos cuyos requisitos cambian a menudo, por lo que en este espacio, las metodologías ágiles emergen como una posible respuesta.

Capítulo 1. Fundamentos teóricos

Existen diferentes metodologías ágiles, una de las más utilizadas es XP (Extreme Programing), por las ventajas que esta proporcionan.

Metodología XP

La metodología XP fue desarrollada por **Kent Beck**. Surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos, no es más que una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. (23)

- XP se plantea como una metodología a emplear en proyectos de riesgo.
- XP aumenta la productividad.

¿Por qué usar XP?

Se elige usar la metodología XP porque es muy adaptable a las necesidades actuales del proyecto, además flexibiliza el ciclo de vida completo en el desarrollo de un software. Al estar enfocado en el trabajo en grupo, es más fácil distribuir responsabilidades entre los integrantes del proyecto, garantizando luego integrar las tareas de forma rápida y eficiente. En comparación con otras metodologías como la RUP, XP es mucho más rápido, ya que conlleva menos protocolo, lo que evita que existan jerarquías dentro del grupo.

Lenguaje de Modelado. Definición de UML

Es un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos. (24)

- Los lenguajes de programación son herramientas que nos permiten crear programas y software. Entre ellos tenemos Delphi, Visual Basic, Pascal, Java, etc.
- Una computadora funciona bajo control de un programa el cual debe estar almacenado en la unidad de memoria; tales como el disco duro.
- Los lenguajes de programación de una computadora en particular se conocen como código de máquinas o lenguaje de máquinas.
- Facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar.
- Representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por una persona. (25)

Capítulo 1. Fundamentos teóricos

Lenguaje de programación C++

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

C++ es uno de los lenguajes de programación más utilizados en la actualidad, esto se debe principalmente a las siguientes características:

Permite la agrupación de instrucciones, presenta un conjunto completo de instrucciones de control, incluye el concepto de puntero[1], los argumentos de las funciones se transfieren por su valor, es un lenguaje estructurado, puede manejar actividades de bajo-nivel, implementación de apuntadores, uso extensivo de apuntadores para la memoria, arreglos, estructuras y funciones, los argumentos de las funciones se transfieren por su valor.

¿Por qué C++?

Después de realizar el estudio de las características fundamentales que potencian la programación y el desarrollo de los plugin, se valoró que la solución propuesta se implementará en C++ por las siguientes razones:

- Las librerías del IDE de desarrollo QT están basadas en este lenguaje.
- Problemas de compatibilidad con la herramienta de administración HABD.

Ventajas del lenguaje C++ en la creación de plugin

En la creación de plugin, el lenguaje C++ es uno de los más utilizados en la actualidad, ya que además de poseer una amplia documentación, la mayoría de los framework que permiten desarrollar plugin, así como muchas aplicaciones informáticas de estos tiempos utilizan este lenguaje, lo cual hace que un plugin implementado en C++ tenga alta compatibilidad con sistemas informáticos y posean gran demanda en el mercado. Además es un lenguaje orientado a objetos muy potente mediante el cual se puede programar a alto y bajo nivel.

[1]Puntero: variable que contiene la dirección de otra variable

1.4 Tecnologías y Herramientas

Framework de desarrollo: QT

Según, Alan D. Osorio Rojas, QT trata de un framework para desarrollar aplicaciones que requieran una interfaz gráfica de usuario. Proporciona una excelente compatibilidad para desarrollar aplicaciones para: Microsoft Windows: 98, NT 4.0, ME, 2000, y XPUnix/X11 Linux, Sun Solaris, HP-UX, HP Tru64 UNIX, IBM AIX, SGI IRIX y muchas otras. (26)

Ventajas de QT

- Multiplataforma.
- Software Libre.
- Tiene el apoyo de Nokia (quien es su dueña).
- Posee cientos de recursos y guías en Fórum Nokia.
- Maneja efectos: transparencias, sombras etc, de forma fácil.
- Ide Drag and Drop. Menos código y resultados rápidos.
- Soporte completo para la plataforma Symbian.

IDE de desarrollo: QT Creator

QT Creator es un completo Entorno de Desarrollo Integrado (IDE) para la creación de aplicaciones con el framework de aplicaciones QT. (27)

El objetivo principal de QT Creator es satisfacer las necesidades de desarrollo de los desarrolladores de QT que buscan la simplicidad, facilidad de uso, la productividad, la extensibilidad y la apertura (27). Entre sus principales características se encuentran: (27)

- Es un IDE multiplataforma que permite a los desarrolladores crear aplicaciones para escritorio y plataformas de dispositivos móviles.
- Se integra con los sistemas de control de la versión más popular, incluyendo Git, Subversion, Bazaar, Perforce, CVS y Mercurial.
- Ofrece soporte para la edición de C + +, ayuda sensible al contexto, completado de código, navegación, etc.
- Genera todos los archivos necesarios si se importa un proyecto existente o crear uno desde cero.

Herramienta CASE (por sus siglas en inglés Computer Aided Software Engineering): Visual Paradigm

Visual Paradigm, es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones con ganancia en calidad y coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (28).

Herramienta de control de versiones: Subversion

- Es una herramienta de software libre para el control de versiones.
- Subversion es capaz de manejar los cambios hechos sobre ficheros y directorios a lo largo de tiempo.
- Permite recuperar versiones antiguas y ver el historial de cambios de un sistema de archivos.
- Es una herramienta distribuida capaz de trabajar sobre diversos protocolos (svn, ssh, http, https).
- No es una herramienta de gestión de la configuración software, principalmente porque no está diseñada para manejar exclusivamente software, sino archivos arbitrarios.
- Diseñado para sustituir CVS (Concurrent Version System). (29)

CONCLUSIONES

Del análisis realizado a lo largo de este capítulo, esencialmente basándose en la validación de los datos y la necesidad de la integración del proceso de normalización de BDR a la HABD se concluye:

- El estudio del proceso de normalización de BDR revela la importancia de la integración de esta teoría en la HABD, minimizando así, la existencia de anomalías y garantizando una elevada calidad, robustez y extensibilidad de los modelos de datos en el SGBD PostgreSQL.

- La integración de la teoría de normalización de BDR a la HABD a través de un plugin garantiza un alto nivel de compatibilidad y estabilidad en la misma.

- El frameworkQT resultó ser la propuesta para llevar a cabo la implementación del plugin, haciendo uso de QT Creator como IDE de desarrollo, debido a la potencialidad que posee en la creación de estos, y el hecho de brindar una programación orientada a objetos, utilizando C++ como lenguaje de programación, condiciones bajo las cuales se desarrolla HABD.

- XP como metodología de desarrollo, garantiza la agilidad y adaptabilidad necesaria en el desarrollo de la solución que se propone.

CAPÍTULO 2.CARACTERÍSTICAS DEL SISTEMA PROPUESTO

INTRODUCCIÓN

El Segundo Capítulo “Descripción de la propuesta. Análisis y Diseño”, expone los diferentes elementos que fundamentan el desarrollo de la propuesta, para darle solución al problema de la investigación planteado anteriormente. Donde se describe la herramienta que se propone a través de las historias de usuario y los requisitos no funcionales. De esta forma se generan los artefactos que propone la metodología XP en correspondencia con las etapas de Planificación y Diseño, así como una breve descripción de los patrones de diseño que se utilizarán en el transcurso del diseño del ciclo de vida del sistema.

2.1 IDENTIFICACIÓN DEL PROBLEMA

Para darle solución al problema de la investigación identificado se propone el desarrollo de un Plugin que permita gestionar el proceso de normalización de las BDR, permitiendo normalizar desde la 2da FN hasta la FNBC y que sea capaz de integrarse a HABD.
Para garantizar calidad en la construcción del plugin de normalización de BDR, fueron analizadas algunas herramientas existentes en la actualidad que permiten normalizar BDR, lo cual arrojó como resultado que la más ajustable a las necesidades actuales es la herramienta SINORGES(20), herramienta creada por Msc. Yanet Espinal Martín que define un profundo proceso de normalización de BDR, pues permite normalizar hasta la FNBC, característica por la cual fue tomada dicha aplicación como punto de referencia para la construcción del plugin de normalización de BDR, ya que ninguna otra propuesta de aplicación profundiza tanto en el proceso de normalización.

2.2 MODELO DE DOMINIO

El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto, es decir, un diagrama con los objetos que existen (reales) relacionados con el proyecto que vamos a acometer y las relaciones que hay entre ellos. Se dice que es estática porque no representa la interacción en el tiempo de los objetos, sino que representa una visión "parada" de las clases y sus interacciones. (30)

El Modelo de Dominio ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar el plugin (30).Véase Ilustración 7

Capítulo 2.Características del sistema propuesto

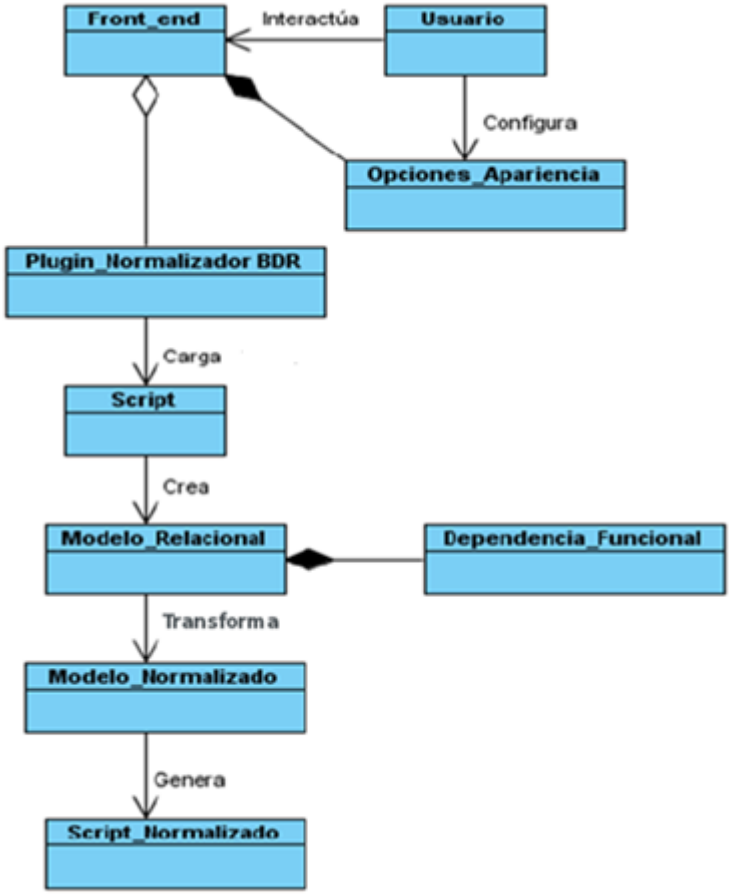


ILUSTRACIÓN 6: MODELO DE DOMINIO.

Aunque la metodología XP no utiliza el modelo de dominio este se decidió confeccionar, para un mejor entendimiento del problema a resolver, ya que no hay un trabajador de negocio, existe solapamiento de roles y el proceso de negocio no está bien definido.

- ✓ **Usuario:** Esta clase es la encarda de utilizar el front_end y ejecutar todas sus operaciones.
- ✓ **Front-End:** Esta clase es la encargada de administrar los plugin y es la cara para el usuario.
- ✓ **Opciones-Apariencia:** Esta es la clase que permite que el usuario modifique la apariencia del front_end.
- ✓ **Plugin normalizador BDR:** Esta clase permite al front_end identificar el plugin normalizador NBR.
- ✓ **Script:** Esta clase garantiza la construcción del modelo relacional y de controlar todo lo referente al script, verificando que sea el script escogido por el usuario y que dicho script sea de PostgreSQL.

- ✓ **Modelo-Relacional:** La clase modelo relacional contiene las relaciones y dependencias contenidas en el script necesarias para el proceso de Normalización.
- ✓ **Dependencias-Funcionales:** Es la clase que contiene las dependencias funcionales para cada relación en el modelo relacional.
- ✓ **Modelo-Normalizado:** Esta clase es el resultado del proceso de Normalización aplicado al modelo relacional una vez insertadas las dependencias.
- ✓ **Script-Normalizado:** Esta clase es el resultado del modelo normalizado llevado a un script para PostrgreSQL.

2.3 Descripción del Sistema Propuesto

El plugin permitirá cargar un script de postgreSQL, a partir del cual se llena el modelo relacional perteneciente a dicho script con las relaciones, atributos y dependencias funcionales contenidas en el mismo. Una vez obtenido el modelo relacional el usuario, a partir de las reglas definidas en el negocio, podrá insertar las dependencias funcionales necesarias. Después de realizado el procedimiento se podrá determinar en qué forma normal se encuentra el modelo o la relación seleccionada, puesto a que la inserción de una dependencia funcional puede afectar considerablemente el modelo relacional. Luego el usuario podrá escoger en que forma normal quiere que su modelo sea normalizado, para este caso el sistema brinda las opciones de normalizar a 2da FN, 3ra FN o FNBC. Una vez realizado el proceso de normalización, se muestra el estado del modelo ya normalizado y se permite realizar cualquier operación anteriormente mencionada o para finalizar, el usuario podrá exportar el script a un directorio determinado, de ser necesario esta funcionalidad cuenta con la opción de poder visualizar el interior del script.
El producto final de este trabajo es un plugin de normalización de BDR que sea capaz de integrarse a HABD, por lo que los niveles de privilegio para acceder a sus funcionalidades son en correspondencia con el tipo de usuario que trabaje sobre HABD.

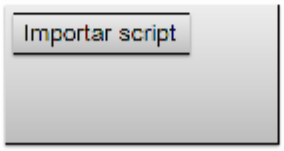
2.4 Historia de Usuario

El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las “Historias de usuarios” (HU) sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el

momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios (31).

Durante el diseño se identificaron un total de ocho Historias de Usuario que serán realizadas en dos iteraciones, aquí se muestra la HU ¨*Importar Script*¨, el resto de las HU se encuentran descritas en los anexos.

TABLA 1: HISTORIA DE USUARIO ¨IMPORTAR SCRIPT¨

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Importar Script.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Grether Méndez Gómez	**Iteración asignada:**1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 1 semana
Descripción: El usuario tendrá la posibilidad de abrir el directorio donde está guardado el Script, para poder cargarlo en la aplicación.	
Observaciones: El plugin solo cargará los Scripts que cumplan con lo definido en la aplicación.	
Prototipos de interfaces: 	

2.5 Lista de Reserva del Producto

La lista de Reserva del Producto es una tabla que contiene los requisitos funcionales que debe cumplir la aplicación, ordenados por la prioridad de implementación, así como los requisitos no funcionales del sistema a desarrollar. Además en esta tabla se refleja la estimación de cada uno de ellos y su implementación por semanas, definiendo de esta forma el rol que hizo la estimación.

TABLA 2: LISTA DE RESERVA DEL PRODUCTO.

Ítem	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Importar script.	1 semana	Analista
2	Insertar dependencias funcionales.	2 semanas	Analista
3	Normalizar a 2da FN.	3 semanas	Analista
4	Normalizar a 3ra FN.	3 semanas	Analista
5	Normalizar a FNBC.	3 semanas	Analista
Prioridad: Alta			
6	Determinar forma normal	1 semana	Analista
Prioridad: Media			
7	Exportar script normalizado.	1 semana	Analista
8	Generar ayuda.	1 semana	Analista
Requisitos no funcionales			
1	**Requisitos de Rendimiento:** El tiempo de respuesta dependerá del tamaño de las tablas.		Analista
2	**Requisitos de Portabilidad:** Será un sistema multiplataforma, lo que permitirá poder disponer del mismo en cualquier sistema operativo.		Analista
3	**Requisitos de Seguridad:** **Confidencialidad:** Solo tendrá acceso a la visualización de los datos sensibles el usuario que posea los privilegios necesarios para esto.		Analista
4	**Requisitos de Integridad:** El plugin no permitirá hacer cambios en las tablas a menos que el usuario lo decida.		Analista
5	**Requisitos de Disponibilidad:** Se podrá hacer uso de la aplicación siempre que esté instalada e integrada a HABD.		Analista
6	**Restricciones del diseño y la implementación:** Para el diseño e implementación del plugin se utiliza la metodología XP, haciendo uso del Visual Paradigm 6.4 para el modelado. También se utilizará como lenguaje de		Analista

	programación C++ y como IDE de desarrollo QT Creator en su versión 4.7.		
7	**Requisitos de Apariencia o interfaz externa:** Interfaz intuitiva, amigable, organizada, con una navegabilidad flexible y de fácil comprensión, de forma que el objetivo del sistema para con los clientes, se pueda conseguir rápidamente. Debe estar diseñada para verse en cualquier resolución igual o inferior a 1024x768. El diseño gráfico debe estar acorde con las pautas de diseño de la Universidad.		Analista
8	**Requisitos de Facilidad de uso:** Para utilizar el sistema es necesario poseer conocimientos elementales de computación, así como de base de datos.		Analista
9	**Requisitos de Soporte:** Se dispondrá de documentación técnica que describa todas las funcionalidades del sistema, de modo que existirá un manual para el uso de la aplicación orientada a clientes y usuarios finales. Sistema Multiplataforma con licencia GPL/, PostgreSQL 8.4.x o versiones.		Analista
10	**Requisitos de Software:** Sistema Operativo: Multiplataforma. Librerías QT. Servidor de bases de datos: SGBD PostgreSQL 8.4.x o versiones superiores.		Analista
11	**Requisitos de Hardware:** Se necesita 100 MB de memoria RAM mínimo, 100 MB de espacio libre en el disco duro para su instalación y el micro a 300 MHz.		Analista
12	**Requisitos de Seguridad:** Garantizar que cada conexión que se realice al servidor de bases de datos solicite autenticación. Permitir que las funcionalidades del sistema se muestren de acuerdo al tipo de usuario que esté activo. Permitir hacer salvas y restauración cuando se caiga la conexión con el servidor.		Analista

2.6 Tareas de ingeniería

Se evalúa cada escenario, entiéndase por escenario historias de usuarios o requisitos y lo divide en tareas donde cada una de estas representa una característica del sistema. A continuación se presentará la tarea de ingeniería ¨*Importar script*¨. El resto de las tareas se encuentran descritas en los anexos.

TABLA 3: TAREA DE INGENIERÍA ¨IMPORTAR SCRIPT¨.

Tarea de Ingeniería	
NúmeroTarea: 1	**Número Historia de Usuario:** 1
NombreTarea: Importar Script.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
FechaInicio:3/2/12	**Fecha Fin:**9/2/12
Programador Responsable: Osmil Guillen Tamayo	
Descripción: El sistema brindará la opción de seleccionar el Script en el directorio donde se encuentra guardado, para poder cargarlo en la aplicación.	

2.7 Plan de Iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. De esta misma forma, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir. (31)

Una vez definidas las HU, es necesario crear un plan de iteraciones con el objetivo de definir cuáles son las tareas con más prioridad y establecer los tiempos de implementación.
Las HU quedarán definidas de la siguiente forma:

- Se determinó que las HU Importar script (1), Insertar dependencia funcional (2), Normalizar 2da FN (4), Normalizar 3ra FN (5) y Normalizar FNBC (6) estén en la primera iteración, debido a su nivel de complejidad y por su importancia en el proyecto. Estas son las encargadas de cargar, crear y normalizar el script.
- Se determinó que las HU Determinar forma normal (3), Exportar script (7) y Generar ayuda (8) estén en la segunda iteración. Dado que estas son las responsables de determinar, generar y guardar el script.

2.8 Modelo de Diseño

La realización del modelo de diseño es una parte central de todas las actividades que conducen a la producción de buen software, es garantizar satisfacción a las necesidades de los usuarios y ayuda a comprender mejor el sistema que se desarrolla.

2.8.1 Diagrama de Clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Se dice que los diagramas de clases son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases «conocen» a qué otras clases o qué clases «son parte» de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas (32). Véase Ilustración 8

Capítulo 2.Características del sistema propuesto

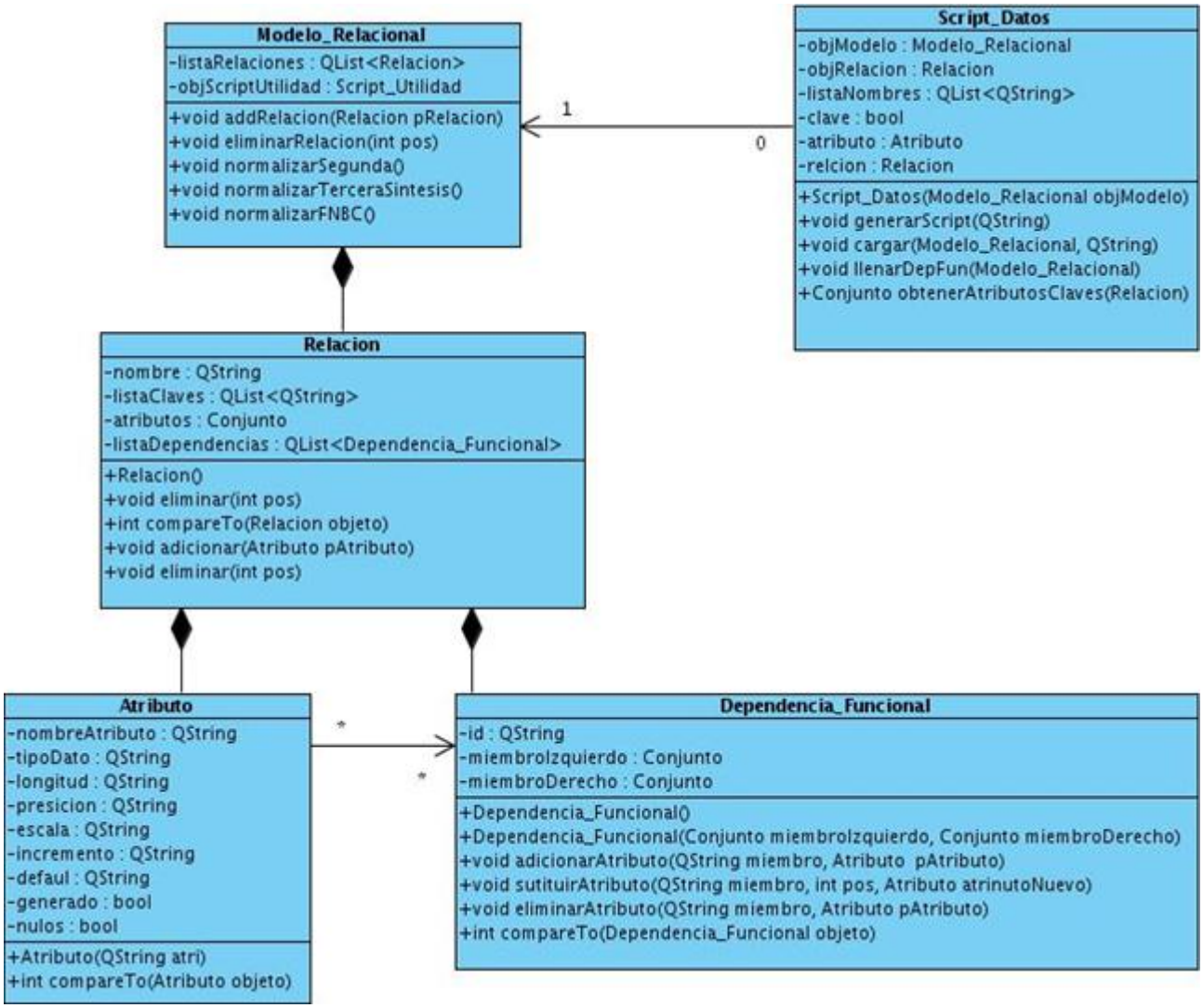


ILUSTRACIÓN 7: DIAGRAMA DE CLASES.

- ✓ **Modelo_Relacional:** Controla el proceso de normalización.
- ✓ **Relación:** Contiene las funcionalidades inmersas en el proceso de normalización.
- ✓ **Script_Datos:** Se encarga de crear el modelo relacional y generar el script de postgreSQL.
- ✓ **Atributo:** Define los atributos en las relaciones.
- ✓ **Dependencia_Funcional:** Define las dependencias funcionales en las relaciones.

2.8.2 TARJETAS CRC

Para el diseño de aplicaciones informáticas XP no se requiere la presentación del sistema mediante diagramas utilizando UML como lenguaje de modelado. En su lugar se utilizan otras técnicas como las tarjetas CRC (Clase, Responsabilidad, Colaboración). (31)

Estructura de las tarjetas CRC

Las tarjetas CRC están divididas en tres partes:

1. Una **clase** es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realiza, sus atributos y métodos.
2. Los **colaboradores** de una clase son las demás clases con las que trabaja en conjunto.
3. Llevar a cabo sus **responsabilidades**.

El formato físico de las tarjetas CRC facilita la interacción entre clientes y equipo de desarrollo, en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de especificación de requisitos o historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones. Luego en un estado de diseño avanzado o ya en la implementación del sistema, las tarjetas CRC se convierten en clases con métodos, atributos, relaciones de herencia, composición o dependencia. (31)

A continuación se mostrará la tarjeta CRC de la clase Dependencia_Funcional. El resto de las tarjetas se encuentran descritas en los anexos.

TABLA 4: TARJETA CRC ¨DEPENDENCIA FUNCIONAL¨.

Tarjeta CRC	
Clase: Script.	
Responsabilidades	**Colaboraciones**
-Importar Script. -Exportar Script.	Dependencia. Normalizar.

Esta tarjeta como todas las demás desarrolladas se obtuvo del agrupamiento de las HU. En este caso las HU agrupadas fueron importar script y exportar script, que dieron lugar a la clase script.

2.9 Patrones de Arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático. (33)
Estas Arquitecturas están definidas muchas veces por el tipo de tecnología a la cual se enfrenta un programador o grupo de programadores, por lo cual algunos tipos de arquitectura son más recomendables que otras para ciertas tecnologías. (33)

La arquitectura de software a utilizar será modelo-vista-controlador (MVC), Véase en la Ilustración 9 (34):

1. **El modelo es el responsable de:** Definir las reglas del negocio, es decir la funcionalidad del sistema, así como mantiene un registro de las vistas y controladores del sistema.

2. **El controlador es responsable de:** Recibir los eventos de entrada y ejecuta la lógica del modelo y las vistas.

3. **Las vistas son responsables de:** Recibir datos del modelo y lo muestra al usuario, es decir, representa la parte que será visualizada.

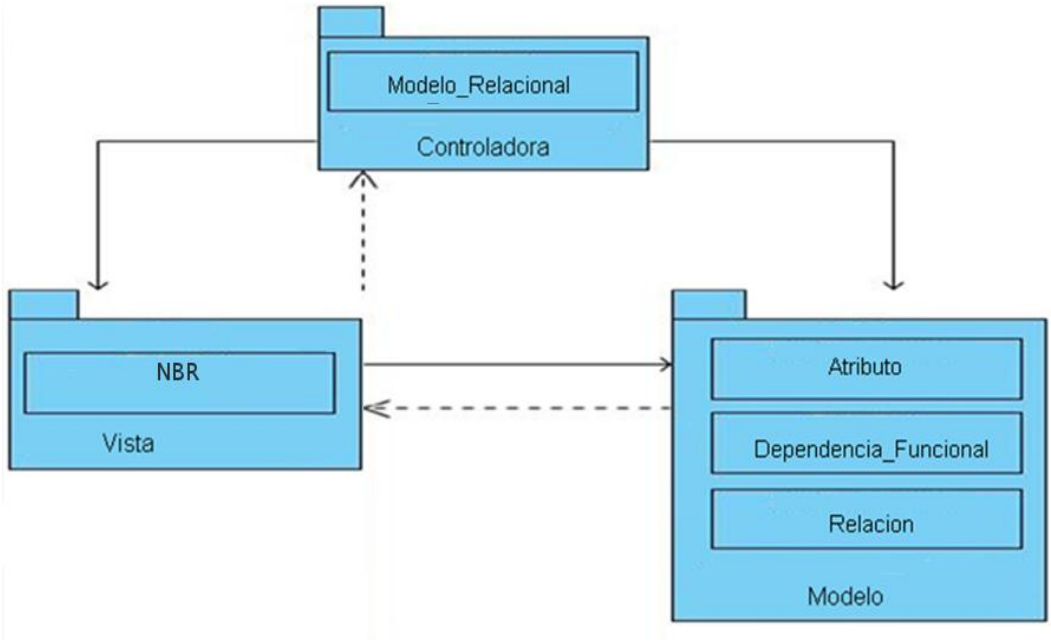


ILUSTRACIÓN 8: ARQUITECTURA DE SOFTWARE MODELO-VISTA-CONTROLADOR.

En el diagrama se muestran la arquitectura del plugin en tres paquetes. El paquete vista contiene la clase NBR, encargada de servir de interfaz visual al usuario, en el paquete modelo se encuentra representado el negocio partir de las clases Atributo, Dependencia_Funcional y Relación, y en el paquete controlador se encuentra la clase Modelo_Relacional, quien se encarga de controlar el negocio.

2.10 Patrones de Diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación (35).
Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso. (35)

Los patrones GRASP (Patrones de Software para la asignación General de Responsabilidad) constituyen un apoyo para la enseñanza, que ayuda a entender el diseño de objetos. De los diferentes patrones que ofrece GRASP se emplearon para modelar el plugin los siguientes:

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Un elemento con bajo (o débil) acoplamiento no depende demasiado de otros elementos (36).

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable (36).

El uso de los patrones Bajo Acoplamiento y Alta Cohesión se evidencia en la clase Script_Datos, ya que sus funcionalidades son independientes de otras clases y se identifica por ellas. Véase Ilustración 9.

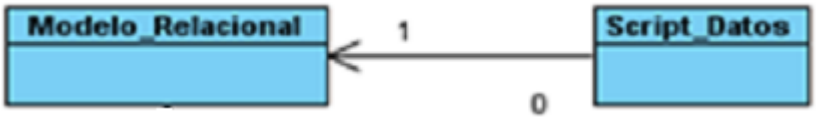


ILUSTRACIÓN 9: PATRÓN DE DISEÑO ¨BAJO ACOPLAMIENTO.¨

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión (36).

El uso del patrón Controlador se evidencia en la clase Modelo_Relacional, ya que sus funcionalidades están enfocadas en controlar el negocio. Véase Ilustración 10.

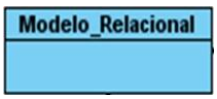


ILUSTRACIÓN 10: PATRÓN DE DISEÑO ¨CONTROLADOR¨.

Patrón Experto: Lo que plantea este patrón es asignar una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para cumplir con la responsabilidad. El problema que resuelve el patrón experto está referido al principio más básico mediante el cual las responsabilidades son asignadas en el diseño orientado a objetos (36).

El uso del patrón Experto se evidencia en las clases Modelo_Relacional y Relacion, donde fue preciso asignarle a cada clase la responsabilidad que le corresponde. Véase Ilustración 11.

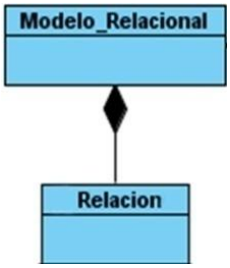


ILUSTRACIÓN 11: PATRÓN DE DISEÑO ¨EXPERTO¨.

Patrón Creador: El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias creadas del objeto. (36)

El uso del patrón Creador se evidencia en las clases Modelo_Relacional y Relacion, al crear instancias de la clase Relacion en la clase Modelo_Relacional para realizar correctamente sus responsabilidades. Véase Ilustración 12.

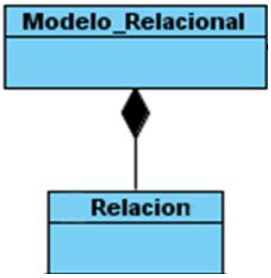


ILUSTRACIÓN 12: PATRÓN DE DISEÑO ¨CREADOR¨.

2.11 Estándares de Codificación

Los estándares de codificación son aquellos que permiten entender de manera rápida, fácil y sencilla el código empleado en el desarrollo de un software. Además, garantizan un mantenimiento óptimo de dicho código por parte del programador y mejoran la legibilidad permitiendo que otras personas puedan colaborar, ya que la redacción del código no les resulta incómoda. A continuación se muestra el estándar de codificación que se llevó a cabo en la implementación del plugin de normalización de BDR.

- **Comentarios**

A cada clase, método y variables se les dejó comentadas para su mejor entendimiento.

Ejemplo: Un pequeño ejemplo del formato de los comentarios.
// Devuelve la UNION de dos conjuntos.

- **Declaración de variables**

Las variables son declaradas en una sola línea, estarán comentadas y ordenadas alfabéticamente. El nombre de la variable comenzará con letras minúsculas y la palabra que le sigue con mayúscula la primera letra.

Ejemplo: Variables simples y compuestas.

```
cantClaves
```

- **Declaración de funciones**

No existe espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Habrá un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo estarán en la línea siguiente. La llave que cierra estará alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

- **Nombre de las clases**

Las clases comenzarán con mayúscula, si la clase es compuesta empezarán con mayúsculas las dos letras iniciales de cada palabra y estarán separadas por un guion bajo.

Ejemplo: Nombre clase simple.

```
class Relacion
```

Ejemplo: Nombre clase compuesta.

```
class Modelo_Relacional
```

- **Nombres sugerentes**

Las clases, atributos y métodos contendrán nombres que permitan asociarlo con la actividad que se está realizando.

Ejemplo: Nombre del método normalizar a segunda.

```
void Modelo_Relacional::normalizarSegunda()
```

2.12 Interfaces de la Aplicación

La aplicación estará formada por varias interfaces que serán las que le darán el cuerpo al plugin, permitiendo de esta forma que el usuario pueda interactuar con el mismo.

A continuación se mostrarán las interfaces principales del plugin de normalización de BDR para HABD.

Ejemplo de interfaz *¨Principal¨:* Esta interfaz es la cara principal del plugin. Véase Ilustración 13.

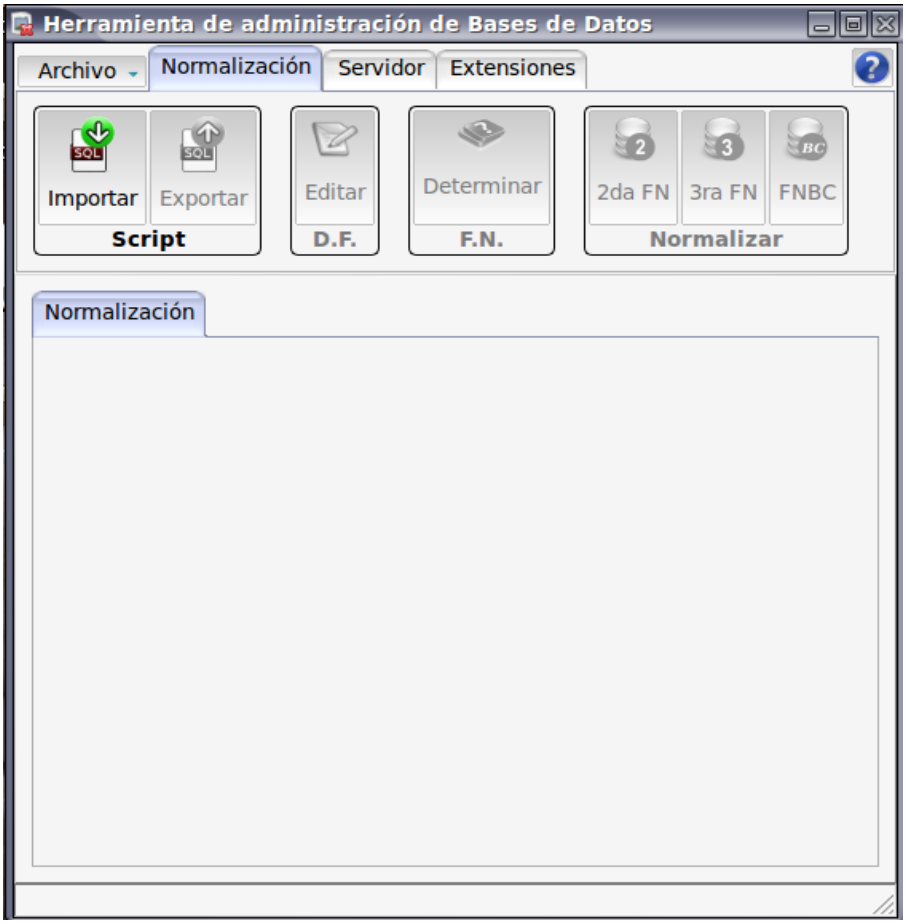


ILUSTRACIÓN 13: INTERFAZ ¨PRINCIPAL¨.

Ejemplo de interfaz ¨*Importar el script*¨: Esta es la interfaz que permite que el usuario pueda cargar el script para realizar las posteriores tareas. Véase Ilustración 14.

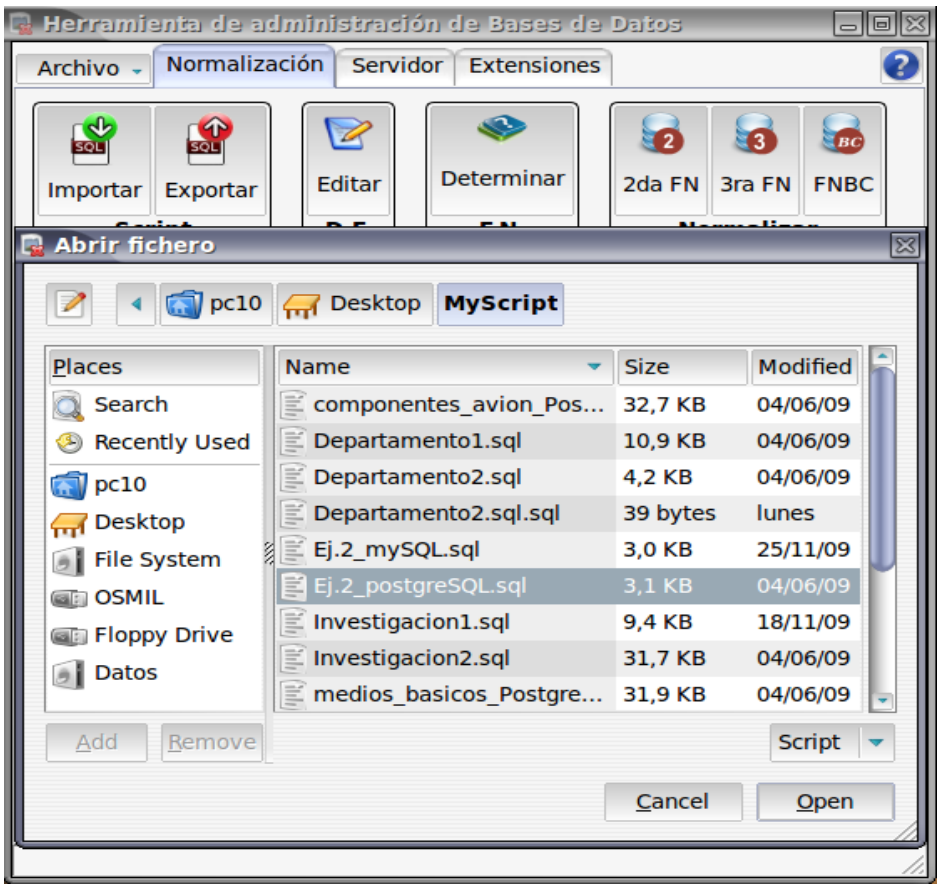


ILUSTRACIÓN 14: INTERFAZ ¨IMPORTAR SCRIPT¨.

Ejemplo de interfaz ¨Insertar atributo¨: Esta interfaz permite al usuario entrarlos atributos que el usuario desee para luego crear las dependencias funcionales. Véase Ilustración 15.

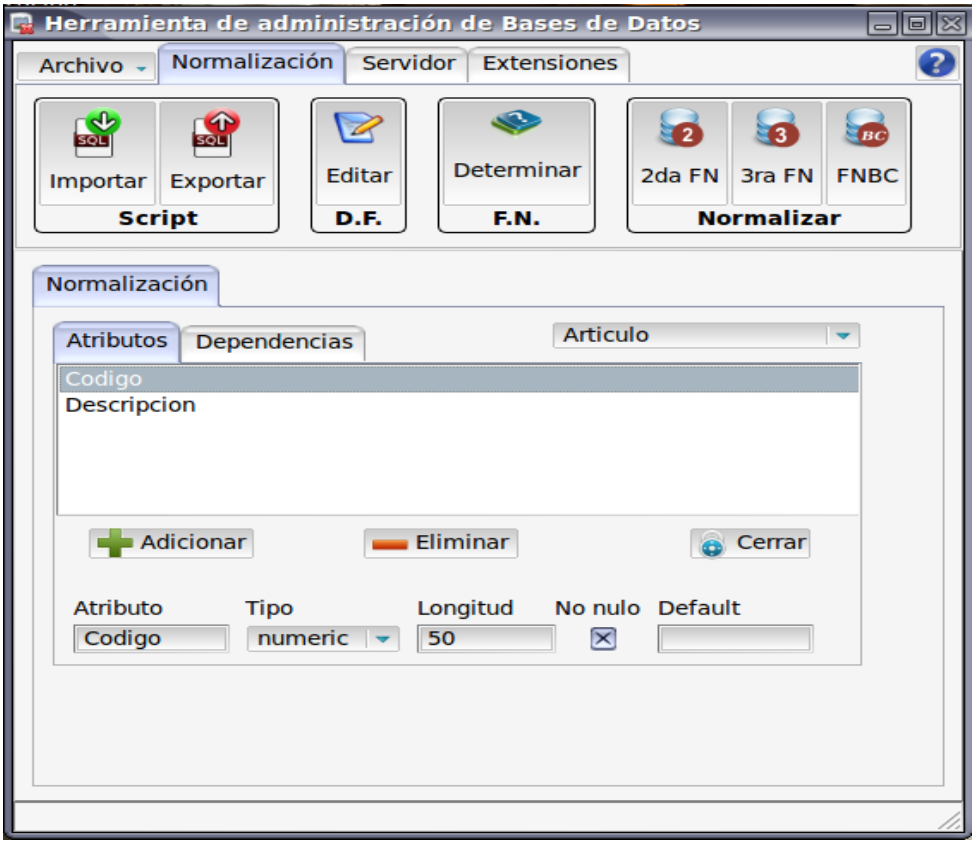


ILUSTRACIÓN 15: INTERFAZ ¨INSERTAR ATRIBUTO¨.

Ejemplo de interfaz ¨Insertar dependencias¨: Esta interfaz permite al usuario entrar las dependencias funcionales y a partir de ellas formar el modelo relacional. Véase Ilustración 16.

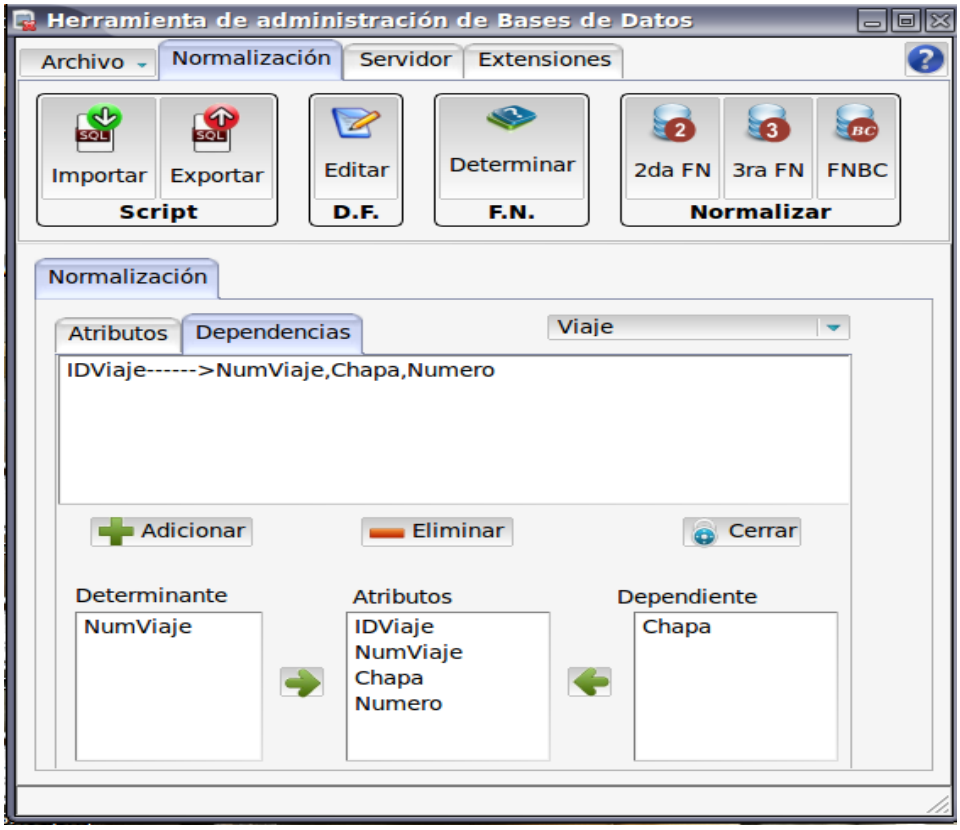


ILUSTRACIÓN 16: INTERFAZ ¨INSERTAR DEPENDENCIA¨.

Ejemplos de interfaz ¨Forma normal¨: En esta interfaz el usuario tendrá la posibilidad de saber en qué forma normal se encuentra el modelo y la relación. Véase Ilustración 17.

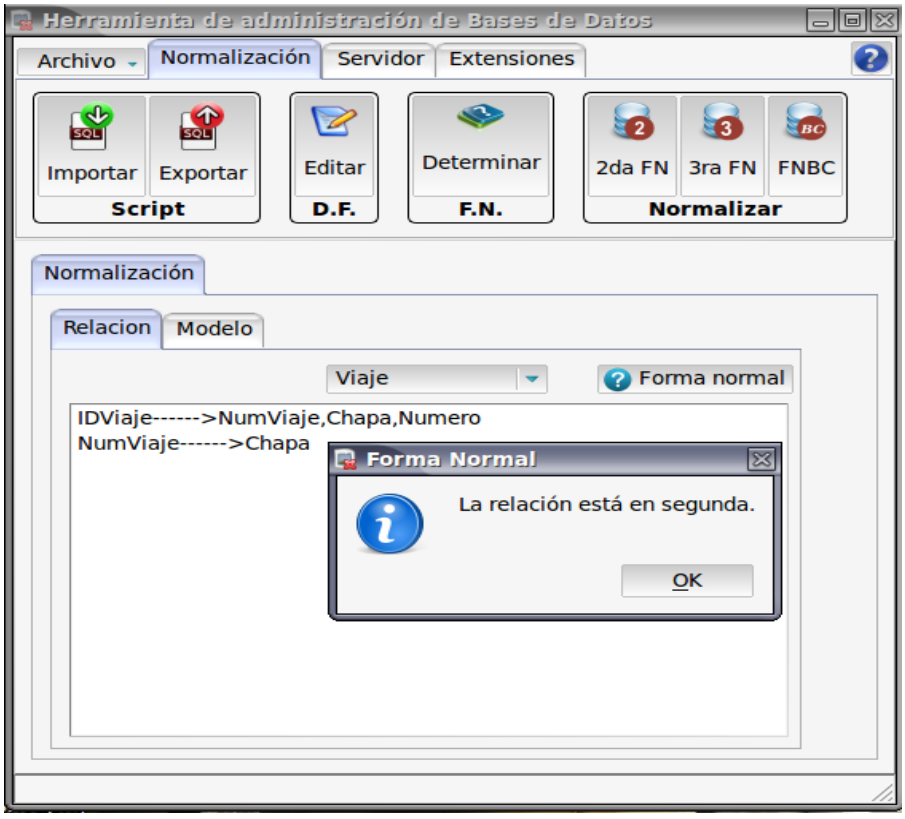


ILUSTRACIÓN 17: INTERFAZ ¨FORMA NORMAL¨.

Ejemplos de interfaz ¨Normalizar a 3ra FN¨: En esta interfaz el usuario tiene la oportunidad de normalizar a la FN que este desee, el ejemplo que se muestra a continuación es normalizar a 3ra FN. Véase Ilustración 18.

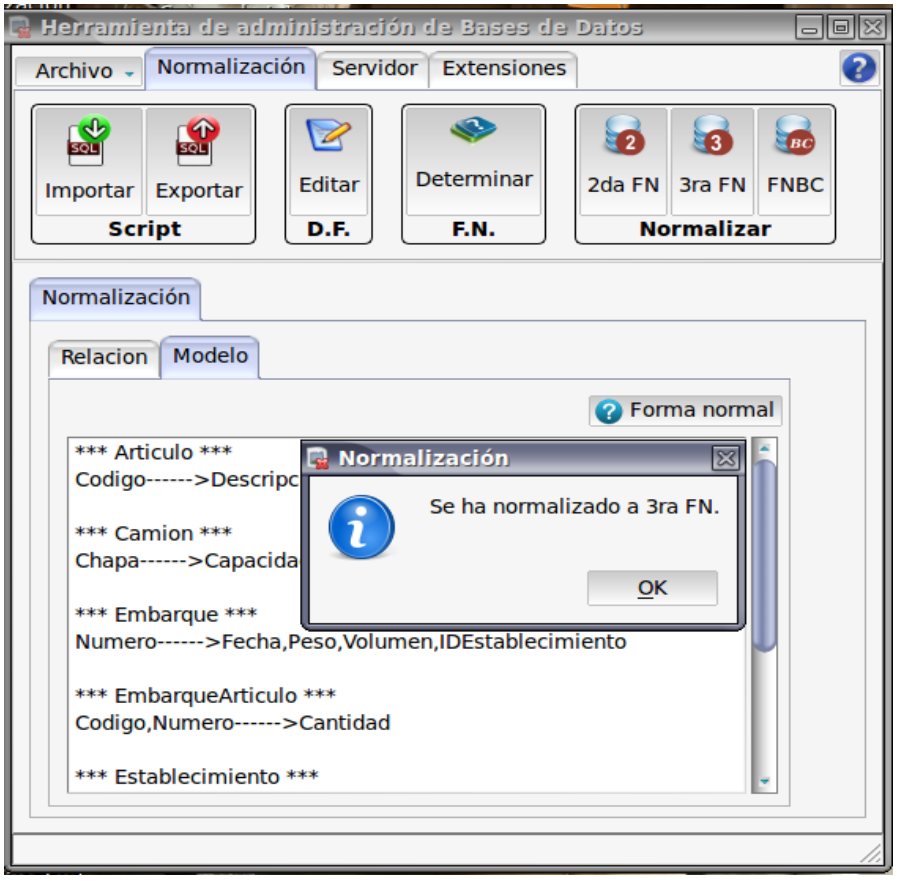


ILUSTRACIÓN 18: INTERFAZ ¨NORMALIZAR 3RA FN¨.

Ejemplos de interfaz ¨Exportar script¨: Esta interfaz permite al usuario exportar el script normalizado en el directorio que este desee. Véase Ilustración 19.

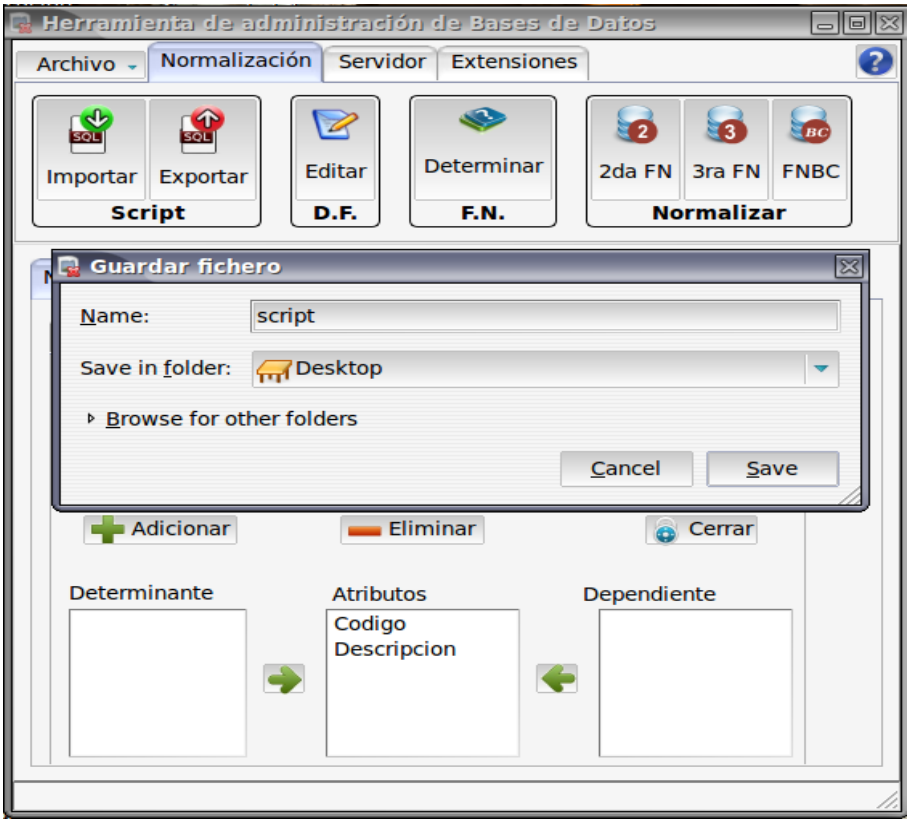


ILUSTRACIÓN 19: INTERFAZ ¨EXPORTAR SCRIPT¨.

Ejemplos de interfaz ¨Visualizar script¨: Luego de haberse exportado el script, el sistema permite que el usuario si lo desea visualice dicho script. Véase Ilustración 20.

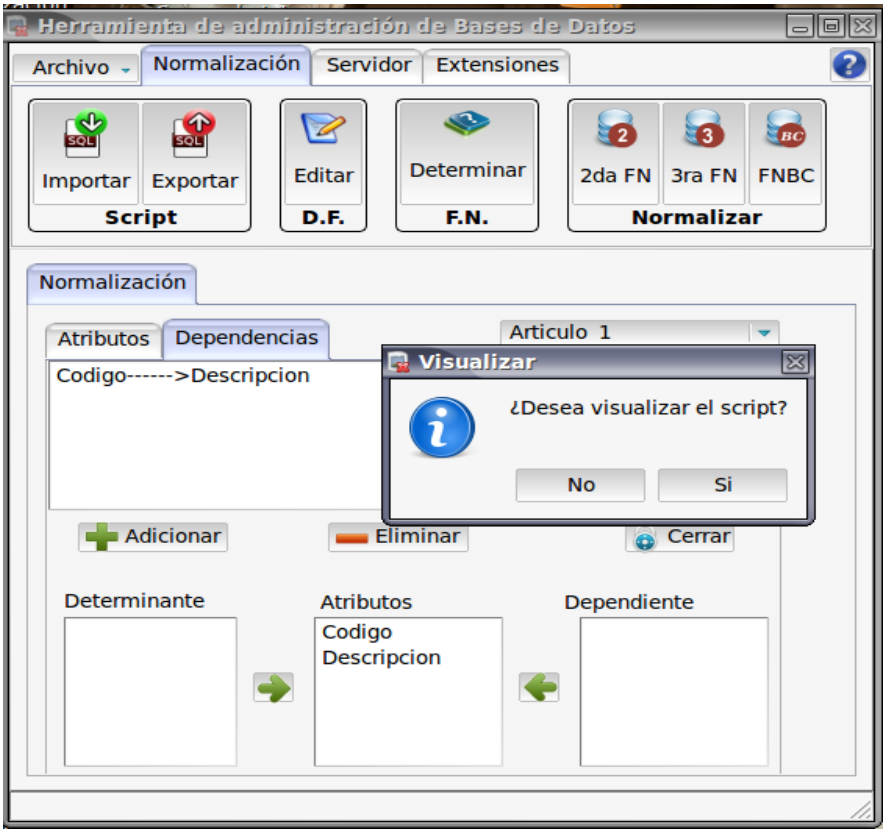


ILUSTRACIÓN 20: INTERFAZ ¨ACEPTAR VISUALIZAR SCRIPT¨.

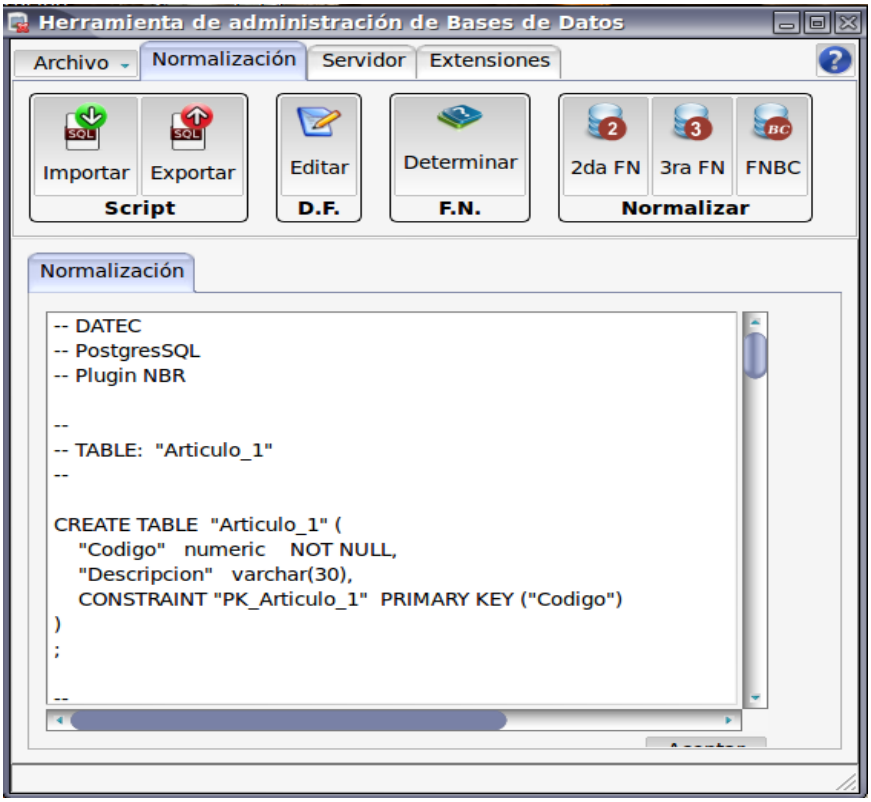


ILUSTRACIÓN 21: INTERFAZ ¨VISUALIZAR SCRIPT¨.

CONCLUSIONES

En el capítulo han sido analizados los procesos fundamentales que están relacionados con la solución informática que se debe desarrollar. Concluyendo lo siguiente:

- Las HU 1, 2, 4, 5 y 6 de complejidad muy alta, y correspondientes a la primera iteración proponen las funcionalidades esenciales para llevar a cabo el proceso de normalización, mientras que la HU 3 de complejidad alta y las restantes 7 y 8 de complejidad media, correspondientes a la segunda iteración, se encargan del resto de las funcionalidades en el plugin.
- Los patrones de diseño aplicados al desarrollo del plugin de normalización, garantizan la organización y el buen funcionamiento del mismo.
- Se definieron y reflejaron en el informe los estándares de codificación empleados durante la implementación de la aplicación; imprescindibles para el entendimiento entre los miembros del equipo de desarrollo.

CAPÍTULO 3. VALIDACIÓN DEL SISTEMA PROPUESTO

INTRODUCCIÓN

El siguiente capítulo presenta las técnicas de validación y pruebas al sistema, para garantizar que las tareas realizadas en el diseño y los requisitos cumplen con las normativas planteadas por los clientes y se encarga de verificar que el producto funcione como se diseñó y que los requisitos se cumplan al máximo.

3.1 ENFOQUES DE DISEÑOS DE PRUEBAS

El proceso de prueba es uno de los pilares fundamentales de la metodología XP, este le proporciona la posibilidad al cliente de verificar y concretar las funcionalidades de las HU, favorece la comunicación entre el cliente y el equipo de desarrollo. Esta filosofía ayuda a identificar y corregir fallos u omisiones cometidas en las mismas, se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección. Permite identificar HU adicionales que no fueron obvias para el cliente o en las que este no hubiese pensado de no enfrentarse a dicha situación. Todo esto contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones. (37)

Existen tres enfoques principales para el diseño de casos (38):

- **Enfoque estructural o de caja blanca:** consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba.
- **Enfoque funcional o de caja negra:** consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos.
- **Enfoque aleatorio:** consiste en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba.

XP como metodología ágil y enfocada principalmente en las necesidades del cliente, propone para el correcto funcionamiento del producto que se diseñe, la realización de dos pruebas: La primera prueba ¨Aceptación¨ y como segunda prueba ¨Las unitarias¨.

Pruebas de aceptación

En XP las necesidades de los usuarios se representan mediante “historias de usuario” que representan los requisitos del sistema. Cada historia de usuario lleva asociada criterios de aceptación y para cada criterio de aceptación se definen casos de prueba. Las pruebas de aceptación se escriben en las fases tempranas del desarrollo, y antes de que el sistema se implemente, para validar las necesidades de los usuarios y para dirigir la implementación. Se puede considerar que las historias de usuario y, por extensión, las pruebas asociadas juegan el papel de especificaciones del sistema.

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario.

Pruebas unitarias

Las pruebas unitarias son pruebas de caja blanca, definidas por el programador, estas pruebas se le hacen a pequeñas porciones de código, por separados, para verificar su correcta funcionalidad, las mismas se pueden ir efectuando desde que se está implementando, no necesariamente se tiene que esperar al final de la implementación del producto.

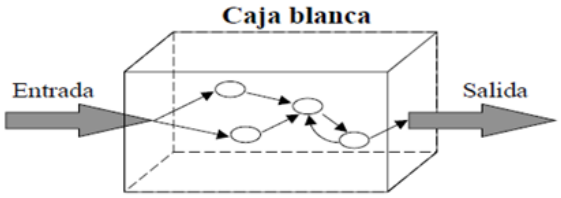


ILUSTRACIÓN 22: MÉTODO DE CAJA BLANCA.

3.1.1 MÉTODO SELECCIONADO

Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requisitos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y

atención. Las pruebas de aceptación permiten al cliente saber cuándo el sistema funciona, y que los programadores conozcan que es lo que resta por hacer. (37)

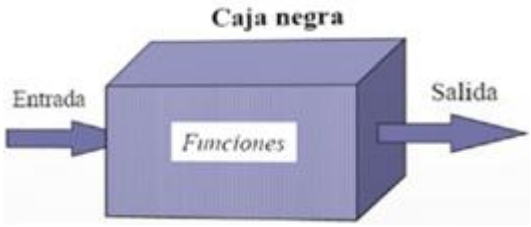


ILUSTRACIÓN 23: MÉTODO CAJA NEGRA.

¿Por qué, usar pruebas de aceptación?

Como anteriormente se había explicado, la metodología XP se enfoca en satisfacer las necesidades del cliente, es un tipo de prueba que cumple con lo planteado. Además estas pruebas las hace el cliente para verificar si su producto está al 100% correcto. Dentro de las pruebas de aceptación se encuentran diferentes tipos de pruebas, pero la escogida es la prueba de funcionalidad, basada en las historias de usuario que anteriormente el cliente había identificado. Por tales motivos se decide usar las pruebas de aceptación, basadas en los casos de prueba.

3.2 Casos de Prueba

Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como, que la integridad de la información externa se mantiene. Este tipo de prueba permite obtener un conjunto de condiciones de entrada que ejerciten de forma completa todos los requisitos funcionales de un programa, es un enfoque complementario que intenta descubrir diferentes tipos de errores, por ejemplo: errores de interfaz y errores en estructuras de datos.

Casos de prueba basados en HU

A continuación se describe el caso de prueba realizado a la HU 1 ¨*Importar script*¨:

TABLA 5: CASO DE PRUEBA ¨IMPORTAR SCRIPT¨.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Importar script correcto.	El usuario carga un script, con los datos correctos.	N/A	Cuando se carga el script, se llena el modelo relacional y se activan las restantes pestañas del plugin.	1. El usuario debe accionar el botón **Importar** este le brindará la posibilidad de buscar el script deseado. Al dar clic en el botón **Aceptar** se llena el modelo relacional.
EC 1.2 Importar script incorrecto.	El usuario carga un script, con los datos no correctos.	I (Script con error interno).	El sistema muestra el mensaje ¨Script incompatible.¨	

TABLA 6: DESCRIPCIÓN DE LAS VARIABLES DETECTADAS EN ¨IMPORTAR SCRIPT¨.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
variable1	Importar	Campo de selección (QFileDialog).	No	Este campo brindará la opción de escoger el script deseado.

Con el objetivo de probar las funcionalidades de la aplicación se diseñaron ocho casos de pruebas de aceptación (ver Anexos), los cuales arrojaron como resultado tres no conformidades; encontradas en la primera iteración. Las mismas fueron solucionadas inmediatamente, garantizando de esta manera todas las especificaciones del cliente.

TABLA 7: NO CONFORMIDADES.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	Al importar el script y llenar el modelo relacional, las relaciones que no contienen clave primaria se llenan con solamente el miembro derecho en la dependencia funcional principal, provocando que se cerrara el plugin.	Importar script	Prueba	S	24/5/2012 PD 25/5/2012 RA	Se corrigió la funcionalidad, permitiendo importar el script aunque existan relaciones sin clave primaria, cargando dichas relaciones tal y como aparecen en el script.

Aplicación	2	El sistema muestra un mensaje con error. Ejemplo “El mdelo está en boyce codd”. Es decir tiene una falta de ortografía en la palabra modelo, le falta la “o”.	Determinar forma normal	Prueba	NS	24/5/2012 PD 25/5/2012 RA	Se le añadió la “o” a la palabra modelo.
Aplicación	3	El sistema muestra un mensaje con error. Ejemplo “La relación esta en 2da FN”. Es decir tiene una falta de ortografía en la palabra está, le falta la tilde a la “a”.	Normalizar 2da FN	Prueba	NS	24/5/2012 PD 25/5/2012 RA	Se le añadió la tilde a la palabra “está”.

CONCLUSIONES

En el presente capítulo se hizo un análisis de los resultados de las pruebas de aceptación aplicadas al plugin a través de los diferentes casos de pruebas que cubrieron la totalidad de sus funcionalidades, para ello se analizaron las ocho HU identificadas. Permitiendo encontrar errores que afectaban el funcionamiento de esta. Los errores encontrados con las pruebas fueron corregidos lo que permitió que actualmente la aplicación esté al 100% de su funcionamiento.

CONCLUSIONES GENERALES

Al concluir el desarrollo de la presente investigación, se arriba a las siguientes conclusiones:

- ✓ Las herramientas estudiadas, no cumplen con las características que se necesitan para integrarse a la herramienta de administración de bases de datos HABD, por lo que se decidió implementar un plugin capaz de normalizar BDR hasta la FNBC, generando modelos libres de restricciones artificiales y de problemas de inserción, eliminación y actualización de los datos.

- ✓ Mediante el modelo de diseño se determinaron cinco clases del plugin de normalización de BDR.

- ✓ Fueron identificados 12 tareas de ingeniería agrupadas en ocho historias de usuario, las cuales fueron implementadas logrando la realización del plugin de normalización de BDR.

- ✓ La realización de las pruebas de aceptación al plugin de normalización de BDR, validaron las ocho funcionalidades de la aplicación, detectando en la primera iteración tres no conformidades, las cuales fueron corregidas en la segunda iteración, demostrando que el sistema cumple satisfactoriamente con los requisitos que garantizan su correcto funcionamiento.

RECOMENDACIONES

Para dar continuidad al trabajo realizado los autores de la presente investigación proponen las siguientes recomendaciones:

- Importar y exportar el script directamente desde y hacia el gestor PostgreSQL.

- Añadir las funcionalidades normalizar 4ta y 5ta FN, al plugin de normalización de BDR.

REFERENCIAS BIBLIOGRÁFICAS

1. **Date, Christopher J.** *Introducción a los Sistemas de Bases de Datos. .* Séptima Edición. s.l. : Editorial Méxicana, 2001. pp. pp 9-10, 845, 331-333, 350-351. ISBN-968-444-419-2. 960.

2. **María Mato Gracía, Rosa.** *Diseño de Bases de Datos.* Segunda edición corregida y aumentada. s.l. : Pueblo y educación, septiembre del 2005. p. pp 2. 968-444-419-2. 165.

3. **Torre, CESM-Centro de Estudios Superiores Mártires de la.** *apuntes-bd-relacionales.*

4. **PgAdmin.** [Online] http://www.pgadmin.org/.

5. **PHPpgadmin.** [Online] http://sourceforge.net/projects/phppgadmin.

6. **PHPgenerator.** [Online] http://www.sqlmaestro.com/products/postgresql/phpgenerator/.

7. **Pgaccess.** [Online] http://www.pgaccess.org.

8. **Graham, C.** *"DBMS Software Market: Flat but Not Calm".* 6 de mayo 2002. ISBN 3-540-64264-1.

9. **Batini, Carlos.** *Diseño conceptual de bases de datos.* 2 ejemplares, Donación CEFI, Copia anillada. 2004. pp. pp. 3-4. ISBN 0-201-60120-6.

10. *El modelo relacional de bases de datos.* **Quiroz, Javier.** Número 6, 2003, Boletín de Política Informática. nº 6.

11. **Orallo, José Hernández.** *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* Universitat Politècnica de València. mayo 2002. Dep. de Sistemas Informaticos y Computación..

12. **Baizán, María Covadonga Fernández.** *El modelo racional de datos: De los fundamentos a los modelos deductivos.* Ediciones Díaz de Santos. Madrid : s.n., 1987. p. 149 . ISBN 8486251613.

13. **objetos, Bases de datos orientadas a.** *Diseño de Sistemas de Bases de Datos.* 12 de abril del 2002.

14. **Seijas, Dra: Letticia M.** *Diseño de Base de Datos Relacionales,.*

15. **Mercedes, Avda Reina.** *Diseño de bases de datos relacionales.* [ed.] Sevilla. V 2005.01.1. Sevilla : V 2005.01.1 , Abril/2005. 22.

16. **Franco, Andrés Cordón.** Principles of Database and Knowledge Base Systems. [Online] 2006. [Cited: 12 4, 2011.] Computers Science Press. http://www.cs.us.es/cursos/bd-2005/tema-BD-5.pdf.

17. **Kroenke, David M.** *Procesamiento de bases de datos: fundamentos, diseño e implementación.* Octava Edición. s.l. : Prentice Hall, 2003. ISBN 970-26-0325-0 .

18. **Peter Rob, Carlos Coronel.** *Sistemas de bases de datos: diseño, implementación y administración.* 5ta edición. 2003. pp. pp 7, 21-22, 30-31, 196-197. ISBN 9706862862..

Referencias bibliográficas

19. **Lic. Yanet Espinal Martin, Manuel Enrique Puebla.** Sistema para la integracion del proceso de normalizacion de dases de datos relacionales con gestores de bases de datos (SINORGES). [Online] Universidad de las Ciencias Informaticas . http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=1748&tmpl=component&format=raw&Itemid=285.

20. mailxmail, Modulo informático. [Online] [Cited: 12 21, 2011.] http://www.mailxmail.com.

21. Curso C++, Librerías informáticas. [Online] [Cited: 11 28, 2011.] http://www.zator.com/Cpp/E1_4_4b.htm.

22. Corpoica, GESTION DE TECNOLOGÍAS DE. [Online] 9 3, 2010. [Cited: 11 2, 2011.] http://www.corpoica.org.co/sitioweb/intranet/Download/Documentos/GT-P-01ProcedimientoparaeldesarrollodeSoftwareV3.pdf.

23. **Escribano, Gerardo Fernández.** Introducción a Extreme Programming. 12 : 9, 2002.

24. **Gracia, Joaquin.** IngenieroSoftware. [Online] mayo 7, 2005. [Cited: 1 7, 2012.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

25. Lenguajes de Programacion. [Online] [Cited: 1 7, 2012.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

26. **Rojas, Alan D. Osorio.** 16 de enero de 2008.

27. Qt Creator IDE y Herramienta. [Online] [Cited: 1 15, 2012.] http://qt.nokia.com/products/developer-tools/.

28. Visual paradigm. [Online] [Cited: 1 30, 2012.] www.visual-paradigm.com/.

29. **S´anchez, Pablo.** [Online] [Cited: 2 12, 2012.] Dpto. Matemáticas, Estadística y Computación. Universidad de Cantabria, Santander (Spain). http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/practicas-1/Practicas/p4-gestionConfiguracionSoftware.pdf.

30. **Miguel.** Modelo dominio. [Online] 12 14, 2007. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio?tmpl=component&print=1&page=.

31. **Joskowicz, Ing. José.** Tareas. [Online] 2 10, 2008. [Cited: 3 4, 2012.] http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

32. Diagrama de clases. [Online] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

33. Arquitectura. [Online] [Cited: 3 19, 2012.] http://www.mastermagazine.info/termino/3916.php.

34. Patron MVC. [Online] [Cited: 4 1, 2012.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

35. Patron de diseño. [Online] [Cited: 3 4, 2012.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee.

36. Patrones. [Online] [Cited: 4 22, 2012.] http://ldc.usb.ve/~teruel/ci3711/patron3a/index.html#experto.

Referencias bibliográficas

37. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* s.l. : Autoedición, 2011. ISBN 970-686-190-4.

38. **Barrios, Johanna Rojas - Emilio.** Diseño de pruebas, Grupo ARQUISOFT . [Online] 2007 . [Cited: 3 21, 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node24.html.

Bibliografía

Date, Christopher J. *Introducción a los Sistemas de Bases de Datos. .* Séptima Edición. s.l. : Editorial Méxicana, 2001. pp. pp 9-10, 845, 331-333, 350-351. ISBN-968-444-419-2. 960.

María Mato Gracía, Rosa. *Diseño de Bases de Datos.* Segunda edición corregida y aumentada. s.l. : Pueblo y educación, septiembre del 2005. p. pp 2. 968-444-419-2. 165.

Torre, CESM-Centro de Estudios Superiores Mártires de la. *apuntes-bd-relacionales.*

PgAdmin. [Online] http://www.pgadmin.org/.

PHPpgadmin. [Online] http://sourceforge.net/projects/phppgadmin.

PHPgenerator. [Online] http://www.sqlmaestro.com/products/postgresql/phpgenerator/.

Pgaccess. [Online] http://www.pgaccess.org.

Graham, C. *"DBMS Software Market: Flat but Not Calm".* 6 de mayo 2002. ISBN 3-540-64264-1.

Batini, Carlos. *Diseño conceptual de bases de datos.* 2 ejemplares, Donación CEFI, Copia anillada. 2004. pp. pp. 3-4. ISBN 0-201-60120-6.

El modelo relacional de bases de datos. **Quiroz, Javier.** Número 6, 2003, Boletín de Política Informática. nº 6.

Orallo, José Hernández. *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* Universitat Politècnica de València. mayo 2002. Dep. de Sistemas Informaticos y Computación..

Baizán, María Covadonga Fernández. *El modelo racional de datos: De los fundamentos a los modelos deductivos.* Ediciones Díaz de Santos. Madrid : s.n., 1987. p. 149 . ISBN 8486251613.

objetos, Bases de datos orientadas a. *Diseño de Sistemas de Bases de Datos.* 12 de abril del 2002.

14. **Seijas, Dra: Letticia M.** *Diseño de Base de Datos Relacionales,.*

Mercedes, Avda Reina. *Diseño de bases de datos relacionales.* [ed.] Sevilla. V 2005.01.1. Sevilla : V 2005.01.1 , Abril/2005. 22.

Franco, Andrés Cordón. Principles of Database and Knowledge Base Systems. [Online] 2006. [Cited: 12 4, 2011.] Computers Science Press. http://www.cs.us.es/cursos/bd-2005/tema-BD-5.pdf.

Kroenke, David M. *Procesamiento de bases de datos: fundamentos, diseño e implementación.* Octava Edición. s.l. : Prentice Hall, 2003. ISBN 970-26-0325-0 .

Peter Rob, Carlos Coronel. *Sistemas de bases de datos: diseño, implementación y administración.* 5ta edición. 2003. pp. pp 7, 21-22, 30-31, 196-197. ISBN 9706862862..

Lic. Yanet Espinal Martin, Manuel Enrique Puebla. Sistema para la integracion del proceso de normalizacion de dases de datos relacionales con gestores de bases de datos (SINORGES). [Online]

Bibliografía

Universidad de las Ciencias Informaticas . http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=1748&tmpl=component&format=raw&Itemid=285.

mailxmail, Modulo informático. [Online] [Cited: 12 21, 2011.] http://www.mailxmail.com.

Curso C++, Librerías informáticas. [Online] [Cited: 11 28, 2011.] http://www.zator.com/Cpp/E1_4_4b.htm.

Corpoica, GESTION DE TECNOLOGÍAS DE. [Online] 9 3, 2010. [Cited: 11 2, 2011.] http://www.corpoica.org.co/sitioweb/intranet/Download/Documentos/GT-P-01ProcedimientoparaeldesarrollodeSoftwareV3.pdf.

Escribano, Gerardo Fernández. Introducción a Extreme Programming. 12 : 9, 2002.

Gracia, Joaquin. IngenieroSoftware. [Online] mayo 7, 2005. [Cited: 1 7, 2012.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

Lenguajes de Programacion. [Online] [Cited: 1 7, 2012.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

Rojas, Alan D. Osorio. 16 de enero de 2008.

Qt Creator IDE y Herramienta. [Online] [Cited: 1 15, 2012.] http://qt.nokia.com/products/developer-tools/.

Visual paradigm. [Online] [Cited: 1 30, 2012.] www.visual-paradigm.com/.

S´anchez, Pablo. [Online] [Cited: 2 12, 2012.] Dpto. Matemáticas, Estadística y Computación. Universidad de Cantabria, Santander (Spain). http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/practicas-1/Practicas/p4-gestionConfiguracionSoftware.pdf.

Miguel. Modelo dominio. [Online] 12 14, 2007. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio/37-modelado_negocio/46-modelo-de-dominio?tmpl=component&print=1&page=.

Joskowicz, Ing. José. Tareas. [Online] 2 10, 2008. [Cited: 3 4, 2012.] http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

Diagrama de clases. [Online] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

Arquitectura. [Online] [Cited: 3 19, 2012.] http://www.mastermagazine.info/termino/3916.php.

Patron MVC. [Online] [Cited: 4 1, 2012.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

Patron de diseño. [Online] [Cited: 3 4, 2012.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee.

Patrones. [Online] [Cited: 4 22, 2012.] http://ldc.usb.ve/~teruel/ci3711/patron3a/index.html#experto.

Joskowicz, José. *Reglas y Prácticas en eXtreme Programming.* s.l. : Autoedición, 2011. ISBN 970-686-190-4.

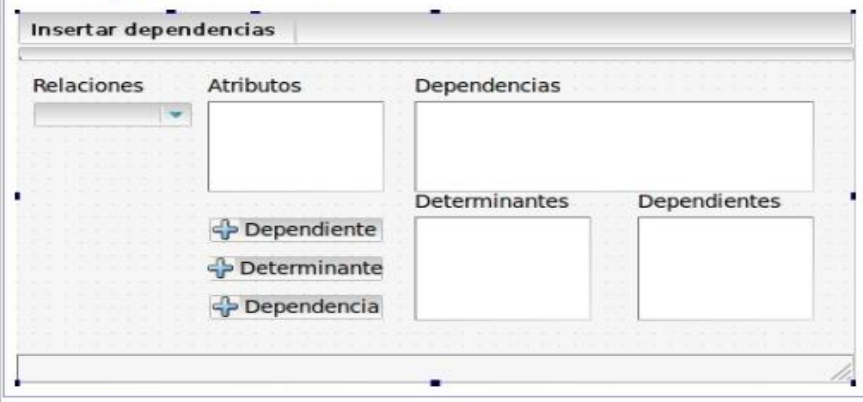
Bibliografía

Barrios, Johanna Rojas - Emilio. Diseño de pruebas, Grupo ARQUISOFT . [Online] 2007 . [Cited: 3 21, 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node24.html.

ANEXOS

A continuación se exhibirán en los anexos, sólo dos ejemplos de HU, TI, TCRC y Casos de pruebas. Para más información remitirse al expediente de proyecto: https://repositorio.datec.prod.uci.cu/svn/cbd_tecnologia/.

Anexo # 1 “HU 2, Insertar dependencias funcionales”.

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Insertar dependencias funcionales.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Grether Méndez Gómez	**Iteración asignada:**1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Muy Alta	**Puntos reales:** 2 semanas
Descripción: El usuario tendrá la posibilidad de insertar las dependencias funcionales necesarias a partir de las reglas definidas en el negocio.	
Observaciones: El plugin solo normalizará si se le entran las dependencias funcionales.	
Prototipos de interfaces: 	

Anexo # 2 “TCRC, Normalizar”.

Tarjeta CRC	
Clase: Normalizar	
Responsabilidades	**Colaboraciones**
Normalizar 2da FN. Normalizar 3era FN. Normalizar FNBC.	Script. Dependencias. Determinar FN.

Anexo # 3 “TI 1, Importar Script”.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Importar Script.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 semana
Fecha Inicio: 3/2/12	**Fecha Fin:** 9/2/12
Programador Responsable: Osmil Guillen Tamayo	
Descripción: El sistema brindará la opción de seleccionar el Script en el directorio donde se encuentra guardado, para poder cargarlo en la aplicación.	

Anexo # 4 Caso de prueba Insertar dependencias.

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Crear dependencias funcionales correctas	El usuario selecciona los atributos determinantes y los atributos dependientes, luego se añade las dependencias.	V (Relación: Viaje)	V (Determinante: NumViaje Dependiente: Numero)	El sistema añade la dependencia creada por el usuario.	1. El usuario selecciona la relación a la cual desea añadir la dependencia. 2. Luego escoge el atributo en el componente Atributos. 3. Seguidamente este acciona el botón correspondiente para pasar este atributo a ser determinante o dependiente. 4. Se procede a accionar el botón **Adicionar,** quien adiciona la dependencia al componente de las Dependencias.
EC 1.2 Crear dependencias funcionales incorrectas.	El usuario añade una dependencia ya existente.	V (Relación: Viaje)	I (Determinante: IDViaje Dependiente: NumViaje, Chapa, Numero)	El sistema muestra un mensaje ¨La dependencia ya existe.¨	

Anexo # 5 Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
variable1	Relación	Campo de selección (Combo Box).	No	En este campo se procede a seleccionar la relación a la cual se le insertará la dependencia.
variable2	Atributo	Campo de selección (List Widget).	No	En este campo se procede a seleccionar los atributos, para formar la dependencia funcional.

Glosario de términos

BDR.- **Bases de Datos Relacional**. Es una base de datos que se percibe por los usuarios como una colección de tablas.

Herramientas CASE. - **Computer Aided Software Engineering.** Aplicaciones Informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

IBM.- **International Business Machines**. El mayor fabricante de ordenadores del mundo. IBM es el inventor del PC y con sus sistemas medios (AS/400) y sus grandes ordenadores (mainframes) han revolucionado el mundo de la empresa.

Plugin: pequeño programa que proporciona alguna funcionalidad específica a otra aplicación mayor o más compleja.

SGI. - **Sistemas de Gestión de la Información**. Es una colección de datos debidamente recopilados y estructurados, que proporcionan información sobre una parcela de la realidad.

SGBD. - **Sistemas Gestores o de Gestión de Bases de Datos**. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.