Universidad de las Ciencias Informáticas
Facultad 15

Tema: “Diseño e implementación de la base de datos para generar una Base de Conocimiento de Patrones”.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informático

Autor: Vladimir Creagh Ortiz

Tutor: Ginisleisy Morales Gómez

Junio 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 15 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Vladimir Creagh Ortiz

_____
Firma de Autor

Ginisleisy Morales Gómez

_____
Firma de Tutor

AGRADECIMIENTOS

A mi mamá, por ser mi bujía inspiradora, esa persona que en los momentos más difíciles siempre está ahí, para darme fuerzas, amor y ayudarme a salir adelante.
A mi papá, por ser el mejor papá del mundo.
A mis hermanas que son la luz de mis ojos.
A mi abuela por estar a mi lado desde pequeñito, siempre enseñándome y guiándome.
A mi tío Miguel, que para mí es como mi segundo padre, por brindarme su apoyo incondicional en cada momento de mi vida y saberme guiar con su ejemplo por ella.
A Yani, por ocupar un lugar tan importante en mi corazón y por su inmensa ayuda para que se pudiera cumplir este sueño.
A mi tutora Ginita, porque supo guiarme y apoyarme en todo momento, por su sacrificio y entrega, para ella es también este título.
A mis amigos Yaser, Osnel, Genoldis, Yuniel, Alain, que los considero mis hermanos.
A mi familia y a la familia de mi novia por preocuparse por mí.
A Alain, Iosmel, Shalimar, Vlamir, Jorge Luis por su ayuda en la realización de este trabajo.
A todas aquellas personas que de una forma u otra hicieron posible este sueño.

Dedicatoria

A mis padres, porque siempre confiaron en mí, para ellos aquí está el mayor de mis regalos
A mi abuela
A mis tío Miguel
A Yani
A mi familia

Vladimir Creagh Ortiz

“(...) Después de escalar una montaña muy alta, descubrimos que hay muchas otras montañas por escalar (...)”

Nelson Mandela

RESUMEN

La aplicación de los patrones en los proyectos productivos es una práctica que juega un papel fundamental para alcanzar software de mayor calidad. Actualmente esta no es una actividad que se realice frecuentemente ya que no se cuenta con una forma de almacenar la información referente a los mismos. En busca de soluciones surge la investigación actual y la idea de desarrollar una Base de Datos (BD) que permita y ayude a gestionar la información de los patrones. En este trabajo se realiza el diseño e implementación de la BD que contribuirá al almacenamiento y manejo de los datos asociados a los patrones.

Palabras Clave

Base de datos, patrones, gestionar, información, almacenamiento.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1: "FUNDAMENTACIÓN TEÓRICA" 6

INTRODUCCIÓN 6
1 HISTORIA 6
1.1 TENDENCIA DE LAS BASES DE DATOS 7
INTERNACIONAL 7
NACIONAL 8
1.2 CONCEPTOS DE IMPORTANCIA 10
1.3 ¿QUÉ ES UNA BASE DE DATOS? 10
1.3.1 CONCEPTOS DE SISTEMA GESTOR DE BASE DE DATOS. 11
1.3.2 BASE DE CONOCIMIENTO 11
1.3.3 BASE DE CONOCIMIENTO VS BASE DE DATOS 11
1.4 MODELOS DE BASE DE DATOS 12
1.4.1 MODELO JERÁRQUICO 12
1.4.2 MODELO RED 13
1.4.3 MODELO RELACIONAL 13
1.4.4 MODELO ORIENTADO A OBJETO 13
1.4.5 MODELO DOCUMENTALES 14
1.6 ETAPAS DE DISEÑO 14
1.6.1 DISEÑO CONCEPTUAL 14
1.6.2 DISEÑO LÓGICO 15
1.6.3 DISEÑO FÍSICO 16
1.7 METODOLOGÍA 17
1.7.1 RUP 17
1.8 HERRAMIENTAS 20
1.8.1 HERRAMIENTAS CASE 20
1.8.2 SISTEMAS GESTORES DE BASE DE DATOS (SGBD) 23
1.9 HERRAMIENTAS DE ADMINISTRACIÓN 26
1.10 CONCLUSIONES PARCIALES 27

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA……...28

INTRODUCCIÓN 28
2.1 ARQUITECTURA DE POSTGRE 29
2.2 REQUISITOS FUNCIONALES 31
2.3 DIAGRAMA ENTIDAD-RELACIÓN 33
2.4 DIAGRAMA DE ESQUEMAS RELACIONALES 34
2.5 DESCRIPCIÓN DEL DIAGRAMA DE ESQUEMAS RELACIONALES 35
2.6 SEGURIDAD DE LA BASE DE DATOS 43

2.7 CONCLUSIONES PARCIALES 44

Capítulo 3: Implementación y validación de la base de datos.....45

INTRODUCCIÓN 45
3.1 VALIDACIÓN TEÓRICA DEL DISEÑO DE LA BASE DE DATOS. 46
3.1.1 INTEGRIDAD DE LOS DATOS. 46
3.1.2 NORMALIZACIÓN. 48
3.2 VALIDACIÓN DE LA BASE DE DATOS POR LISTA DE CHEQUEO. 50
3.3 VALIDACIÓN FUNCIONAL 54
3.4 TÉCNICAS DE OPTIMIZACIÓN. 55
3.4.1 TÉCNICA DE OPTIMIZACIÓN DE CONSULTAS. 55
3.4.2 TÉCNICA DE OPTIMIZACIÓN DE DISEÑO. 55
3.5 VALIDACIÓN POR TIEMPO DE RESPUESTA. 56
3.6 CONCLUSIONES PARCIALES. 64

BIBLIOGRAFÍA **66**

ANEXOS **68**

INTRODUCCIÓN

Desde la antigüedad ha existido la necesidad de contar con sistemas de información donde centralizar los conocimientos, con el objetivo de conservar la información y permitir futuras búsquedas, las que resultaban algo complejas.

Es por esto que con el surgimiento de la Informática, se desarrollan los Sistemas Gestores de Base de Datos (SGBD), los que han evolucionado, permitiendo almacenar, organizar y realizar consultas de datos, en breves períodos de tiempo y de forma eficiente.

Actualmente el desarrollo de software en todo el mundo está inmerso en una constante evolución lo cual provoca la necesidad de crear software de mayor complejidad con los cuales han surgido algunos inconvenientes, dándole una efectiva y rápida solución con el uso de los patrones.

Particularmente en Cuba el desarrollo del software en sus inicios se veía frenado por el poco personal capacitado, pero se logró un avance importante con la creación de la Universidad de las Ciencias Informáticas (UCI), la cual logra establecer una pauta en el desarrollo de software, convirtiéndose en el mayor centro de desarrollo de software del país, ya que cuenta con una de las mayores infraestructuras tecnológicas existentes a nivel nacional, además de una gran concentración de personal especializado, lo cual permite la implementación de múltiples programas para la informatización de sectores estratégicos, evidencia de ello son los proyectos vinculados a el sector jurídico. Ejemplo: en la Facultad 15, se desarrollan proyectos para informatizar dicho sector, con los proyectos Sistema de Gestión Fiscal y TPC[1].

[1] Tribunal Popular de la República de Cuba

Al ser la UCI un centro estudiantil, donde el 82%[2] de los que trabajan en el desarrollo de proyectos de software son estudiantes, se crea un problema a la hora del desarrollo de los mismos, debido a:

- Insuficiente conocimiento sobre los patrones.
- Poco uso de los patrones al desarrollar los software.
- La falta de una política central sobre el empleo de los patrones en los procesos de desarrollo de Software en la UCI.

Provocando que el conocimiento sobre los mismos se pierda o no se aproveche de forma eficiente por lo cual cada vez que estos se van a aplicar en los proyectos, conlleva a una pérdida de tiempo en investigación o en el mejor de los casos preguntándole a un compañero sobre patrones. Es por eso que existe una necesidad cada vez mayor de contar con una forma o método para almacenar dicho conocimiento. Problema que pudiera mitigarse en parte si se fomentara el intercambio de conocimiento sobre patrones en los diferentes eventos científicos que se realizan en la universidad tanto a nivel de facultad como a nivel central. Evitando así un gasto innecesario de tiempo en investigación, que se atrase el proyecto y logrando un aumento de la calidad, flexibilidad y robustez de los productos.

Luego de un análisis de la problemática anterior y con el fin de solucionar esta carencia se propone el problema expresado con la siguiente interrogante: ¿Cómo hacer perdurable y reutilizable el conocimiento sobre los patrones?

El Objeto de estudio de este trabajo será: Proceso de Desarrollo de Software y como Campo de acción: Diseño e Implementación de Bases de Datos.

---

[2] Cantidad de profesores vinculados proyectos ->112;
Cantidad de estudiantes vinculados a proyectos->528;
Promedia 82% de estudiantes por proyecto.

Por tanto el objetivo general es: Desarrollar una base de datos para generar una base de conocimiento que establezca la gestión de patrones, como un sistema de consultoría y gestor del conocimiento temático que permita mejorar la calidad del diseño del software de la UCI.

De los que se derivan los siguientes objetivos específicos:

- Elaborar el diseño teórico de la investigación.
- Realizar un modelado correcto de la base de datos.
- Implementar la base de datos.

Hipótesis

Si se elabora una base de datos para gestionar la información de una base de conocimiento de patrones, entonces se facilitará el proceso de estudio y utilización de patrones en la Facultad 15 y en la UCI.

Para lograr el cumplimiento de los objetivos anteriores se deben realizar las siguientes.

Tareas de Investigación:

- Realización de búsquedas bibliográficas encaminadas a:
    - Realizar un estudio del arte sobre las tendencias mundiales y nacionales sobre bases de datos.
    - Aspectos teóricos relacionados con el diseño de base de datos
    - Cómo se lleva a cabo el proceso de modelado de una base de datos.
    - Estudiar los tipos de topologías de bases de datos y seleccionar una para su aplicación.

- Realizar el estudio de las principales herramientas utilizadas en el manejo de datos.

- Estudio e identificación de los aspectos fundamentales sobre bases de conocimientos y patrones.
- Recopilación de todos los patrones a utilizar en el proceso de desarrollo de software según bibliografías.
- Modelado y diseño de la base de datos.
- Implementación de la base de datos.

Los métodos científicos son los que brindan una base para el estudio de lo que se va a hacer en el trabajo. Para lograr esto primeramente se debe establecer cuáles son los métodos a utilizar.

Métodos Teóricos:

- **Histórico lógico:** Se utiliza este método investigativo al estudiar cómo ha evolucionado y se han desarrollado las bases de datos desde su surgimiento hasta nuestros días, sus herramientas, formas de trabajos, modelos, etc.
- **Modelación:** Este método se usa para modelar las bases de datos para una mayor comprensión del trabajo que se realiza y los objetivos que se deben cumplir, dando respuesta a los requerimientos planteados con anterioridad.
- **Analítico-Sintético:** Utilizado para el análisis de la bibliografía referente al trabajo que se realizará.

Métodos Empíricos:

- **Entrevista:** Con la ayuda de este método se hace una modelación del negocio acorde a las necesidades del cliente, además se lleva a cabo un levantamiento de requisitos óptimo en la solución de la problemática que afecta al cliente.

Estructura del contenido

La investigación consta de 3 capítulos, en los cuales se desarrollan aspectos de importancia para lograr el objetivo que se persigue.

Capítulo I: Fundamentación Teórica

En este capítulo se realiza un estudio del estado del arte, sobre bases de datos (BD), base de conocimientos y patrones, así como un análisis de las principales herramientas que son utilizadas en el desarrollo de BD y se justifica además el por qué del uso de la herramienta escogida para el desarrollo de la BD.

Capítulo II: Análisis y diseño de la solución propuesta

En este capítulo se definen los requisitos funcionales y no funcionales del sistema previamente definidos por el cliente, se hace un estudio de los patrones de diseño de bases de datos y se define la nomenclatura. Se representa el diagrama de clases persistentes, se detalla cada una de las clases y se diseña la BD.

Capítulo III: Implementación y validación de la base de datos.

En este capítulo se realiza la implementación de la base de datos además de la validación y análisis de los resultados del uso de la BD para suplir la carencia de conocimientos sobre patrones. También se estudiará y normalizará la BD, así como el análisis y eliminación de la redundancia de la información y seguridad de la BD.

CAPÍTULO 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

En el presente Capítulo se hace una breve reseña histórica sobre el origen de las Bases de Datos, así como de las tendencias nacionales e internacionales. Además menciona los distintos modelos de Bases de Datos (BD) y las etapas de diseño. También se realizó un estudio sobre las distintas herramientas para gestionar y modelar la BD, haciendo énfasis en las que se utilizarán durante el desarrollo del trabajo.

1 HISTORIA

Los predecesores de los sistemas de bases de datos fueron los sistemas de ficheros, pero aunque no hay un momento concreto en el que hayan dejado de utilizarse para dar paso a las bases de datos, la creciente complejidad de los programas y los grandes volúmenes de datos existentes conllevaron al surgimiento de las bases de datos. Era necesaria una forma de almacenamiento que permitiera la realización de operaciones complejas y que garantizara mayor seguridad e integridad de los datos.

La primera topología que surge es la jerárquica pero debido a que el acceso a los datos en ella era unidimensional se hacía muy complejo su uso. Como solución a la necesidad de representar relaciones entre datos más complejos que las que se podían modelar con los sistemas jerárquicos surge el sistema de red.

Los sistemas jerárquico y de red constituyen la primera generación de los SGBD. Pero estos sistemas presentan algunos inconvenientes:

- Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.
- La independencia de datos es mínima.
- No tienen un fundamento teórico.

Para solucionar los problemas de las topologías anteriores a finales de los setenta y principios de los ochenta se comienza a desarrollar el sistema relacional el cual proporcionó una absoluta libertad en las relaciones entre las tablas e incorporó un lenguaje de consultas estructurado denominado SQL (Lenguaje Estructurado de Consultas).

Como respuesta a la creciente complejidad de las aplicaciones que requieren bases de datos, surgió el modelo de datos orientado a objetos para eliminar las deficiencias de los modelos anteriores.

1.1 TENDENCIA DE LAS BASES DE DATOS

INTERNACIONAL

Desde la década del 70 el uso de los ficheros para almacenar la información fue cada vez menor, puesto que para trabajar de un modo más efectivo, surgieron las *bases de datos (*BD*)* y los *sistemas de gestión de bases de datos (SGBD)* y su uso se generalizó en todo el ámbito empresarial al reconocer su importancia y efectividad en el manejo de los datos.

Debido al auge de la utilización de las BD han surgido nuevas aplicaciones para estas como son:

- Bases de datos de multimedia: con dibujos, videos, sonidos.
- Bases de datos orientadas a objetos.
- Bases de datos distribuidas.

- Sistemas de información geográfica: con mapas, datos del tiempo, imágenes de satélite.
- Almacenes de datos (o Data Warehouses): analizan y extraen información útil para la toma de decisiones.
- Base de datos activas y de tiempo real: útiles en procesos de control industrial y fabricación.

Caso de estudio:

HI5 usa PostgreSQL como el SGBD principal para la mayoría de sus de necesidades comerciales de salida y han tenido bastante éxito creciendo como compañía en cuanto a:

- El número de usuarios que posee.
- La complejidad de servicios que ofrece.
- La versatilidad de sus servicios.

NACIONAL

El desarrollo y uso de las bases de datos en Cuba ha ido proliferando de forma considerable, a partir de la utilización ordenada y masiva de las tecnologías de la información y las comunicaciones en todas las esferas de la sociedad cubana.

La clasificación de los SGBD se realiza a través del modelo lógico en el que se basa. La mayoría de los SGBD utilizan con mayor frecuencia el modelo relacional, el de red y el jerárquico, aunque en algunos más modernos se utilizan los orientados a objetos. Los SGBD más utilizados en Cuba son MySQL y Access, PostgreSQL y Oracle en algunos casos.

A pesar de que Cuba no tiene acceso a los principales avances de la ciencia y la técnica a nivel mundial por estar sometida a un férreo bloqueo económico, el desarrollo de la informática, y con ella las BD, en el país se ha realizado de una manera constante. Es por ello que para el desarrollo de bases de datos se utilizó el gestor de bases de datos Access que viene con el paquete de Office de Windows como herramienta para la elaboración de las

bases de datos de las instituciones cubanas, por su facilidad de uso y la poca complejidad que presenta, ejemplo de ello es:

Instituto Nacional de Recursos Hidráulicos (INRH) que cuenta con una base de datos elaborada en Microsoft Access para recopilar los datos hidrológicos y de la calidad del agua. También podemos observar casos como el Acuario Nacional de Cuba (ANC), el Centro de Inspección y Control Ambiental (CICA) y el Jardín Botánico Nacional entre muchos otros que usan este gestor de bases de datos.

Sin embargo como consecuencia de los efectos del bloqueo y por las posibilidades que brinda al país el uso del software libre se ha visto una progresiva migración hacia los SGBD de código abierto como son el Postgre y MySQL, los cuales presentan licencia de código abierto y están a disposición del público de forma gratuita.

Ejemplo de sitios cubanos donde son utilizados los gestores:

- Centro Provincial de Información de Ciencias Médicas Las Tunas:

  Con el principal objetivo de dotar a la Biblioteca Médica Provincial de Las Tunas de un espacio Web que mostrara los servicios, se elabora una plataforma Web. Siempre se pensó en la utilización de la Arquitectura Cliente – Servidor por la amplia gama de posibilidades que brinda. La infraestructura de software utilizada fue: el paquete Wamp5 que dispone de un servidor Apache, MySQL como sistema gestor de bases de datos y PHP como lenguaje de programación.

- En la UNIVERSIDAD DE CIENCIAS INFORMÁTICA (UCI):

  El Proyecto Kainos de la FACULTAD 1 tiene una base de datos relacional, usa PostgreSQL como SGBD y PHP y JAVA como lenguajes de alto nivel de programación. Este proyecto utiliza herramientas para gestionar el acceso a datos como Drupal en el caso de PHP e Hibérnate para JAVA.

1.2 CONCEPTOS DE IMPORTANCIA

Atributos o campos: Es la unidad menor de información, es decir, representa una propiedad de un objeto.

Dominio: Son los diferentes valores que pueden tomar estos atributos.

Ocurrencia del atributo: Uno de los valores de los atributos que puede tomar y que se definió con anterioridad en el dominio.

Artículo o registro: Una colección de campos y que representa un objeto con sus propiedades.

Ocurrencia de artículo o tupla: Es un grupo de ocurrencia de campos relacionados, representando una asociación entre ellos.

Llave o clave: Atributo o conjunto de atributos de un artículo que define que cada ocurrencia de artículo de la base de datos sea único.

Interrelaciones: Las interrelaciones son las relaciones que existen entre varias tablas del sistema.

1.3 ¿QUÉ ES UNA BASE DE DATOS?

¨Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquinas accesibles en tiempo real y compatible con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.¨ (Rodriguez)

1.3.1 CONCEPTOS DE SISTEMA GESTOR DE BASE DE DATOS.

Sistema de gestión de base de datos (o en inglés Database management system (DBMS)): es una agrupación de programas que sirven para definir, construir y manipular una base de datos.

1.3.2 BASE DE CONOCIMIENTO

Las Bases de Conocimiento (*KB: Knowledge Base*) surgieron a partir de la investigación en Inteligencia Artificial como respuesta a las necesidades que las aplicaciones de esta disciplina planteaban.

Las bases de conocimiento son la evolución lógica de los sistemas de bases de datos tradicionales, en un intento de plasmar no ya cantidades ingentes de datos, sino elementos de conocimiento (normalmente en forma de hechos y reglas) así como la manera en que éste ha de ser utilizado. También se les trata de dotar de conocimiento sobre sí mismas, es decir, una KB ha de "saber lo que sabe". Por ejemplo, ante una pregunta del tipo "¿Tienen todos los estudiantes de la UCI aprobada la prueba de nivel de programación?", una base de datos tras consultar la información relacionada con los estudiantes, daría una respuesta afirmativa o negativa, independientemente de que tenga o no la información correspondiente a estos trabajadores; en cambio, una KB respondería "sí", "no" o "no lo sé", en el caso de que le faltase información relativa a las notas sobre alguno de los estudiantes o de que no tuviese información sobre "todos" los estudiantes.

1.3.3 BASE DE CONOCIMIENTO VS BASE DE DATOS

Una Base de Datos almacena únicamente hechos. Las funciones que el gestor de base de datos se limita a facilitar son, fundamentalmente, las de edición y consulta de los datos.

Una base de conocimiento, por otra parte, puede almacenar, además de hechos, un conjunto de reglas que se sirven de esos hechos para obtener información que no se encuentra almacenada de forma explícita. El tipo de base de conocimiento al que se dota de

una considerable capacidad de deducción a partir de la información que contiene se denomina sistema experto.

Ambos sistemas de información cuentan con sus correspondientes gestores para simplificar al administrador las tareas comunes de mantenimiento: el sistema gestor de bases de datos (DBMS: Database Management System) y el sistema gestor de bases de conocimiento (KBMS: Knowledge Base Management System). También encontramos diferencias sustanciales en este aspecto. Básicamente los DBMSs actuales se encuentran perfectamente estandarizados, ofreciendo un número de características y metodologías comunes que posibilitan la comunicación entre diversos tipos. La comercialización de los KBMSs es prácticamente anecdótica, y los esfuerzos en cuanto a su estandarización se están produciendo en estos momentos, en lo que se ha dado en denominar KIF (Knowledge Interchange Format o Formato de intercambio de conocimientos).

1.4 Modelos de Base de Datos

1.4.1 MODELO JERÁRQUICO

Una base de datos jerárquica consiste en una colección de registros que se conectan entre sí por medio de ligas. Los registros y las ligas son similares a los del modelo de red, pero en el modelo jerárquico se organiza en forma de árbol con raíz (donde la raíz es nodo ficticio); de tal manera que una base de datos jerárquica es una colección de árboles de este tipo, formando un bosque.

A cada árbol con raíz se le denomina árbol de base de datos.

En este modelo un registro puede repetirse en varios sitios pudiendo ocasionar los siguientes problemas:

- Riesgos de la inconsistencia al llevar a cabo actualizaciones.
- Inevitable desperdicio de espacio en el medio de almacenamiento secundario.

1.4.2 MODELO RED

Este modelo se distingue del jerárquico porque en su concepto de nodo: permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

1.4.3 MODELO RELACIONAL

Una base de datos relacional es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Permiten establecer relaciones entre los datos (que están guardados en tablas), y trabajar con ellos conjuntamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

A diferencia de otros modelos como el jerárquico y el de red en este modelo no tiene importancia el lugar y la forma en que se almacenan los datos. Permitiendo una considerable ventaja al hacerlo más fácil de entender y de utilizar para usuarios ocasionales de la BD.

1.4.4 MODELO ORIENTADO A OBJETO

A finales de los 80's aparecieron las primeras BDOO, es una base de datos inteligente. Soporta el paradigma orientado a objetos almacenando datos y métodos, y no sólo datos. Está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Evita el acceso a los datos; esto es mediante los métodos almacenados en ella. Es más segura ya que no permite tener acceso a los datos (objetos); esto debido a que para poder entrar se tiene que hacer por los métodos que haya utilizado el programador. (Tijerina)

Las Base de Datos orientada a objetos incorporan todos los conceptos importantes del paradigma de objetos:

- Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.

- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

1.4.5 MODELO DOCUMENTALES

En las BD documentales cada registro se corresponde con un documento, permitiendo la indexación a texto completo y la realización de búsquedas más potentes. Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

1.5 MODELO UTILIZADO

Para el desarrollo de este trabajo se usó el modelo de Base de Dato Relacional porque permite una absoluta libertad en las relaciones entre las tablas, es el más usado en la actualidad para representar problemas reales y para realizar la administración de datos de forma dinámica. Además es de fácil utilización por usuarios esporádicos, lo cual facilita su uso y mantenimiento.

1.6 ETAPAS DE DISEÑO

1.6.1 DISEÑO CONCEPTUAL

En esta etapa se debe construir un esquema de la información que se usa en el proyecto, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos del proyecto: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

1.6.2 DISEÑO LÓGICO

El diseño lógico es el proceso de construir un esquema de la información que utiliza el proyecto, basándose en un modelo de base de datos específico e independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La *normalización* es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento del negocio y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel del negocio, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

1.6.3 DISEÑO FÍSICO

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.

- Diseñar el modelo de seguridad del sistema.

1.7 Metodología

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Es por eso que en el siguiente trabajo se utilizó la metodología RUP[3] junto al Lenguaje Unificado de Modelado (UML).

1.7.1 RUP

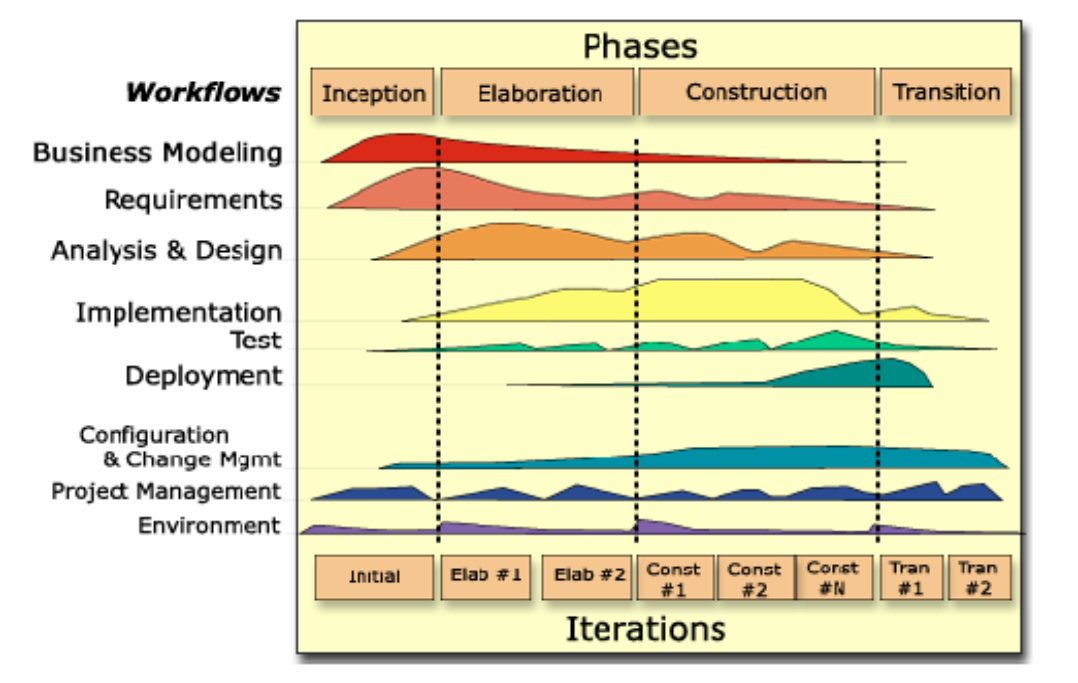
El Proceso Unificad Racional (RUP) es el resultado de varios años de desarrollo y la unión de técnicas de desarrollo a través de UML, convirtiéndose así en la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

[3] Proceso Unificado Racional

Características principales de RUP:



- Guiado por Casos de Uso.
- Iterativo e Incremental.
- Centrado en la Arquitectura.

El rol de Diseñador de BD según esta metodología debe desarrollar una serie de artefactos y actividades para garantizar el cumplimiento satisfactorio de su labor, entre ellas está:

- Identificar las clases persistentes.
- Elaborar el diagrama de clases persistentes.
- Definir las tablas de referencia.
- Crear clave primaria y restricciones de integridad.
- Definir las reglas de integridad referencial y de la información.
- Normalizar el diseño de la BD para su optimización.

El Diseñador de BD es responsable de la integridad del modelo de datos, y de asegurar que los datos modelados sean correctos, coherentes y comprensibles.

UML

El Lenguaje de Modelado Unificado (UML) es un lenguaje que se utiliza para especificar, visualizar, construir y documentar los artefactos de sistemas intensivos de software. UML es gratuito, accesible a todos, y conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelado de sistemas grandes y complejos.

UML cuenta con varios tipos de diagramas.

- Diagramas de Estructura.
- Diagramas de Comportamiento.
- Diagramas de Interacción.

1.8 HERRAMIENTAS

Para realizar el diseño y construcción de una Base de Datos es necesaria la utilización de herramientas que permitan un correcto y rápido modelado de la misma, el control de la seguridad de la BD, la integridad de los datos así como su accesibilidad a través de SGBD. Entre las disímiles herramientas que permiten esto se encuentra:

1.8.1 HERRAMIENTAS CASE

ERWIN

Erwin es una herramienta CASE empleada en el diseño de base de datos, con la cual se puede hacer un trabajo productivo por las facilidades que brinda para la generación y mantenimiento de aplicaciones de forma sencilla para el diseñador. Permite realizar el diseño del modelo lógico de los requerimientos de información, así como el diseño del modelo físico, ya con nivel mayor, refinando las características de la base de datos diseñada. Con esta herramienta CASE es posible visualizar la estructura de la base de datos diseñada, lo cual tiene como ventaja que el diseñador pueda observar en su totalidad el trabajo realizado, para realizar un análisis y si fuera necesario hacer cambios en busca de optimizar el diseño final. Esta herramienta permite generar, de forma automática, las tablas y el código referente a los procedimientos almacenados para los principales tipos de bases de datos. La migración automática garantiza la integridad referencial de la base de datos, es decir, evita la pérdida de información durante este proceso. Erwin puede hacer una conexión entre dos bases de datos, una diseñada y otra sin diseño, estableciendo una transferencia entre ellas y la aplicación de ingeniería reversa. Mediante esta conexión puede automáticamente generar índices, vistas, tablas, reglas de integridad referencial (llaves foráneas, llaves primarias), restricciones de dominios, campos y valores por defecto. Erwin básicamente soporta bases de datos del tipo relacionales SQL, y es compatible con otras como: Oracle, Microsoft SQL Server, Sybase, PostgreSQL, dBase, FoxPro, Informix, SQL Base y SQL Anyware, entre otras. Con un mismo modelo es posible generar múltiples bases de datos o hacer una conversión de una aplicación de una base de datos a otra. Es

compatible con los sistemas operativos de la familia Windows, como Windows 95, Windows 98, Windows XP y Windows NT.

Luego de un profundo análisis se determinó que la herramienta a utilizar es el Erwin:

- El trabajo se desarrolla en un gestor relacional, por lo que es conveniente el uso del Erwin puesto que esta herramienta modela el diseño de las bases de datos de modo relacional.
- La herramienta Erwin permite el diseño y construcción de bases de datos a nivel lógico y físico permitiendo la sincronización bidireccional entre ambos diseños, lo que posibilita que si se hace algún cambio en uno de los dos diseños antes mencionados, se refleje de forma automática en el otro.
- Como en la base de datos se manipulan grandes volúmenes de información pues es conveniente utilizar la herramienta CASE Erwin debido a que esta permite la generación automática de procedimientos almacenados que ayudaran a agilizar el trabajo con la base de datos.
- Erwin está equipado para crear y manejar diseños de bases de datos funcionales y confiables.
- Ofrece la posibilidad de construir automáticamente bases de datos y documentación basada en HTML.
- Permite la creación de reportes de forma sencilla.
- Da la posibilidad de hacer reingeniería inversa de bases de datos.
- Realiza el diseño de la base de datos de forma óptima.
- Permite el indexado de tablas.
- Posibilita el direccionamiento de datos.

DBDesigner 4

DBDesigner 4 es un software visual que nos ayuda para el diseño, modelado, creación y mantenimiento de nuestras bases de datos. Permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en MySQL, Oracle, MSSQL y otras bases de datos ODBC y modelos XML. Dispone de una representación visual de las tablas y las relaciones que figuran en su proyecto. Se puede ver rápidamente los campos en una tabla o cómo cada cuadro se refiere a los demás. DBDesigner puede exportar el esquema de la base de datos en un SQL script, o conectarse directamente a un soporte de base de datos y crear allí. También puede importar de las BD ya existentes. Scripts SQL DB o de generación. Por supuesto, puede guardar un proyecto en su formato original (XML) de manera que toda la información se mantiene (por ejemplo, no se puede guardar las posiciones de los cuadros en el espacio de trabajo en una SQL script). Debido a su arquitectura de plugin, DBDesigner es fácilmente extensible para trabajar con muchos servidores de Bases de Datos.

Principales características de DBDesigner

- Disponible para Linux y Windows.
- Modo diseño y modo Consulta.
- Ingeniería inversa para las bases de datos MySQL, Oracle, MSSQL y cualquier base de datos ODBC.
- Generación de esquemas definido por el usuario.
- Sincronización del modelo a la base de datos.
- Soporte de índices.
- Soporte de entidades débiles.
- Sincronización de Inserts estándar.
- Capacidad de documentar la base de datos.
- Impresión avanzada de modelos.
- Soporta plugins.

VISUAL PARADIGM

Visual Paradigm es una herramienta CASE (Computer-Aided Software Engineering) que utiliza UML como lenguaje de modelado. Se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- JBuilder
- NetBeans IDE
- Oracle JDeveloper
- BEA Weblogic

Está disponible en varias ediciones, cada una destinada según las necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Ofrece ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras, integración con MS Visio, plugins, integración con Visual Studio. Entre sus nuevas características se incluyen el modelado colaborativo con CVS y Subversión, interoperabilidad con modelos UML2 a través de XMI, etc. Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

1.8.2 SISTEMAS GESTORES DE BASE DE DATOS (SGBD)

MYSQL

MySQL es una idea originaria de la empresa de código abierto MySQL AB establecida inicialmente en Suecia en 1995. El objetivo que persigue es que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad. MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU, disponibles bajo sistema operativo Windows o Linux.

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-hilo, multi-usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. El software MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los

términos de la licencia GNU General Public License (GPL) o pueden adquirir una licencia comercial estándar de MySQL AB.

Desventajas:

Aunque MySQL se incluye en el grupo de sistemas de bases de datos relacionales, sus desarrolladores fueron implementando únicamente lo que necesitaban para garantizar su funcionamiento óptimo, y carece de algunas de sus principales características como:

- Subconsultas.
- Procedimientos almacenados y desencadenadores.
- Transacciones: A partir de las últimas versiones se incluye el soporte para transacciones, pero hay que activarlo de modo especial, no existe por defecto.

POSTGRESQL

PostgreSQL está considerado como uno de los gestores de bases de datos de código abierto más avanzados del mundo. Es básicamente una herramienta Cliente/Servidor para la gestión de bases de datos. PostgreSQL es un SGBD objeto-relacional libre, su licencia es de tipo "BSD"[4] y por tanto la sostenibilidad del esfuerzo está asegurada. Comparándolo con otros SGBD's libres como MySQL, Ingres o Firebird, dada la amplitud de posibilidades que PostgreSQL ofrece, es sin duda el SGBD que más se asemeja en versatilidad a los SGBD propietarios como Oracle, Sybase, DB2 o SQLServer.

Características de PostgreSQL

- Implementación del estándar SQL92/SQL99.

[4] La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Cumple la prueba ACID (Atomicity (atomicidad), Consistency (consistencia), Integrity (integridad), Durability (durabilidad)) y tiene soporte completo para llaves foráneas, joins, vistas, subconsultas (incluyendo subconsultas en la cláusula FROM) y procedimientos almacenados en varios lenguajes.

Posee productos que lo complementan, y que ayudan a mejorar su desempeño, eficiencia y manipulación:

- PgCluster y Slony-I: Utilizados en la replicación de datos, el primero en réplicas multi-maestro y el segundo en réplicas maestro - esclavo.
- PgAdmin3, PgAccess, PhpPgAdmin, Psql: Herramientas de administración.

El principal soporte de PostgreSQL lo provee la gran comunidad de usuarios que existe en el mundo, quienes aportan experiencias y soluciones obtenidas del trabajo con este gestor.

1.9 HERRAMIENTAS DE ADMINISTRACIÓN

PgAdmin es una herramienta de propósito general para diseñar, mantener, y administrar las bases de datos de PostgreSQL. Funciona bajo Windows 95/98 y NT. La interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. Conexión de servidor pueden hacerse a través de TCP/IP o sockets de dominio Unix, y puede ser encriptado SSL para la seguridad. PgAdmin es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas. Es un software libre publicado bajo la licencia BSD.

Características:

- Entradas SQL aleatorias.
- Pantallas de información y ayudas para bases de datos, tablas, índices, secuencias, vistas, programas de arranque, funciones y lenguajes.
- Preguntas y respuestas para configurar Usuarios, Grupos y Privilegios.
- Control de revisión con mejora de la generación de script.
- Configuración de las tablas de Microsoft MSysConf.
- Ayudas para importar y exportar datos.
- Ayuda para migrar Bases de datos.
- Informes predefinidos en bases de datos, tablas, índices, secuencias, lenguajes y vistas.

1.10 CONCLUSIONES PARCIALES

Después de realizado el estudio de las tendencias y las tecnologías actuales de los diferentes gestores de base de datos, así como las diferentes definiciones y términos necesarios para la realización del trabajo, se concluye con la necesidad de diseñar e implementar una base de datos para gestionar los conocimientos existentes sobre los Patrones en la UCI en la que:

- Se ha escogido el modelo de base de datos relacional.
- Como metodología de desarrollo de software RUP.
- Como lenguaje de modelado UML, para realizar el diagrama de clases persistentes.
- La herramienta CASE escogida para el diseño de la base de datos es el Erwin.
- El SGBD a utilizar es PostgreSQL utilizado para el desarrollo y gestión de la base de datos.
- Y para administrar la base de datos se utilizara PgAdmin III.

CAPÍTULO 2

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA

INTRODUCCIÓN

En este capítulo se realizará la descripción de la arquitectura de PostgreSQL, también se hará referencia a los requisitos funcionales que deberá cumplir la aplicación, así como los modelos lógicos y físicos de la base de datos y las descripciones de las tablas y sus atributos. Además se analizará varios aspectos relacionados con la seguridad de las bases de datos.

2.1 ARQUITECTURA DE POSTGRE

PostgreSQL utiliza un simple modelo cliente/servidor de "proceso por usuario". Una sesión de PostgreSQL consiste en los siguientes procesos Unix (programas) cooperando:

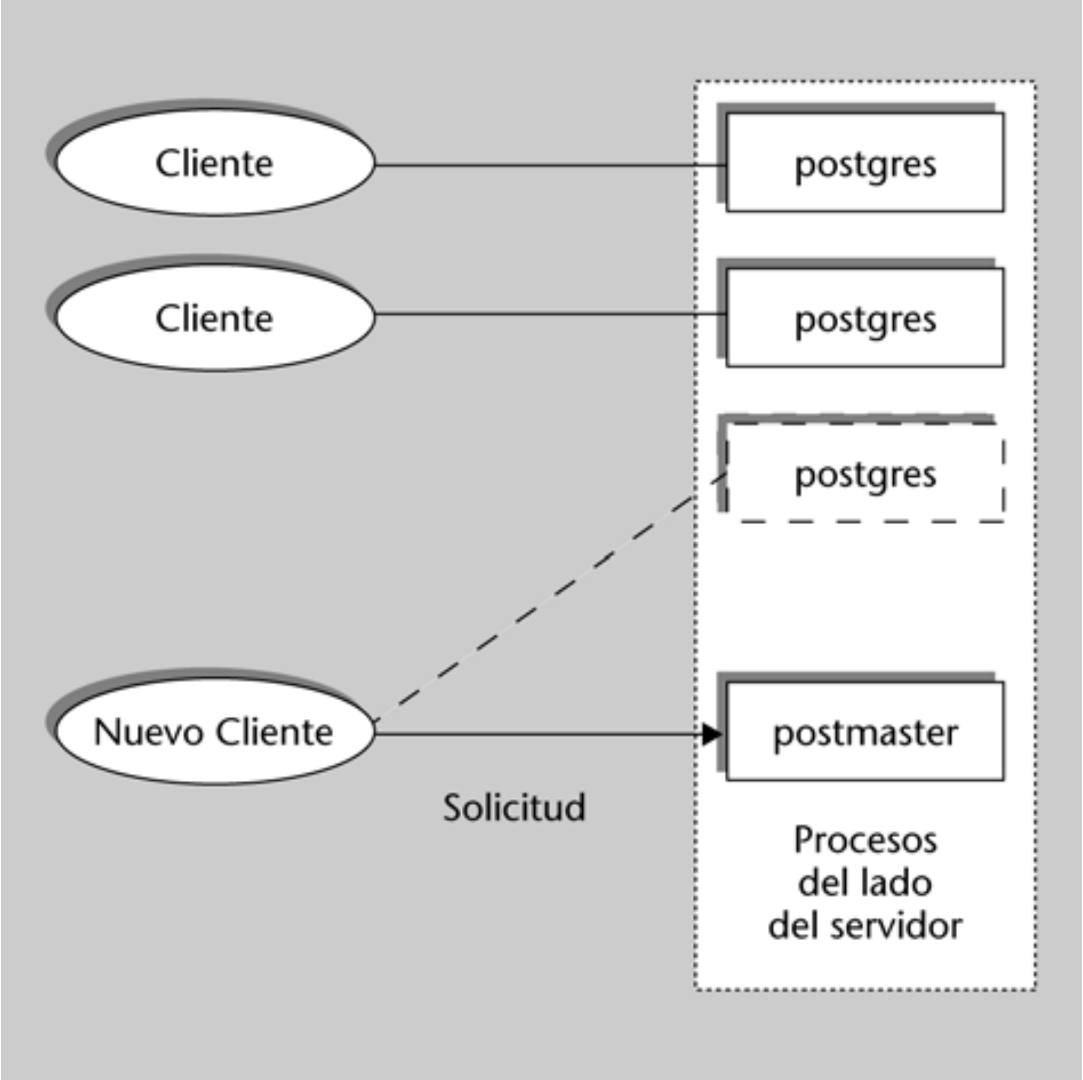
- Un proceso supervisor (postmaster)
- La aplicación de interface del usuario (por ejemplo, el programa psql)
- Uno o más procesos servidores de acceso a la base de datos (backend en inglés)

La arquitectura de PostgreSQL es de 3 niveles. Esta se corresponde suficientemente bien con un gran número de sistemas, aunque no se puede asegurar que cualquier SGBD se corresponda exactamente con ella.

La arquitectura de PostgreSQL se divide en los siguientes niveles generales: interno, lógico global y externo.

- El **nivel interno** es el más cercano al almacenamiento físico, o sea, es el relacionado con la forma en que los datos están realmente almacenados.
- El **nivel lógico** global es un nivel intermedio entre los dos anteriores.
- El **nivel externo** es el más cercano a los usuarios, o sea, es el relacionado con la forma en que los datos son vistos por cada usuario individualmente.

El siguiente gráfico muestra las entidades involucradas en el funcionamiento normal del gestor de bases de datos:



El programa servidor se llama PostgreSQL**.** Un proceso servidor *PostgreSQL* puede atender exclusivamente a un solo cliente; es decir, hacen falta tantos procesos servidor *PostgreSQL* como clientes haya. El proceso postmaster es el encargado de ejecutar un nuevo servidor para cada cliente que solicite una conexión.

Se llama sitio al equipo anfitrión (*host*) que almacena un conjunto de bases de datos PostgreSQL. En un *sitio* se ejecuta solamente un proceso *postmaster* y múltiples procesos *PostgreSQL*. Los clientes pueden ejecutarse en el mismo sitio o en equipos remotos conectados por TCP/IP

2.2 REQUISITOS FUNCIONALES

Un requisito funcional (RF) define el comportamiento interno del software, es decir, capacidades o condiciones que el sistema debe cumplir.

1RF. Gestionar la información de los patrones.

1.1 Debe permitir crear, modificar o eliminar un Patrón.

1.2 Debe validarse que al crearse un patrón se registre además su clasificación y tipo.

1.3 Debe dar la posibilidad de adicionar ejemplos de la aplicación de los patrones.

2RF. Gestionar los proyectos.

2.1 Debe permitir adicionar, modificar o eliminar los proyectos de la Universidad.

2.2 Al adicionar un proyecto habrá que especificarle los roles que contemplará.

3RF. Gestionar los usuarios

3.1 Debe permitir gestionar los datos de los usuarios de la aplicación, ya sea insertar, modificar o eliminar un usuario.

4RF. Gestionar Roles

4.1 Debe permitir crear, modificar o eliminar los roles existentes en la universidad.

5RF. Permitir consultar información sobre los patrones.

5.1 Debe permitir consultar la información sobre un patrón pudiendo obtener:

- Problema que resuelve.
- Patrones relacionados con él.
- Solución que ofrece.
- Contraindicaciones.
- Ejemplo de su uso.

5.2 Debe permitir dada una clasificación mostrar un listado de los patrones con dicha clasificación.

6RF. Obtener información sobre el uso de patrones en los proyectos.

6.1 Permitir conocer el uso de un patrón en los proyectos.

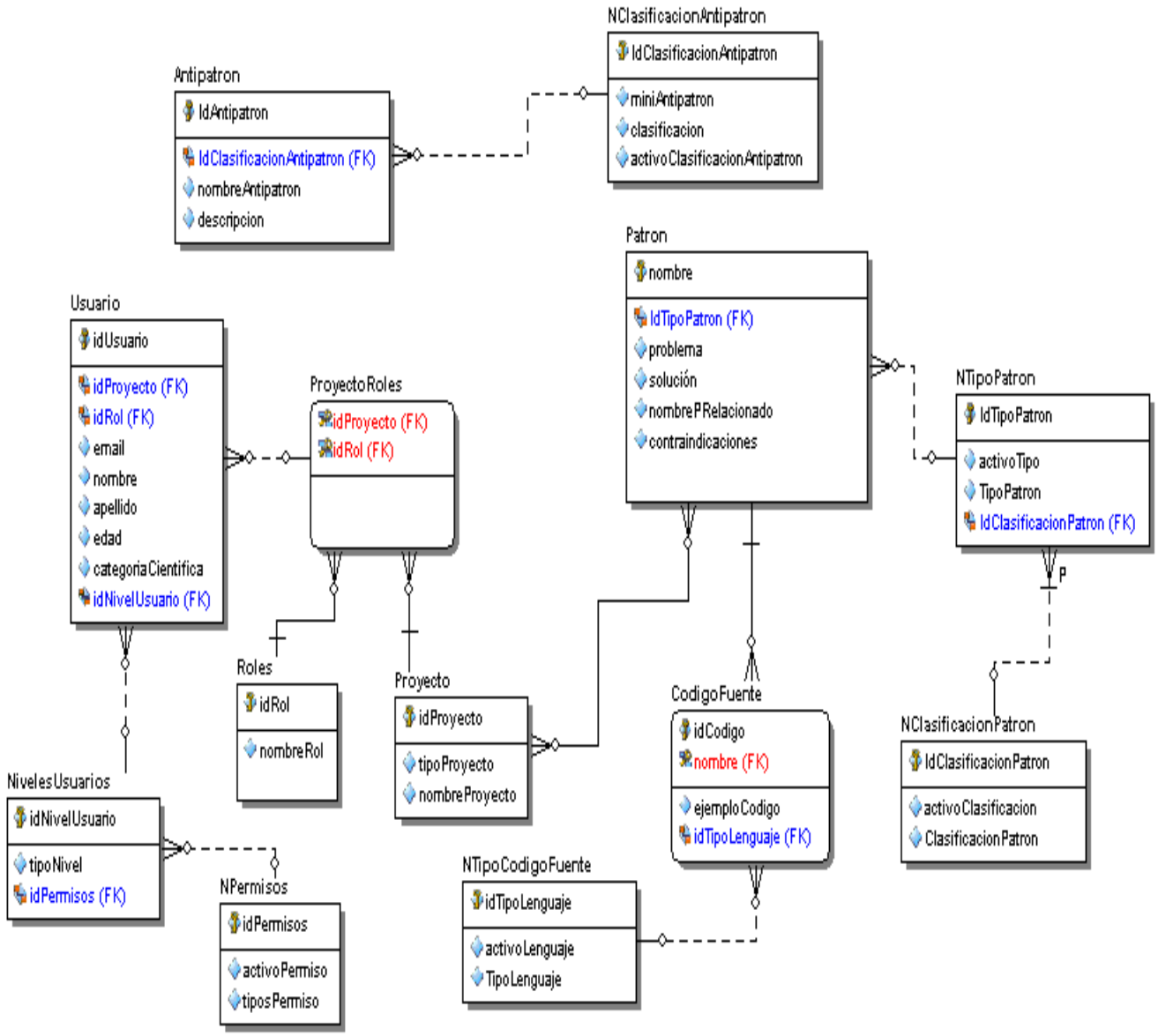
6.2 Permitir conocer el patrón más utilizado en los proyectos productivos de la universidad.

7RF. Permitir consultar la información sobre un anti patrón pudiendo obtener:

- Clasificación, Tipo, Nombre.
- Problemas en los que no se debe aplicar.

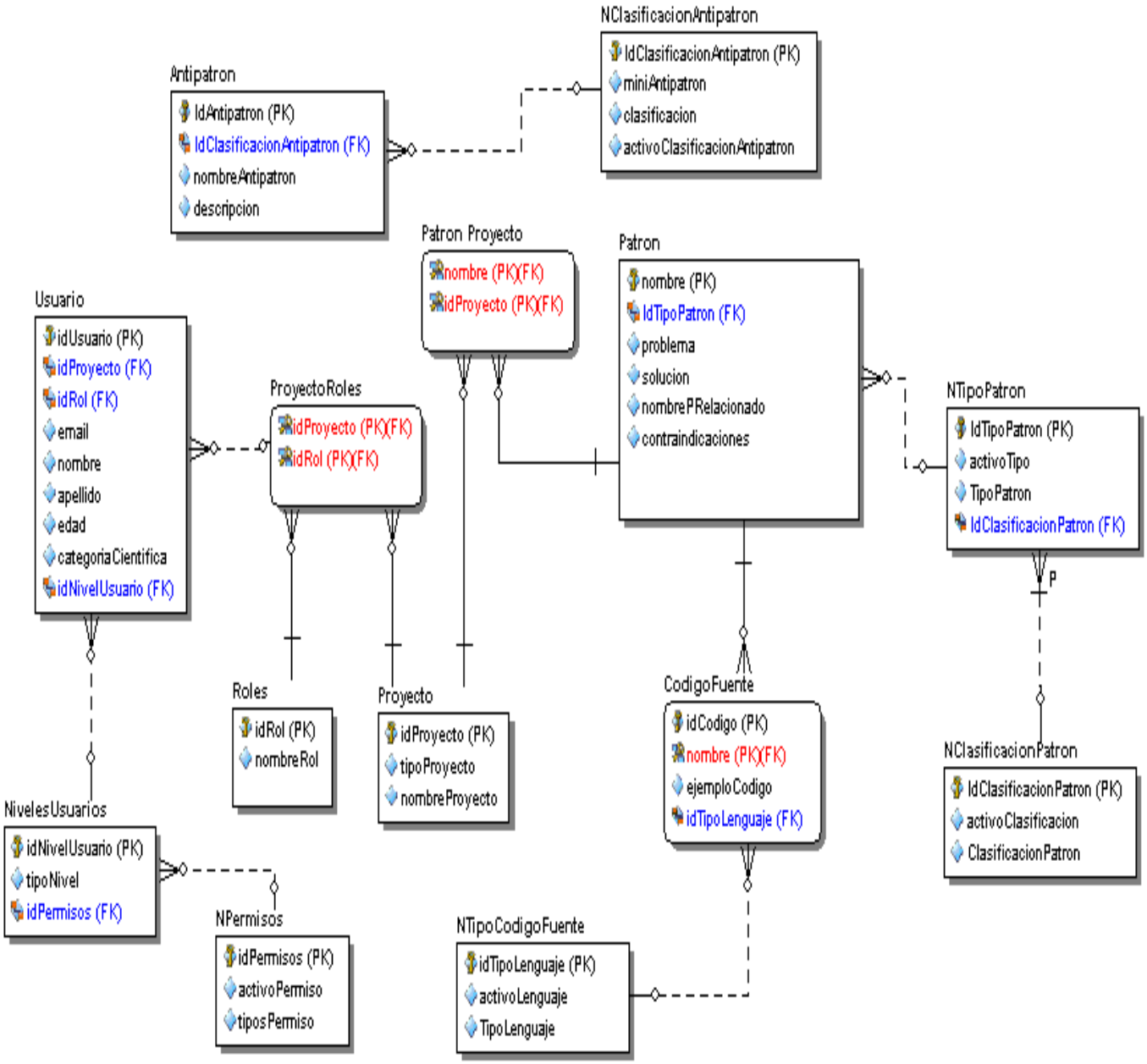
2.3 DIAGRAMA ENTIDAD-RELACIÓN

El modelo de entidad-relación es una herramienta que actualmente posee el mayor nivel de abstracción, ya que no tiene un nivel de implementación exacto, el objetivo de su utilización es comprender la naturaleza de los datos, de la información y su significado, por esto es eficiente para analizar lo que se convertirá en una Base de Datos. (SOFIA, y otros)



2.4 DIAGRAMA DE ESQUEMAS RELACIONALES

El Diagrama de Esquemas Relacionales está compuesto por un conjunto de relaciones e interrelaciones que forman la estructura necesaria para almacenar la información referente a Patrones.



El Diagrama de Esquemas Relacionales esta compuesto por un total de 12 relaciones, entre las que se pueden destacar como las más relevantes las relacionadas con patrón, en la que se recoge la información necesaria de los patrones como: clasificación, tipo, ejemplo de código fuente, etc. Además tiene una interrelación con Tipo de Patrón, esta interrelación nos permite conocer de que tipo es el patrón, y a su vez se relaciona con la tabla que contiene la clasificación de la misma. Además la tabla patrón contiene una relación con la tabla de códigos fuentes, en la que se mostrarán ejemplos de código fuentes de los patrones así como imágenes de las mismas, a su vez existente la relación de la tabla patrón con proyectos, facilitándose con ello el conocimiento de los patrones utilizados en los proyectos.

Las relaciones que se establece entre las tablas usuario con los proyectos consiste en que un usuario va a pertenecer a un proyecto pero a su vez cumplirá con un rol dentro del mismo y pertenecerá a un nivel de usuario con el cual se le asignaran los permisos de la base de datos de formas muy limitada y especifica. También se mostrará la información referente a los anti-patrones así como su relación con la tabla que contiene su clasificación y tipo.

2.5 DESCRIPCIÓN DEL DIAGRAMA DE ESQUEMAS RELACIONALES

Tabla 1. Descripción de la tabla usuario.

Nombre: Usuario				
Descripción: Guarda información referente a los usuarios.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idUsuario	char(10)	No	Identificador principal.

FK	idProyecto	Int(4)	No	Identificador del proyecto al que pertenece el usuario.
FK	idRol	Int(4)	No	Identificador del rol al que pertenece el usuario.
	Email	text	No	Dirección de correo.
	nombre	char(15)	No	Nombre del usuario.
	apellido	char(10)	No	Apellido del usuario.
	edad	char(10)	Sí	Edad del usuario.
	categoríaCientífica	text	No	Categoría científica que tiene.
FK	idNivelUsuario	char(10)	No	Identificador del nivel de usuario.

Tabla 2. Descripción de la tabla niveles de usuarios.

Nombre: NivelesUsuarios				
Descripción: En esta tabla se almacenan los niveles de usuario que existen.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idNivelUsuario	char(10)	No	Identificador principal.
	TipoNivel	varchar(15)	No	Tipo de nivel que tiene el

				usuario.
FK	idPermisos	char(10)	No	Identificador de los permisos del usuario.

Tabla 3. Descripción de la tabla de permisos.

Nombre: NPermisos				
Descripción: Nomenclador que guarda los permisos que se otorgan en dependencia del nivel del usuario.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idPermisos	char(10)	No	Identificador principal.
	tiposPermiso	varchar(35)	No	Tipos de permisos existentes.
	activoPermiso	boolean	No	Hace referencia si esta activo o no.

Tabla 4. Descripción de la tabla que relaciona los proyectos y los roles.

Nombre: ProyectoRoles.				
Descripción: Representa la relación existente entre los proyectos y los roles.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**

(PK)(FK)	idProyecto	Int(4)	No	Identificador del proyecto.
(PK)(FK)	idRol	Int(4)	No	Identificador del rol.

Tabla 5. Descripción de la tabla roles.

Nombre: Roles				
Descripción: Guarda la información referente a los roles.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idRol	Int(4)	No	Identificador principal.
	nombreRol	char(10)	No	Nombre del rol.

Tabla 6. Descripción de la tabla proyectos.

Nombre: Proyecto				
Descripción: Guarda toda la información referente a los proyectos.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idProyecto	Int(4)	No	Identificador principal.
	tipoProyecto	text	No	Tipo de proyecto.
	nombreProyecto	char(10)	No	Nombre del proyecto.

Tabla 7. Descripción de la tabla que relaciona patrones y proyectos.

Nombre: ProyectoPatrón				
Descripción: Relaciona los patrones con los proyectos en que se usan.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
(PK)(FK)	idProyecto	Int(4)	No	Identificador principal.
(PK)(FK)	nombre	char(20)	No	Identificador principal.

Tabla 8. Descripción de la tabla patrón.

Nombre: Patrón				
Descripción: Guarda toda la información referente a los patrones.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	nombre	char(20)	No	Nombre del patrón.
FK	IdTipoPatron	char(10)	No	Identificador tipo de patrones.
	problema	text	No	Problema que resuelve el patrón.
	solución	text	No	Solución que propone el patrón.
	nombrePRelacionados	char(20)	No	Nombre de patrones que se relacionan
	contraindicaciones	char(10)	Sí	Contraindicaciones del uso de este patrón.
FK	idCodigo	char(10)	No	Identificador del código.

Tabla 9. Descripción de la tabla código fuente.

Nombre: CodigoFuente				
Descripción: Ejemplo de código fuente del patrón.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idCodigo	char(10)	No	Identificador principal.
	EjemploCodigo	varchar(26)	No	Ejemplo del código.
FK	idTipoLenguaje	char(10)	No	Identificador del tipo de lenguaje.

Tabla 10. Descripción de la tabla tipo de lenguaje.

Nombre: NTipoCodigoFuente				
Descripción: Nomenclador que guarda el tipo de lenguaje en que esta implementado el código de ejemplo.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idTipolenguaje	char(10)	No	Identificador principal.
	activoLenguaje	boolean	No	Hace referencia si esta activo o no.
	TipoLenguaje	char(10)	No	Tipo de lenguaje.

Tabla 11. Descripción de la tabla clasificación.

Nombre: NClasificacionPatron				
Descripción: Nomenclador que guarda la clasificación del patrón.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	IdClasifcacionPatron	char(10)	No	Identificador principal.
	activoClasificacion	boolean	No	Hace referencia si esta activo o no.
	clasificaciónPatron	text	No	Clasificación del patrón.

Tabla 12. Descripción de la tabla tipo.

Nombre: NTipoPatron				
Descripción: Nomenclador que guarda la información de los tipos de patrones.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	idTipoPatron	char(10)	No	Identificador principal.
	activoTipo	boolean	No	Hace referencia si esta activo o no.
	TipoPatrón	char(10)	No	Tipo de patrón.
FK	idClasificaciónPatron	char(10)	No	Identificador de la clasificación.

Tabla 13. Descripción de la tabla anti-patrón.

Nombre: Antipatron				
Descripción: Guarda toda la información referente a los antipatrones.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	IdAntipatron	char(10)	No	Identificador principal.
FK	IdClasificacionAntipatron	Char(10)	No	Identificador de NClasificacion_antipatron
	NombreAntipatron	char(20)	No	Nombre del anti patrón
	descripcion	char(10)	No	Descripción del anti patrón

Tabla 14. Descripción de la tabla clasificación de anti patrón.

Nombre: NClasificacionAntipatron				
Descripción: Nomenclador que guarda la clasificación del anti patrón.				
Llave	**Atributo**	**Tipo**	**¿Acepta Nulos?**	**Descripción**
PK	IdClasificacionAntipatron	char(10)	No	Identificador principal.
	activoClasificacionAntipatron	boolean	No	Hace referencia si esta activo o no.
	miniAntipatron	boolean	No	Hace referencia a si es del tipo mini anti patrón o no.
	clasificacion	char(35)	No	Clasificación del anti -patrón.

2.6 SEGURIDAD DE LA BASE DE DATOS

En la actualidad las aplicaciones de bases de datos son unos de los puntos más importantes de los sistemas informáticos, es por eso que durante su desarrollo se deben tomar algunas medidas para garantizar la seguridad de la BD. Por lo cual una de las medidas de seguridad que se propone es que la conexión a la BD no se establezca con el usuario propietario de la BD o súper usuario (usuario con acceso total a la base de datos), ya que estos usuarios pueden, por ejemplo, ejecutar cualquier consulta según su deseo, modificando el esquema (ejemplo: eliminando tablas) o borrando su contenido completo. Para esto se deben crear diferentes usuarios de la base de datos para cada aspecto de la aplicación con derechos muy limitados sobre los objetos de la base de datos. Para garantizar esto en la BD se le otorgan privilegios estrictamente necesarios, evitando que el mismo usuario pueda interactuar con la base de datos en diferentes casos de uso. Esto quiere decir que si un intruso gana acceso a la base de datos usando una de éstas credenciales, él solo podrá realizar las operaciones de las que tiene privilegios, reduciendo de esta forma las posibilidades de ataques o errores por acceso en la BD de forma completa. Para garantizar esto se establece que los usuarios definidos lógica y físicamente trabajaran de forma paralela.

Además para garantizar la disponibilidad de la información que se encuentra almacenada en la base de datos, se creará una tarea programada del sistema, la cual permitirá hacer un respaldo (backup) automático de la base de datos cada 24 horas.

2.7 CONCLUSIONES PARCIALES

En este capítulo se realizó la descripción de la arquitectura de PostgreSQL, así como el diseño de los modelos lógicos y físicos de la base de datos y las descripciones de las tablas y sus atributos. Además se analizaron varios aspectos fundamentales que deberá cumplir la base de datos para garantizar la seguridad e integridad de los datos.

CAPÍTULO 3

Capítulo 3: Implementación y validación de la base de datos.

INTRODUCCIÓN

En este capítulo se describe la validación teórica y funcional del diseño realizado. Se realiza además un estudio y análisis de la Base de Datos de varios parámetros importantes a medir dentro de la misma a través de una lista de chequeo. Se exponen algunos aspectos que se tuvieron presentes para realizar el diseño de Base de Datos, tales como: la integridad de los datos, la normalización, la seguridad, etc. Además se realiza un análisis de las consultas más utilizadas así como su tiempo de respuesta.

3.1 VALIDACIÓN TEÓRICA DEL DISEÑO DE LA BASE DE DATOS.

3.1.1 INTEGRIDAD DE LOS DATOS.

El término **integridad de datos** se refiere a la corrección y completitud de los datos en una Base de Datos. Cuando los contenidos se modifican con sentencias *INSERT*, *DELETE* o *UPDATE*, la integridad de los datos almacenados puede perderse de muchas maneras diferentes sin poder garantizar la no contradicción entre los datos almacenados, de modo que en cualquier momento, los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Relacionándose así con la minimización de la redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia de los datos.

La semántica y la integridad de los datos son conceptos que en la Base de Datos suelen ir asociados, aunque no son idénticos. La integridad de los datos presenta varias restricciones que posibilitan la declaración y comprobación de condiciones para expresar la consistencia, corrección y exactitud de datos almacenados, éstas son: Integridad de entidad, de dominio, referencial, integridad de usuario.

- **Integridad de entidad**

  Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla; si no, la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El sistema administrador de la Base de Datos (DBMS) comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallará.

- **La integridad de dominio**

  La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones CHECK, o el intervalo de

valores posibles mediante restricciones FOREIGN KEY, restricciones CHECK, definiciones DEFAULT, definiciones NOT NULL y reglas.

- **Integridad referencial**

  La integridad referencial es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

  Asegura la integridad entre las llaves foráneas y primarias (relaciones padre/hijo). Existen cuatro actualizaciones de la base de datos que pueden corromper la integridad referencial:

  - ✓ La inserción de una fila hijo se produce cuando no coincide la llave foránea con la llave primaria del padre.
  - ✓ La actualización en la llave foránea de la fila hijo, donde se produce una actualización en la llave foránea de la fila hijo con una sentencia UPDATE y la misma no coincide con ninguna llave primaria.
  - ✓ La supresión de una fila padre, con la que, si una fila padre -que tiene uno o más hijos- se suprime, las filas hijos quedarán huérfanas.
  - ✓ La actualización de la llave primaria de una fila padre, donde si en una fila padre, que tiene uno o más hijos se actualiza su llave primaria, las filas hijos quedarán huérfanas.

  Para conseguir esa coherencia en la base de datos de Patrones, se precisó que no existieran referencias a valores inexistentes, se definió las opciones de borrado en las llaves foráneas, al cambiar el valor de una llave, todas las referencias a ella se cambian en consecuencia en toda la base de datos.

  Un ejemplo de esto lo constituye entre proyecto y roles pues al modificar el identificador del proyecto se modifican todas las tuplas relacionadas en la relación proyecto Roles.

- **Integridad definida por el usuario**

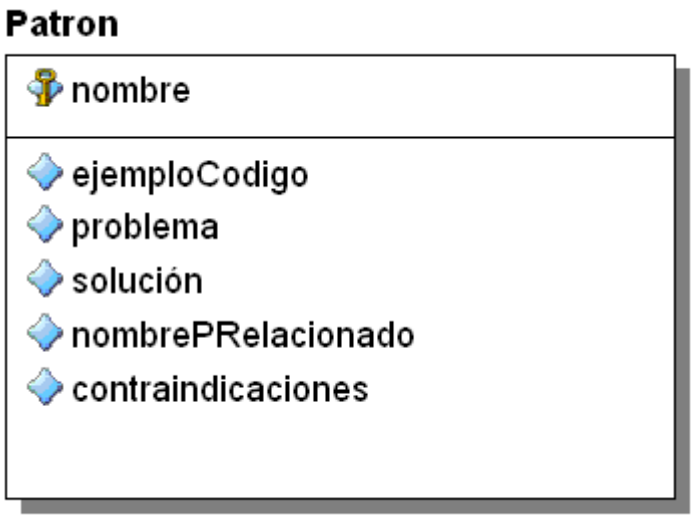
La integridad definida por el usuario permite definir reglas de empresa específicas que no pertenecen a ninguna otra categoría de integridad. Todas las categorías de integridad admiten la integridad definida por el usuario. Esto incluye todas las restricciones de nivel de columna y nivel de tabla en CREATE TABLE, procedimientos almacenados y desencadenadores. Ejemplo de ello es que cuando se inserte un patrón se tenga que especificar el tipo de patrón así como su clasificación y ejemplo de código fuente.

3.1.2 NORMALIZACIÓN.

Cuando se va a realizar el diseño de una base de datos relacional, se tiene en cuenta que las tablas que la componen cumplan con las reglas de normalización, que no es más que el proceso que permite eliminar la redundancia de los datos y evitar los problemas al insertar, eliminar y actualizar los datos, es decir, optimizar el trabajo con la información almacenada. En este proceso existen varios niveles de normalización, pero los tres primeros son los más usados. Las reglas de normalización se relacionan entre ellas de forma dependiente, lo que indica que para que una base de datos se encuentre en una determinada forma normal, primeramente tiene que cumplir con las reglas de los niveles inferiores de normalización. Esto quiere decir que una base de datos está en 2da Forma Normal si previamente cumple con las reglas de normalización del 1er nivel. Cada regla que se cumple aumenta el grado de normalización del esquema de relación. Si una regla no se cumple, el esquema se debe descomponer en varios esquemas de relación que si la cumplan por separado. Se dice que una base de datos se encuentra en determinada forma normal si cumple con las restricciones correspondientes a dicha forma normal.

Para que un esquema relacional este en 1ra Forma Normal se tiene que asegurar que cada tupla contiene solamente atributos con valores, es decir, no existan campos multievaluados y además no existen campos compuestos ni derivados. En el caso de la base de datos para la gestión del conocimiento sobre los patrones en la primera iteración no se encontraba en la primera forma normal, pues el atributo *ejemploCodigo* era multievaluado.

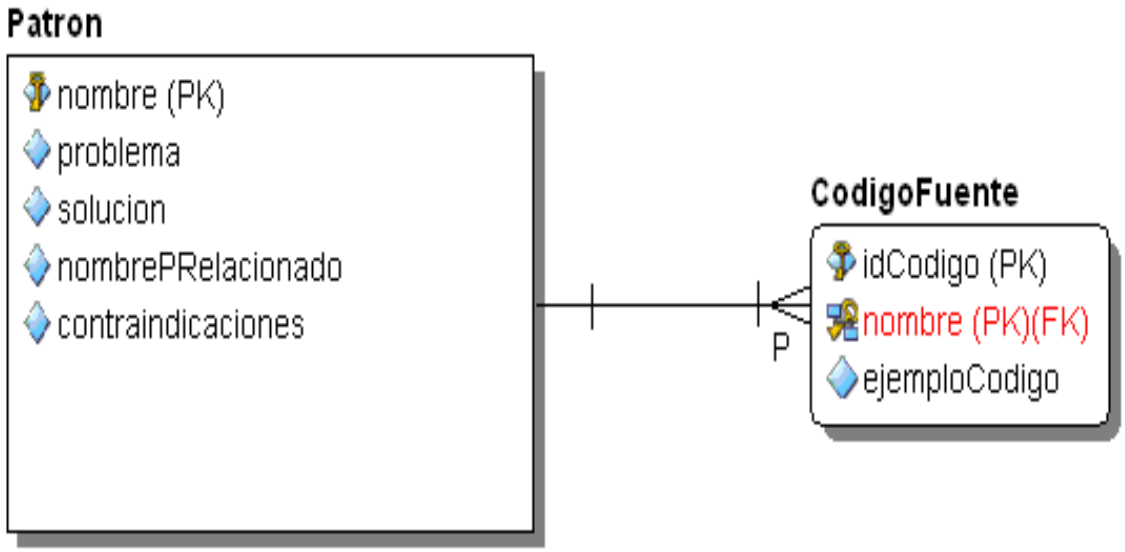
Antes de normalizar



Nombre ->ejemploCodigo, problema, solución, nombrePRelacionado, contraindicaciones.

En una segunda iteración para la aplicación de la 1ra Forma Normal se creó la tabla *CodigoFuente* para eliminar con ella la existencia del atributo multievaluado.

Después de normalizar



Nombre ->problema, solución, nombrePRelacionado, contraindicaciones.
Nombre, idCodigo -> ejemploCodigo.

Llaves candidatas y primarias

Nombre
idCodigo

También se cumple que el esquema de relación está en 2da Forma Normal, porque primeramente se encuentran en 1ra Forma Normal, y además todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria, o sea no existen dependencias parciales de la llave primaria. Finalmente se puede plantear que los esquemas de relación se encuentran en 3ra Forma Normal, ya que se encuentran en 2da Forma Normal, y además no existen dependencias transitivas entre llaves candidatas y atributos no primos.

3.2 VALIDACIÓN DE LA BASE DE DATOS POR LISTA DE CHEQUEO.

No.	Aspectos a comprobar	Descripción	Cumplido
1.	Las clases persistentes están mapeadas en la estructura de la base de datos	Se realizo el mapeo de las clases persistentes de forma correcta.	1ra iteración
2.	Las relaciones de muchos a muchos tienen una tabla de intersección.	Entre las tablas patrón y proyecto se creó una tabla *PatronProyecto* con las llaves primarias de ambas.	1ra iteración
3.	Las llaves primarias han sido definidas en cada tabla, al no ser que no haya razón de su	Todas las tablas cuentan con su llave primaria.	1ra iteración

	existencia.		
4.	La distribución de los datos ha sido planeada.	En el modelado de la BD se realizó la distribución de los datos.	1ra iteración
5.	Las restricciones de los datos han sido definidas.	A todos los datos se le especificó el tipo de dato que aceptar.	1ra iteración
6.	Los procedimientos almacenados han sido definidos.	Se definieron todas las funciones.	1ra iteración

Diseño

No.	**Aspectos a comprobar**	**Descripción**	**Cumplido**
Modelo de datos			
1.	Ya está confeccionado el diagrama de Entidad-Relación.	Se realizó el modelado del diagrama entidad relación con la herramienta Erwin.	1ra iteración
2.	Ya están confeccionados los diccionarios de datos.	Se generó el diccionario de datos.	2da iteración
3.	Los mismos presentan la descripción de las entidades, atributos y relaciones de la BD.	Los reportes generados en el diccionario de datos presentan todas las descripciones de entidades, atributos y relaciones.	2da iteración
Diseño de Seguridad y Auditoria			
Para la garantía y protección de la Información se encuentra definido:			
4.	Tipos de usuario.	Se establecieron los tipos de usuarios: administrador, usuario e invitado.	1ra iteración

5.	Tipos de acceso.	Se especificaron los accesos que tendrá cada usuario con respecto a la base de datos así como a sus consultas. Ejemplo: lectura o lectura-escritura.	1ra iteración
6.	Niveles de usuario.	Para cada tipo de usuario se asignó un nivel de usuario.	1ra iteración
7.	Facultades para cada nivel.	A cada nivel de usuario se le asignaron permisos específicos y limitados	1ra iteración

Operaciones Básicas

No.	Aspectos a comprobar	Descripción	Cumplido
1.	Realizan las operaciones básicas: insertar, eliminar, modificar, y que otras consultas se han desarrollado.	Se elaboraron todas las consultas y funciones necesarias para cumplir con los requerimientos funcionales.	1ra iteración

Tiempo de respuesta a la consulta

No.	Aspectos a comprobar	Descripción	Cumplido
1.	El tiempo de respuesta de la consulta es breve	Se cuenta con una carta de aceptación por parte del cliente que el tiempo de respuesta es satisface las necesidades.	2da iteración

Integridad de los datos

No.	Aspectos a comprobar	Descripción	Cumplido
1.	Se modifica y se eliminan los datos en las tablas tal como se especifica en la funcionalidad.	Se cumple con lo especificado en los requisitos.	1ra iteración
2.	¿Cuándo se guarda un determinado conjunto en la base de datos, se guarda cada valor completo? (En otras palabras, no se produce el truncamiento de cadenas y el redondeo de los valores numéricos).	Se almacena en la base de datos los valores tal y como los introduce el usuario, es decir no se le hacen variaciones a los datos entrados.	1ra iteración
3.	Los valores predeterminados se guardan en la base de datos cuando no se ha especificado la entrada del usuario.	Ejemplo de ello se encuentra en que los valores booleanos se le asigno true por defecto.	1ra iteración

Recuperación

No.	Aspectos a comprobar	Descripción	Cumplido
1.	¿Escogen el método de backup para realizar recuperaciones satisfactorias?	El método que se escogió fue el iterativo incremental.	2da iteración

Seguridad de cuentas

No.	Aspectos a comprobar	Descripción	Cumplido

1.	¿Alguna cuenta de la BD está ligada con una cuenta de sistema operativo?	Ninguna cuenta se encuentra ligada a una cuenta del sistema operativo.	1ra iteración
2.	¿Los passwords son fijados cuando se crea un usuario?	El único usuario que no tiene contraseña es el invitado es por esto que sus permisos son muy limitados.	1ra iteración
3.	¿Los passwords pueden ser alterados por el DBA (o cualquier usuario que tenga el rol DBA) o por el usuario mismo?	Los passwords sólo pueden ser cambiados por los usuarios.	1ra iteración

3.3 VALIDACIÓN FUNCIONAL.

Para el llenado voluminoso e inteligente de la BD se utilizó como herramienta EMS Data Generator For PostgreSQL 2010, siendo de gran utilidad para generar los datos de pruebas a varias tablas a la misma vez. Esta herramienta da la posibilidad de escoger las tablas a las cuales se quiere llenar. Se realizó un registro de 10 000 tuplas para la tabla patrón, 5 000 para usuario, así como también otras tantas tuplas para las restantes. Se definió para cada campo el rango de valores admitidos en dependencia de la cantidad de tuplas a insertar, cumpliendo en todo momento las pruebas de carga intensiva y escalabilidad propuesto.

3.4 TÉCNICAS DE OPTIMIZACIÓN.

3.4.1 TÉCNICA DE OPTIMIZACIÓN DE CONSULTAS.

En Bases de Datos, una consulta es el método para acceder a los datos; a través de las consultas se puede modificar, borrar, mostrar y agregar datos en una Base de Datos. Para esto se utiliza un lenguaje de manipulación de datos (DML – Data Manipulation Language). Las consultas se pueden expresar de diferentes formas alcanzándose el mismo propósito para lo cual fue diseñado, de ahí que algunas sean más óptimas que otras en dependencia de la lógica empleada y el plan de ejecución.

Cuando hablamos de **optimización de consultas** nos referimos a mejorar los tiempos de respuestas en un sistema de gestión de Base de Datos, pues la optimización es el proceso de encontrar el mejor plan de ejecución con exclusiva eficiencia.

Para el análisis de la optimización de consultas de la BD propuesta se tuvo en cuenta algunos criterios y se definieron algunos parámetros para un adecuado rendimiento:

- Se desglosaron las consultas complejas en varias subconsultas simples.
- Se realizaron las selecciones tan rápido como fue posible.
- Detectaron subconsultas con resultados equivalentes que se reutilizaron a lo largo de una misma consulta.

3.4.2 TÉCNICA DE OPTIMIZACIÓN DE DISEÑO.

Teniendo en cuenta que dentro de los requisitos no funcionales de hardware se especifica que el software trabajará sobre una máquina de 512MB de RAM que es la capacidad promedio de las máquinas de la universidad, se hace necesario realizar el diseño de la base de datos de la manera más eficaz posible. Puesto que no es lo mismo tratar de solucionar o mejorar rendimiento sobre algo no muy correctamente diseñado,

que sobre un diseño adecuado, pues con este se facilitaría el trabajo posterior con la base de datos.

Con el fin de realizar un diseño correcto se normalizó hasta la tercera forma normal, permitiendo que no exista duplicidad en los datos y contribuyendo a un almacenamiento más sólido y eficiente. Además se empleó la técnica de particionamiento de dato, para descentralizar los datos de la tabla patrón, debido a que esta es la tabla que más solicitudes tendrá. Con esto se garantizará que la base de dato sea más liviana y flexible. Logrando con la aplicación de esta técnica evitar que en un momento determinado se devuelva algún campo vacio en las consultas, ya que no siempre se va a tener un ejemplo del código fuente para cada patrón y que el tiempo de respuestas de las solicitudes que se realizan a la Base de Datos sea menor.

3.5 VALIDACIÓN POR TIEMPO DE RESPUESTA.

Con el fin de realizar un análisis de factibilidad y rapidez de la Base de Datos que respondan a las funcionalidades, se analizaron los tiempos de respuestas de las principales consultas, así como la media de tiempo por respuesta y el tiempo total de las consultas, realizando una conexión al servidor de BD desde una PC que se encontraba en otra subred y con una velocidad de conexión de 100 Mbps.

#1.

```
begin
select * into aux from "Patron" where
"Patron".nombre = $1;
if found then
return false;
else
SELECT  "NClasificacionPatron"."IdClasificacionPatron" INTO var FROM "NClasificacionPatron"
WHERE "NClasificacionPatron"."ClasificacionPatron"=$7;
if found then
SELECT "NTipoPatron"."IdTipoPatron" INTO vr FROM "NTipoPatron"
WHERE "NTipoPatron"."TipoPatron"=$6 AND "NTipoPatron"."IdClasificacionPatron"=var;
if found then
SELECT "NTipoCodigoFuente"."idTipoLenguaje" INTO vrr FROM "NTipoCodigoFuente"
```

```
WHERE "NTipoCodigoFuente"."TipoLenguaje"=$9;

INSERT into "CodigoFuente" ("ejemploCodigo","idTipoLenguaje") VALUES($8, vrr);
SELECT "CodigoFuente"."idCodigo" INTO r FROM "CodigoFuente"
ORDER BY "CodigoFuente"."idCodigo" DESC LIMIT 1;

INSERT into "Patron" ("nombre","idCodigo","IdTipoPatron","problema","solucion","nombrePRelacionados","contraindicaciones") VALUES($1,r,vr,$2,$3,$4,$5);
  return true;
  else
  return false;
  end if;
  else
  return false;
  end if;
 end if;
END;
```

El tiempo de respuesta es de 0.11 ms.

#2.

```
BEGIN
select* into var from "Patron" where "Patron".nombre = $1;
if found then
delete from "Patron" where "Patron".nombre = $1;
        return true;
   else
       return false;
end if;
END;
```

El tiempo de respuesta es de 0.09 ms.

#3.

```
BEGIN
SELECT "Patron"."idCodigo" INTO var From "Patron" WHERE "Patron".nombre = $1;
if found then
 UPDATE "Patron"  set  problema = $2,
                       solucion = $3,
                       "nombrePRelacionados" = $4,
```

```
                    contraindicaciones = $5

WHERE "Patron".nombre = $1;
UPDATE "CodigoFuente" set "ejemploCodigo" = $6
WHERE   "CodigoFuente"."idCodigo"  = var;
Return true;
else
Return false;
end if;
END;
```

El tiempo de respuesta es de 0.11 ms.

#4.

```
BEGIN
  select * into rr from "Proyecto" where
"Proyecto"."idProyecto" = $1;
  if found then
     UPDATE "Proyecto"  set  "nombreProyecto" = $2,
                    "tipoProyecto" = $3
      where "Proyecto"."idProyecto" = $1;
 return true;
  else
  return false;
  end if;
END;
```

El tiempo de respuesta es de 0.11 ms.

#5.

```
BEGIN
  select * into vrr from "Roles" where
"Roles"."idRol" = $1;
  if found then
  UPDATE "Roles"  set  "nombreRol" = $2
      where "Roles"."idRol" = $1;
 return true;
  else
  return false;
  end if;
END;
```

El tiempo de respuesta es de 0.08 ms.

#6.

```
BEGIN
SELECT *  FROM  "Usuario" into var
WHERE  "Usuario"."idUsuario" = $6;
if found then
SELECT "Roles"."idRol" INTO aux FROM "Roles"
WHERE "Roles"."nombreRol"=$4;
if found then
select "Proyecto"."idProyecto" INTO vrr FROM "Proyecto"
WHERE  "Proyecto"."nombreProyecto" = $5;
if found then
SELECT * FROM "ProyectoRoles" into pr
WHERE "ProyectoRoles"."idProyecto" = vrr AND
  "ProyectoRoles"."idRol" = aux;
if found then
UPDATE "Usuario"  set  email =$1,
                    edad = $2,
                    "categoriaCientifica" = $3,
                    "idRol" = aux,
                    "idProyecto" = vrr
where  "Usuario"."idUsuario" = $6;
 return true;
 else
return false;
end if;
else
return false;
end if;
else
return false;
end if;
else
return false;
end if;
END;
```

El tiempo de respuesta es de 0.17 ms.

#7.

```
BEGIN
SELECT "Roles"."idRol" INTO aux FROM "Roles"
WHERE "Roles"."nombreRol"=$3;
if found then
INSERT into "Proyecto" ("tipoProyecto","nombreProyecto") VALUES($1,$2);
SELECT "Proyecto"."idProyecto" INTO vrr FROM "Proyecto"
ORDER BY "Proyecto"."idProyecto" DESC LIMIT 1;
INSERT into "ProyectoRoles"("idProyecto","idRol") VALUES(vrr,aux);
 return true;
else
return false;
end if;
END;
```

El tiempo de respuesta es de 0.13 ms.

#8.

```
BEGIN
select * into vrr from "Roles" where
"Roles"."nombreRol" = $1;
 if found then
 return false;
 else
 insert into "Roles"("nombreRol") values($1);
 return true;
 end if;
 END;
```

El tiempo de respuesta es de 0.08 ms.

#9.

```
BEGIN
SELECT "Roles"."idRol" INTO aux FROM "Roles"
WHERE "Roles"."nombreRol"=$6;
if found then
select "Proyecto"."idProyecto" INTO vrr FROM "Proyecto"
WHERE  "Proyecto"."nombreProyecto" = $7;

if found then
```

```
SELECT * FROM "ProyectoRoles" into pr
WHERE "ProyectoRoles"."idProyecto" = vrr AND
  "ProyectoRoles"."idRol" = aux;
if found then
select "NivelesUsuarios"."idNivelUsuario" INTO rr FROM "NivelesUsuarios"
WHERE  "NivelesUsuarios"."tipoNivel" = 'usuarios';
INSERT into "Usuario"
("idProyecto","idRol","email","nombre","apellido","edad","categoriaCientifica","idNivelUsuario" )
VALUES(vrr,aux,$3,$1,$2,$4,$5,rr);
 return true;
 else
 return false;
 end if;
else
return false;
end if;
else
return false;
end if;
END;
```

El tiempo de respuesta es de 0.09 ms.

#10.

```
SELECT
  "Proyecto"."nombreProyecto"
FROM
  "Patron"
  INNER JOIN "PatronProyecto" ON ("Patron".nombre = "PatronProyecto".nombre)
  INNER JOIN "Proyecto" ON ("PatronProyecto"."idProyecto" = "Proyecto"."idProyecto")
WHERE
  "Patron".nombre = $1;
```

El tiempo de respuesta es de 0.09 ms.

#11.

```
BEGIN
select* into var from "Proyecto" where "Proyecto"."nombreProyecto" = $1;
if found then
delete from "Proyecto" where "Proyecto"."nombreProyecto" = $1;
        return true;
   else
```

```
      return false;
end if;
END;
```

El tiempo de respuesta es de 0.11 ms.

#12.

```
BEGIN
select * into vrr from "Roles" where
"Roles"."nombreRol" = $1;
 if found then
 delete from "Roles" where "Roles"."nombreRol" = $1;
       return true;
else
 return false;
 end if;
END;
```

El tiempo de respuesta es de 0.09 ms.

#13.

```
SELECT
  "Patron".nombre
FROM
  "Patron"
  INNER JOIN "NTipoPatron" ON ("Patron"."IdTipoPatron" = "NTipoPatron"."IdTipoPatron")
  INNER JOIN "NClasificacionPatron" ON ("NTipoPatron"."IdClasificacionPatron" =
"NClasificacionPatron"."IdClasificacionPatron")
WHERE
  "NClasificacionPatron"."ClasificacionPatron" = $1;
```

El tiempo de respuesta es de 0.11 ms.

El análisis del tiempo de respuesta por consulta se realizó luego de insertar información en la base de datos, apoyándose en la herramienta de llenado de la BD EMS Data Generator For PostgreSQL 2010, la cual creó un registro de 10 000 tuplas para la tabla patrón, 5 000 para usuario, 1 000 para la tabla proyecto y así sucesivamente para las restantes tablas. Evidenciándose con ello la validez de la base de datos en el trabajo con grandes volúmenes de datos, alcanzando un total de tiempo de respuesta de 1.37 segundos, y una media de 0.11ms. Mostrándose luego de este análisis la efectividad de la BD, así como su eficiencia en el trabajo con datos a través de una subred distante y con una velocidad de conexión no óptima. Ver acta de conformidad del cliente anexo 1.

3.6 CONCLUSIONES PARCIALES.

En el capítulo 3 se realizó un análisis de varios aspectos de gran importancia para poder demostrar que el modelado de la BD así como su implementación fueron las más óptimas. Se concluyó con un análisis de la consistencia de los datos, así como la integridad de los mismos, la normalización y las distintas formas de seguridad que se pueden adoptar para el control de la información.

CONCLUSIONES GENERALES

En la actualidad el almacenamiento de los datos y conocimientos referentes a patrones en nuestra universidad resulta un gran problema, debido a que no se realiza de forma óptima o se hace de forma manual, por lo que no existe calidad y uniformidad en el trabajo con las misma, por tales razones surge este trabajo, en el cual se lleva a cabo el diseño y modelación de una base de datos que ayuda al almacenamiento y consulta de los conocimientos de patrones.

Para el buen desarrollo del mismo se utilizó la herramienta multiplataforma y pertenecientes a la familia del software libre, de gran versativilidad, dadas las características y condiciones en el mundo de la informática en la actualidad PostgreSQL como SGBD. Además se usó la herramienta CASE ERWIN STUDIO por las facilidades de uso que brinda y las potencialidades que ofrece, entre los que se encuentran:

- El modelado de datos tanto en la etapa lógica como en la física facilitando el trabajo ya que esta función la realiza sincronizando estas dos etapas, lo cual posibilita que cualquier cambio que se realice en una de las dos etapas se pueda realizar de forma automática en la otra.
- Permite el indexado de tabla.
- El direccionamiento de datos.

Se debe agregar que se obtuvieron los diseños de una Base de Datos relacional en Tercera Forma Normal, y se validó la integridad y la redundancia de información a través de una lista de chequeo, lográndose finalmente cumplir con el objetivo propuesto, pues cuenta con todos los requisitos y se logra guardar la información.

BIBLIOGRAFÍA

Andrés, María Mercedes Marqués. 2001. Apuntes de Ficheros y Bases de Datos. *Bases de Datos.* [En línea] 12 de 02 de 2001. [Citado el: 25 de 11 de 2009.] http://www3.uji.es/~mmarques/f47/apun/apun.html.

Miguel A. Guevara López, Mario Rodríguez Rodríguez y Norma González Pestano. DIAG, un sistema experto para el diagnóstico de anomalías craneofaciales. *Rev Cubana Invest Bioméd v.16 n.2 Ciudad de la Habana jul.-dic. 1997.* [En línea] [Citado el: 7 de Febrero de 2010.] http://scielo.sld.cu/scielo.php?pid=S0864-03001997000200003&script=sci_arttext.

Pecos, Daniel. 2008. PostGreSQL vs. MySQL. [En línea] 03 de Febrero de 2008. [Citado el: 28 de Enero de 2010.] http://www.netpecos.org/docs/mysql_postgres/index.html.

Rodriguez, Rubén. Bases De Datos (La Historia). [En línea] [Citado el: 20 de enero de 2010.] http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim.

Tijerina, Martha Esthela Ruiz. *Bases de Datos.* [En línea] [Citado el: 25 de Enero de 2010.] http://www.elrinconcito.com/articulos/BaseDatos/BasesDatos.htm#2.

SOFIA, Albert Anibal Osiris y HEREDIA, Raúl. *Diagrama Entidad-Relación.* [En línea] [Citado el: 10 de 2 de 2010.] http://170.210.92.6/osofia/$Bdd/Practicas/DERw.pdf.

Marco A. Guevara , Cesar R. Flores Nasario. ALLFunsion Erwin Data Modeler. [En línea] [Citado el: 5 de 12 de 09.] http://www.scribd.com/doc/3075477/Manual-de-Usuario-de-Erwin .

EMS Database Management Solutions, Inc. EMS Data Generator 2010 for PostgreSQL . [En línea] [Citado el: 16 de 2 de 10.] http://www.sharewareconnection.com/ems-data-generator-2005-for-postgresql.htm .

André, María Mercedes Marqués. Arquitectura de los sistemas de bases de datos. [En línea] [Citado el: 15 de 1 de 10.] http://www3.uji.es/~mmarques/f47/apun/node33.html .

CA technologies. AllFusion® ERwin Data Modeler. [En línea] [Citado el: 16 de 1 de 10.] http://www.ca.com/es/products/product.aspx?id=260 .

Espinoza, Humberto. 05. PostgreSQL. [En línea] 30 de 12 de 05. [Citado el: 18 de 11 de 09.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf .

MASTERMAGAZINE. Definición de Normalización. [En línea] [Citado el: 20 de 11 de 09.] http://www.mastermagazine.info/termino/6105.php .

PostgreSQL en Chile. 09. PostgreSQL. [En línea] 12 de 6 de 09. [Citado el: 3 de 12 de 09.] http://www.postgresql.cl/ .

Sanchez, María A. Mendoza. 06. *Informatizate.* [En línea] 7 de 7 de 06. [Citado el: 21 de 11 de 09.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.htm

A., Ernesto Quiñones. 09. *SERVICIOS DE ALTA DISPONIBILIDAD DE BASES DE DATOS CON POSTGRESQL. [En línea] 18 de 6 de 09. [Citado el: 2 de 11 de 09.] http://postgresql.org.pe/articles/pgsql_alta_disponibilidad.pdf.*

JuanPLG. 09. TodoSQL . *Pautas para Optimización de Performace en SQL Server. [En línea] 7 de 1 de 09. [Citado el: 12 de 4 de 10.] http://www.todosql.com/pautas-optimizacion-performance-sql-server.*

Menéndez, Dr. Eugenio Santos. 08. *Ejemplos de Optimización de Consultas. [En línea] 1 de 12 de 08. [Citado el: 12 de 4 de 10.] http://bd.eui.upm.es/DYOBD/Ejemplos_optimizacion_consultas_09.pdf.*

Perez, Manuel Bollain. 09. *Asignatura Diseño y Optimización de Bases de Datos. [En línea] 22 de 11 de 09. [Citado el: 5 de 4 de 10.] http://bd.eui.upm.es/DYOBD/bd22.html.*

Trabajos citados

Andrés, María Mercedes Marqués. 2001. Apuntes de Ficheros y Bases de Datos. *Bases de Datos.* [En línea] 12 de 02 de 2001. [Citado el: 25 de 11 de 2009.] http://www3.uji.es/~mmarques/f47/apun/apun.html.

Miguel A. Guevara López, Mario Rodríguez Rodríguez y Norma González Pestano. DIAG, un sistema experto para el diagnóstico de anomalías craneofaciales. *Rev Cubana Invest Bioméd v.16 n.2 Ciudad de la Habana jul.-dic. 1997.* [En línea] [Citado el: 7 de Febrero de 2010.] http://scielo.sld.cu/scielo.php?pid=S0864-03001997000200003&script=sci_arttext.

Pecos, Daniel. 2008. PostGreSQL vs. MySQL. [En línea] 03 de Febrero de 2008. [Citado el: 28 de Enero de 2010.] http://www.netpecos.org/docs/mysql_postgres/index.html.

Rodriguez, Rubén. Bases De Datos (La Historia). [En línea] [Citado el: 20 de enero de 2010.] http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim.

Tijerina, Martha Esthela Ruiz. *Bases de Datos.* [En línea] [Citado el: 25 de Enero de 2010.] http://www.elrinconcito.com/articulos/BaseDatos/BasesDatos.htm#2.

SOFIA, Albert Anibal Osiris y HEREDIA, Raúl. *Diagrama Entidad-Relación.* [En línea] [Citado el: 10 de 2 de 2010.] http://170.210.92.6/osofia/$Bdd/Practicas/DERw.pdf.

ANEXOS

Anexo 1

UCI

Ciudad Habana, 31 de mayo del 2010
"Año del 52 Aniversario del triunfo de la Revolución"

Acta de Conformidad

Yo Anisbert Suárez Batista una vez analizado todas las consultas y funciones con las cuales se le dio cumplimiento a los requerimientos funcionales, planteo que estoy de acuerdo con los tiempos de respuesta de las mismas y los considero óptimos pues dentro de los requerimientos no funcionales de Hardware está que el software trabajará sobre una máquina de 512MB de RAM que es la capacidad promedio de las máquinas de la universidad, con lo cual se demuestra que la Base de Dato responderá de manera eficiente a las consultas que se realicen en ella.

Firma: Anisbert Suárez Batista