Universidad de las Ciencias Informáticas

Facultad 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO

DE INGENIERO EN CIENCIAS INFORMÁTICAS

Diseño e implementación del modelo de datos para el Sistema Integral de Confiabilidad Operacional (SIC)

AUTOR:

Wilder Hernández González

TUTORES:

Ing. Lannie Octavio Herrera Pérez

Ing. Yorji Pérez Hernández

CO-TUTOR:

Ing. Yuniel Sardiñas García

La Habana, Cuba

Junio, 2012

Quien tenga una computadora dispone de todos los conocimientos publicados. La privilegiada memoria de la máquina le pertenece también a él.

Las ideas nacen de los conocimientos y de los valores éticos. Una parte importante del problema estaría resuelta tecnológicamente, la otra hay que cultivarla sin descanso o de lo contrario se impondrán los instintos más primarios.

La tarea que los graduados de la UCI tienen por delante es grandiosa. Espero que la cumplan, y la cumplirán.

Fidel Castro.

DEDICATORIA

A mi mamá por ser muy especial en mi vida, por su ejemplo y su afán de verme convertido en lo que soy hoy.

A mi papá por guiarme por el camino correcto y ser partícipe de mi vida en cada momento.

A mi hermana por quererme tanto y darme lo mejor de sí.

A mi familia en sentido general.

AGRADECIMIENTOS

A mi mamá y mi papá por ser los mejores padres del mundo y siempre confiar en mí.

A mi hermana por tenerme como ejemplo en su vida.

A mi novia por quererme tanto y siempre estar ahí en todo momento.

A mis tutores por guiarme en el transcurso de la tesis.

A mis amigos por estar siempre presentes en los buenos y malos momentos.

A Fidel y a la Revolución por darme la oportunidad de convertirme en quien soy.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente.

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

Autor:

_____

Wilder Hernández González

Tutores:

_____ _____

Ing. Lannie Octavio Herrera Pérez Ing. Yorji Pérez Hernández

Co-Tutor

_____

Ing. Yuniel Sardiñas García

DATOS DE CONTACTO

Tutor: Ing. Lannie Octavio Herrera Pérez

Centro de Informática Industrial (CEDIN), Departamento de Dirección de Proyectos, Facultad 5, Universidad de las Ciencias Informáticas (UCI).

Correo: lherrera@uci.cu

Tutor: Ing. Yorji Pérez Hernández

Centro de Informática Industrial (CEDIN), Departamento de Construcción de Componentes, Facultad 5, Universidad de las Ciencias Informáticas (UCI).

Correo: yphernandez@uci.cu

Co-Tutor: Ing. Yuniel Sardiñas García

Centro de Informática Industrial (CEDIN), Departamento de Construcción de Componentes, Facultad 5, Universidad de las Ciencias Informáticas (UCI).

Correo: yunielsg@uci.cu

RESUMEN

Las bases de datos son herramientas que permiten el almacenamiento, ordenamiento y clasificación de la información para su fácil manipulación. El desarrollo y manejo de la información recopilada en las bases de datos se realiza a través de gestores de bases de datos, aplicaciones muy utilizadas en proyectos de gran envergadura. En el Centro de Informática Industrial, de la Universidad de las Ciencias Informáticas, se desarrolla el proyecto Sistema Integral de Confiabilidad Operacional, el cual cuenta con una base de datos incompleta, que no garantiza la correcta gestión de la información. La presente investigación tiene como objetivo principal diseñar e implementar un modelo de base de datos que contribuya a la integridad, organización e independencia de la información en el Sistema Integral de Confiabilidad Operacional. Para ello se analizan los referentes teóricos actuales, relacionados con el diseño de base de datos relacionales.

Como resultado del presente trabajo se realizó el diseño de una base de datos que soluciona los problemas existentes y a partir de este diseño se implementó una capa de acceso que suministra y almacena toda información para el nivel de negocio. La base de datos generada fue validada a través de las pruebas de rendimiento permitiendo comprobar el logro del objetivo propuesto.

Palabras clave: base de datos, capa de acceso a datos, diseño de base de datos y gestor de base de datos.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

Desde las primeras civilizaciones ha existido la necesidad de contar con sistemas de información donde centrar los conocimientos. Los datos se recopilaban, estructuraban, centralizaban y almacenaban convenientemente con el objetivo de poder recuperar esa información en algún momento determinado para hacer uso de ella o de algún dato asociado sin necesidad de volverlos a recopilar. Nunca había tenido tanto valor la información como en nuestros días, provocando que esta se haya convertido en un eslabón fundamental para el desarrollo de cualquier sociedad.

A raíz del desarrollo tecnológico surge la necesidad de almacenar y gestionar grandes volúmenes de datos, para su posterior consulta, producidos por las nuevas industrias y que se capturaban o generaban mediante aplicaciones. Esta necesidad dio paso al surgimiento de las bases de datos, una de las herramientas más ampliamente difundidas en la actual sociedad de la información y cuyo objetivo es almacenar, ordenar y clasificar los datos para su fácil manejo.

Con el avance de las Tecnologías de la Información y las Comunicaciones (TIC), las herramientas de bases de datos han experimentado un auge extraordinario, lo que ha traído consigo rapidez, efectividad y menos costos en los procesos de grandes flujos de información para satisfacer las necesidades a la hora de optimizar servicios y productos. En la actualidad se le presta gran importancia a la gestión de la información a través de los gestores de bases de datos, que son aplicaciones que permiten manejar la información de modo sencillo y que presentan excelentes funcionalidades para el desarrollo y el manejo de las bases datos.

Cuba no se ha quedado al margen respecto a la utilización de estas tecnologías, en el Centro de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas (UCI), se desarrolla el proyecto Sistema Integral de Confiabilidad Operacional (SIC), el cual tiene entre sus ramas de investigación la Confiabilidad Operacional, que no es más que "(…) una serie de procesos de mejoramiento continuo, que incorporan en forma sistemática, avanzadas herramientas de diagnóstico, técnicas de análisis y nuevas tecnologías, para optimizar la gestión, planeación, ejecución y control de la producción industrial. La Confiabilidad Operacional lleva implícita la capacidad de una instalación (procesos, tecnología, gente), para cumplir su función o el propósito que se espera de ella, dentro de sus límites de diseño y bajo un específico contexto operacional" (1).

El Sistema Integral de Confiabilidad (SIC) cuenta con una base de datos incompleta, que no garantiza la correcta gestión de la información. El acceso a la misma es limitado, lo que dificulta la conexión simultánea de múltiples usuarios. No permite el almacenamiento de grandes volúmenes de datos y tiene limitaciones en el procesamiento de las búsquedas. Dicha base de datos no satisface las necesidades del cliente ni las expectativas del negocio.

A partir de lo antes expuesto se formula el siguiente **Problema Científico:** ¿Cómo gestionar el almacenamiento de la información en el proyecto Sistema Integral de Confiabilidad Operacional?

Por tanto se define como **Objeto de Estudio:** el diseño de base de datos.

Y para darle solución al problema científico propuesto se plantea el **Objetivo General** siguiente: diseñar e implementar un modelo de base de datos que contribuya a la integridad, organización e independencia de la información en el Sistema Integral de Confiabilidad Operacional.

Todo lo anterior define como **Campo de Acción:** el diseño de base de datos relacionales para sistemas de Confiabilidad Operacional.

Se tiene como **idea a defender** que con el diseño e implementación de la base de datos del Sistema Integral de Confiabilidad Operacional (SIC) se logrará una alta integridad, organización e independencia de los datos que contribuirá a la gestión de los mismos.

Para dar cumplimiento a los objetivos planteados, se trazaron una serie de **Tareas de Investigación:**

- Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente sobre el tema.
- Selección de las herramientas libres existentes para dar cumplimiento a los requerimientos del cliente.
- Confección del modelo conceptual de la base de datos.
- Confección del modelo lógico de la base de datos.
- Confección el diseño físico de la base de datos.
- Elaboración del script de datos perteneciente a la estructura del modelo de datos diseñado.
- Realización de pruebas de rendimiento a la base de datos para validarla funcionalmente.

Los **métodos científicos** utilizados en la realización de este trabajo con el fin de estudiar las características del objeto de investigación son los siguientes:

Métodos Teóricos:

Analítico-Sintético: Con el objetivo de realizar el análisis de las tendencias y tecnologías actuales de bases de datos a nivel mundial. Este método facilita el estudio y análisis sobre la arquitectura y normalización de la base de datos definida para el Sistema Integral de Confiabilidad Operacional.

Análisis Histórico-Lógico: Debido a que posibilita estudiar cómo han evolucionado y se han desarrollado las bases de datos desde su surgimiento hasta nuestros días, así como sus herramientas y modelos.

Modelación: Utilizado para realizar fundamentalmente la modelación lógica de la base de datos.

Métodos Empíricos:

Consulta de todo tipo de fuente de información: Con el objetivo de precisar y actualizar los elementos del diseño teórico-metodológico, y la elaboración del marco teórico de la investigación.

Realización de pruebas: Pues permiten acreditar la confiabilidad y validez de los resultados obtenidos.

Consultas de especialistas: Con el propósito de recibir los criterios de validación sobre la aplicabilidad y utilización de lo realizado.

El presente trabajo consta de una introducción, tres capítulos, conclusiones generales, referencias bibliográficas y por último, los anexos. A continuación se detalla brevemente cada una de las diferentes temáticas que se abordan en cada capítulo.

Capítulo I. Describe el marco teórico de la investigación y un análisis del estado del arte. Aborda temas como: las definiciones, características, fases del diseño y modelos de las bases de datos. Se tratan conceptos y características referentes a distintos gestores de base de datos. Además se realiza el estudio de varias herramientas empleadas para administrar, modelar y realizar pruebas a la base de datos.

Capítulo II. Se fundamenta la selección de las herramientas, tecnologías y modelo de base de datos a utilizar para el desarrollo de la siguiente investigación. Se plantea la descripción de la arquitectura a utilizar en el proyecto y la selección y argumentación de los requisitos no funcionales del sistema propuesto. También se confeccionan el diagrama de clases persistentes y el diagrama entidad relación de la base de datos, con el fin de generar las clases del dominio.

Capítulo III. Se realiza una validación funcional de la solución propuesta. Se exponen aspectos como la integridad, seguridad y redundancia de la información. Además se tratan aspectos referentes a las pruebas de volumen y carga.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se hace referencia a los principales aspectos y definiciones que contribuyen a una mejor comprensión de los términos que se emplean a lo largo del trabajo de diploma. Se abordan aspectos relacionados con las bases de datos, tales como su definición, características, arquitectura, fases del diseño y sus modelos. Se describen los gestores que se utilizan para el manejo de las bases de datos, así como sus principales características. Además se realiza un estudio sobre las distintas herramientas para administrar, modelar y realizar pruebas a la base de datos.

1.1. Base de Datos

“Una base de datos es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción han de ser únicas estando almacenadas junto a los mismos. Por último, los tratamientos que sufran estos datos tendrán que conservar la integridad y seguridad de éstos” (2).

“Un sistema de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones” (3).

Las bases de datos se componen de una o más tablas en las que se almacena información. Cada tabla tiene uno o más atributos y tuplas. Los atributos guardan una parte de la información sobre cada elemento que se quiera guardar en la tabla.

1.2. Características de las Bases de Datos

Entre las principales características de los sistemas de base de datos podemos mencionar (4):

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.

- Acceso a través de lenguajes de programación estándar.

1.3. Arquitectura de las Bases de Datos

La arquitectura propuesta por el Grupo de Estudio en Sistemas de Administración de Bases de Datos de ANSI/SPARC se divide en tres niveles, conocidos como interno, lógico global y externo. (Figura 1)

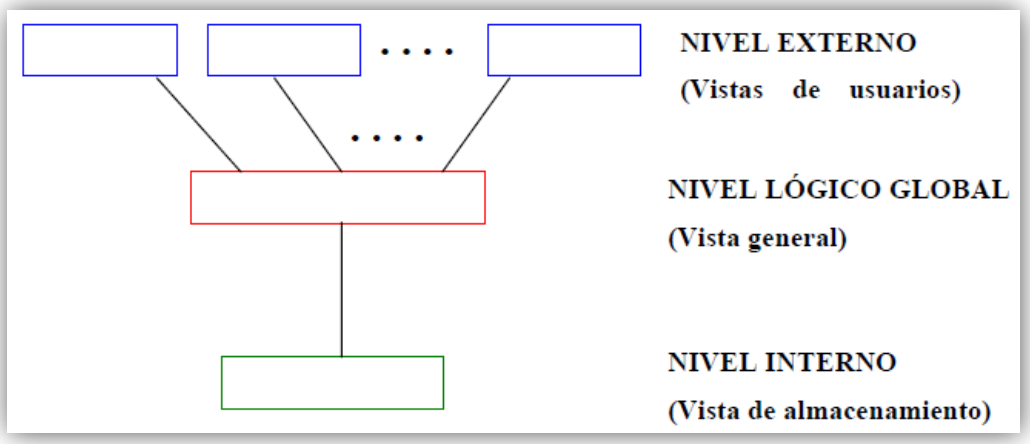

Figura 1. Niveles de la Arquitectura de Base de datos

- El nivel interno (también conocido como el nivel físico) es el que tiene que ver con la forma en que los datos están almacenados físicamente. “Es el nivel más bajo de abstracción, y define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso” (5).

- El nivel externo (también conocido como el nivel lógico de usuario) es el que tiene que ver con la forma en que los usuarios individuales ven los datos. “Es el nivel de mayor abstracción. A este nivel corresponden las diferentes vistas parciales que tienen de la base de datos los diferentes usuarios. En cierto modo, es la parte del modelo conceptual a la que tienen acceso” (5).

- “El nivel lógico global es un nivel intermedio entre los dos anteriores” (6). Es el nivel medio de abstracción. Se trata de la representación de los datos realizada por la organización, que recoge las vistas parciales de los requerimientos de los diferentes usuarios y las aplicaciones posibles. Se configura como visión organizativa total, e incluye la definición de datos y las relaciones entre ellos.

1.4. Fases del diseño de las Bases de Datos

El diseño de una base de datos es un proceso complejo que abarca decisiones en distintos niveles. La complejidad se controla mejor si se descompone el problema en problemas más pequeños y se resuelve cada uno de estos de manera independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

1.4.1. Diseño Conceptual

"El diseño conceptual se refiere a la etapa donde se debe construir un esquema de la información del entorno, independientemente del hardware disponible, del gestor de base de datos a utilizar o cualquier otra consideración física" (7).

En la construcción del esquema se utiliza la información que se encuentra en la especificación de los requisitos de usuario y es donde el diseñador descubre el significado de los datos del entorno y encuentra 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 y el uso de los datos a través de las áreas de aplicación. "El esquema conceptual es una fuente de información para el diseño lógico de la base de datos" (8).

1.4.2. Diseño Lógico

"El diseño lógico es el proceso de construir un esquema de la información, basándose en un modelo de base de datos específico independiente del Sistema de Gestión de Base de Datos (SGBD) y de cualquier otra consideración física" (7), o sea, el esquema conceptual es transformado en un esquema lógico que utiliza las estructuras de datos del modelo de base de datos en el que se basa el SGBD a utilizar, como puede ser el modelo relacional o el modelo orientado a objetos por solo citar dos ejemplos.

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.

1.4.3. Diseño Físico

"El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria, donde las estructuras de almacenamiento y los métodos de

acceso garanticen un acceso eficiente a los datos. En esta etapa se debe haber decidido cuál es el SGBD a utilizar, ya que el esquema físico se adapta a él" (7).

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.

1.5. Modelos de Bases de Datos

"Un modelo de datos es básicamente una descripción de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores" (9). Los modelos de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos.

Un modelo de base de datos es un conjunto de ideas lógicas utilizadas para representar la estructura de datos y las relaciones entre ellos dentro de la base de datos. Estos modelos se pueden agrupar en dos categorías: modelos conceptuales y modelos de ejecución.

- "El modelo conceptual se enfoca en la naturaleza lógica de la representación de datos. Por consiguiente, este modelo está comprometido con lo representado en la base de datos, y no en cómo está representado. Los modelos conceptuales incluyen el modelo de Entidad-Relación (E-R) y el modelo orientado a objetos" (10).

- "En contraste con el modelo conceptual, un modelo de ejecución hace énfasis en cómo los datos están representados en la base de datos o en cómo se ejecutan las estructuras de datos para representar lo que está modelado. Los modelos de ejecución incluyen el modelo de base de datos jerárquico, el de base de datos de red, el modelo de base de datos orientada a objetos y el modelo de base de datos relacional" (10).

1.5.1. Modelo Jerárquico

"Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos" (11). Entre sus características se encuentra el hecho de ser bastante rígido, ocasionando que una vez realizado el diseño de una base de datos, resulta complejo cambiarla y se debe tener un conocimiento

muy amplio de la manera en la que se han almacenado los datos para poder recuperarlos de una forma más efectiva.

Las bases de datos jerárquicas permiten crear estructuras estables y de gran rendimiento, y son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

1.5.2. Modelo de Datos de Red

Es un modelo ligeramente distinto del jerárquico pues se modifica el concepto de un nodo, permitiendo que el mismo tenga varios padres. Al ofrecer una solución eficiente al problema de redundancia de datos, es una gran mejora con respecto al modelo jerárquico.

Las representaciones lógicas basadas en este tipo de modelos, en muchas ocasiones limitan el cambio que el crecimiento de la base de datos exige, hasta tal punto que las representaciones lógicas de los datos pueden variar afectando a programas que los utilizan.

1.5.3. Modelo de Base de Datos Relacional

“El modelo de base de datos relacional es uno de los modelos matemáticos más importantes y actuales para la representación de las bases de datos. Se basa en la teoría matemática de las relaciones, suministrándose por ello una fundamentación teórica que permite aplicar todos los resultados de dicha teoría a problemas como el diseño de sub-lenguajes de datos y otros” (6).

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

La estructura básica del modelo de base de datos relacional es la relación o tablas. Todos los elementos de la base de datos se representan en forma de tabla o relación cuyo contenido varía con el tiempo. Una relación o tabla se representa gráficamente como una estructura rectangular, formada por filas o columnas. Cada columna almacena información sobre una propiedad determinada de la tabla o relación. A estas columnas también se les denomina atributos. Cada fila de la tabla se le denomina tupla y almacena los valores que toma cada uno de los atributos, para cada ocurrencia de la relación.

El lenguaje más utilizado para construir las consultas a bases de datos relacionales es Structured Query Language (SQL), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

1.5.4. Modelo Orientado a Objetos

El modelo orientado a objetos fue desarrollado con el objetivo de compensar las deficiencias del modelo relacional en cuanto a la construcción de consultas complejas, y estructuras de datos sin tener que dividirlas en una estructura relacional de dos dimensiones.

"Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos" (11):

- 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. En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes.

Este modelo trata de almacenar en la base de datos los objetos completos, o sea, el estado y el comportamiento. Combina las mejores cualidades de los archivos planos, las bases de datos jerárquicas y relacionales. Las bases de datos orientadas a objetos ofrecen un mejor rendimiento en cuanto a aplicaciones o clases con estructuras complejas de datos y representan la siguiente etapa en la evolución de las bases de datos.

1.6. Sistema de Gestión de Base de Datos

"Los Sistemas de Gestión de Base de Datos (en inglés Data Base Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan" (4). Se compone de un lenguaje de definición de datos (DDL, por sus siglas en inglés), un lenguaje de manipulación de datos (DML, por sus siglas en inglés) y un lenguaje de control de datos (DCL, por sus siglas en inglés).

1.6.1. Objetivos

Entre los objetivos que deben satisfacer los SGBD se encuentran (13):

- Diseño y utilización orientada al usuario: los datos y aplicaciones deben ser accesibles por los usuarios de la manera más amigable posible. Para ello, los SGBD se basan en un modelo de datos teórico coherente, y proporcionan lenguajes que permiten definir la estructura de la base de datos y acometer la generación, mantenimiento y acceso a los datos en los términos que resulten apropiados.

- Centralización: los datos deben gestionarse de forma centralizada e independiente de aplicaciones. Para satisfacer este objetivo el SGBD proporciona una serie de utilidades que facilitan la administración de ese fondo común. Adicionalmente, aparece la figura del administrador de base de datos como responsable de las tareas que permiten mantener la disponibilidad de ese fondo común.

- Evitar la redundancia y gestionar la concurrencia: como un SGBD provee un fondo de datos común que puede ser compartido por varias aplicaciones, no es preciso, por tanto, duplicar datos. Sin embargo, puesto que varias aplicaciones pueden acceder a datos iguales al mismo tiempo, el SGBD debe disponer de los mecanismos adecuados para gestionar esas concurrencias.

- Mantener la integridad semántica de los datos: un SGBD debe proveer de mecanismos para evitar la alteración deliberada de información por un usuario no autorizado y la alteración accidental de datos debido a un fallo del sistema. La introducción de datos erróneos por parte de un operador humano no puede ser evitado en su totalidad, pero un sistema en sí puede proporcionar los mecanismos para que se logre establecer restricciones a priori acerca de qué tipo de datos no están permitidos en la base de datos. Estos mecanismos, que se conocen como reglas de integridad y reglas del negocio, permiten desestimar la introducción de datos que vulneren esas restricciones.

- Mantener la seguridad: un SGBD debe proveer mecanismos de identificación que permiten definir qué usuario accede a qué recurso.

1.6.2. Ventajas

La utilización por parte de una organización de un SGBD que satisfaga los objetivos mencionados como soporte para la gestión de su información proporciona los siguientes beneficios (13):

- Organizar la información mediante una estructura de datos común, accesible y reutilizable por diferentes usuarios y aplicaciones.
- Realizar un control centralizado de dicha información, lo que permite aumentar la integridad de los datos frente a caídas del sistema, prevenir ciertos tipos de inconsistencias, incrementar la fiabilidad y la disponibilidad de la información, eliminar la redundancia de los datos y establecer directivas de seguridad sobre el acceso a los datos.
- Mejorar el rendimiento del procesamiento de los datos mediante la utilización de estrategias de acceso y arquitecturas de hardware/software optimizadas.
- Se pueden priorizar las necesidades de acceso por parte de las aplicaciones adoptando diferentes criterios de asignación de recursos de procesamiento conforme a las necesidades.

1.6.3. Gestores de Bases de Datos

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

“PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando” (14).

Sus características técnicas lo convierten en uno de los SGBD más potentes y robustos del mercado. Desde el inicio de su desarrollo las características que más se han tenido en cuenta son la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Características más importantes de PostgreSQL (14):

- Es una base de datos 100% ACID.
- Integridad referencial.
- Tablespaces (Espacios de tablas).
- Transacciones Anidadas.
- Replicación asíncrona.
- Punto de recuperación (PITR, por sus siglas en inglés).
- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- Acceso encriptado vía SSL.
- Completa documentación.
- Disponible para Linux y Windows.

Entre las principales ventajas que brinda PostgreSQL se encuentran (15) :

- A pesar de que la velocidad de respuesta pueda parecer deficiente en bases de datos pequeñas, esa velocidad se mantiene al aumentar el tamaño de la base de datos, cosa que no sucede con otros programas, que se enlentecen brutalmente.

- Ahorros considerables de costos de operación: PostgreSQL ha sido diseñado para tener un mantenimiento y ajuste menor que los productos de proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

- Extensible: El código fuente está disponible de forma gratuita para todos los que necesiten extender o personalizar el programa.

- Diseñado para ambientes de alto volumen: Utilizando una estrategia de almacenamiento de filas llamada MVCC, consigue mejor respuesta en grandes volúmenes. Además, MVCC permite a los accesos de solo lectura continuar leyendo datos consistentes durante la actualización de registros, permitiendo copias de seguridad en caliente

- Herramientas gráficas de diseño y administración de bases de datos.

- Soporta los tipos de datos, cláusulas, funciones y comandos de tipo estándar SQL92/SQL99 y extendidos propios de PostgreSQL.

- Puede operar sobre distintas plataformas, incluyendo Linux, Windows, Unix, Solaris y MacOS X.

- Buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas.

- Buena escalabilidad ya que es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

Oracle

"Oracle es una potente herramienta basada en la arquitectura Cliente/Servidor para la gestión de Bases de Datos Relacionales desarrollada por Oracle Corporation. Ofrece una interfaz intuitiva basada en el explorador, que es capaz de administrar las bases de datos, crear tablas, vistas y otros objetos de bases de datos, importar, exportar y visualizar datos de tablas, ejecutar scripts de SQL y generar informes. Además, soporta transacciones, es estable, escalable y multiplataforma" (16).

"Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia. Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente. Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos. Tiene buen rendimiento y hace buen uso de los recursos. Posee un rico diccionario de datos. Brinda soporte a la mayoría de los lenguajes de programación. Es un sistema multiplataforma, disponible en Windows, Linux y Unix. Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal" (17).

El mayor inconveniente que tiene el uso de Oracle es quizás su precio, pues incluso las licencias de Personal Oracle son excesivamente caras.

MySQL

"Microsoft SQL Server. Es un sistema para la gestión de bases de datos creado por Microsoft, el mismo se basa en el modelo relacional. Utiliza como lenguajes de consulta T-SQL y ANSI SQL" (18).

Dentro de sus características fundamentales se encuentran (18):

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red solo acceden a la información.
- Además permite administrar información de otros servidores de datos.

1.7. Grails

Grails es un framework libre construido sobre el lenguaje de programación Groovy para desarrollar aplicaciones web. Está diseñado sobre las bases de los principios que constituyen paradigmas del desarrollo ágil y hace alusión a que cada aspecto de la aplicación existe en un sólo lugar, no es necesario modificar múltiples capas de aplicación para realizar cambios. Debido a esto en su diseño Grails sigue el patrón Modelo-Vista-Controlador (MVC) que consiste en hacer una división clara entre objetos del dominio y la presentación.

Está basado en tecnologías maduras y establecidas, tales como Hibernate (para mapeo objeto-relacional), Spring (para inyección de dependencias), SiteMesh (para las plantillas), Quartz (para planificación de tareas), y JavaServer Pages (para el nivel de visión), como es mostrado en la siguiente figura la cual representa la arquitectura interna de Grails.

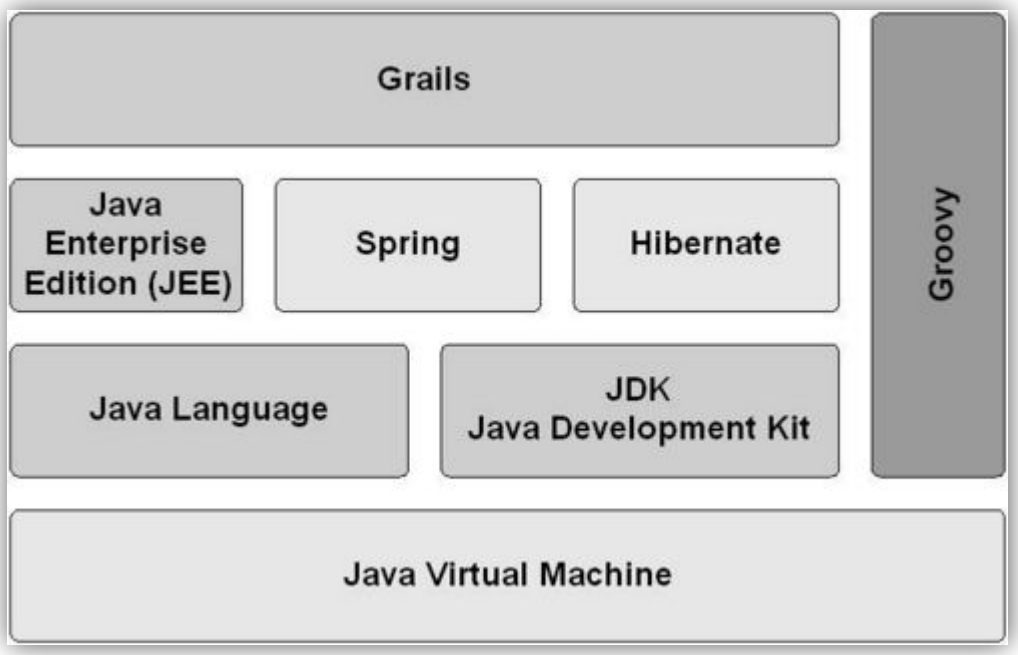

Figura 2. Arquitectura de Grails

Grails brinda una serie de ventajas que facilitan el trabajo con la base de datos. Entre ellas se encuentran que ofrece el manejo automático de las relaciones entre las entidades y el mapeo de persistencia automático para las clases del dominio del problema. Para el acceso a datos brinda scaffolding, permitiendo un desarrollo ágil de operaciones CRUD (Create, Read, Update, Delete). Introduce elementos de lenguaje específicos de dominio, de tal forma que la variable estática constrain permite las validaciones de los campos de un objeto de dominio y posibilita la gestión de errores de un objeto de dominio de una manera mucho más elegante que en otros frameworks.

1.8. Herramientas CASE

Embarcadero ER/Studio

ER/Studio es una herramienta de modelado de datos que brinda productividad en el análisis, diseño, creación y mantenimiento de aplicaciones de base de datos. Brinda un completo soporte al ciclo de vida de la base de datos para el avance y la reversión de la ingeniería. Cuenta con una avanzada comparación y combinación bidireccional para los modelos y estructuras de base de datos. Brinda la posibilidad de utilizar el modelado para producir esquemas XML para asegurar los beneficios del modelado cuando se usa con aplicaciones. Provee datos profesionales con la habilidad de documentar el flujo de datos a través de la organización para garantizar la seguridad e integración. Los modelos y reportes son publicados en distintos

formatos incluyendo HTML, XML Schema, DTD Output, y ER/Studio Viewer. ER/Studio es compatible con una gran cantidad de SGBD entre los que se encuentran (19):

- MySQL® 3.x, 4.x, 5.x
- NCR® Teradata® V2R4, V2R5, V2R6
- Oracle® 7.3.x, 8.x, 9i, 10g, 11g
- PostgreSQL 8.x
- Sybase® Adaptive Server® Enterprise (ASE) 11.9.2, 12.x, 12.5, 15.0
- Microsoft® Access 2.0, 95, 97,2000
- Microsoft SQL Server 7, 2000, 2005, 2008

Visual Paradigm

Visual Paradigm es una herramienta profesional de UML que admite el ciclo de vida completo del software. Posibilita dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y obtener documentación. Es una herramienta que posee licencia gratuita y es multiplataforma.

Esta herramienta presenta características como (20):

- Soporte de UML versión 2.1.
- Modelado colaborativo con CVS y Subversion.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación, distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XML.

1.9. Herramientas de administración

EMS SQL Manager for PostgreSQL

EMS SQL Manager for PostgreSQL es una herramienta de alto rendimiento para la administración y desarrollo de bases de datos PostgreSQL. Cuenta con avanzada interfaz gráfica de usuario de fácil uso. Entre sus principales características para la administración eficiente de PostgreSQL se encuentran (21):

- Soporte completo de PostgreSQL hasta la versión 8.1.
- Nueva interfaz gráfica de usuario último modelo.
- Ágil navegación y administración de base de datos.
- Administración sencilla de todos los objetos PostgreSQL.
- Herramientas de manipulación de datos avanzada.
- Administración efectiva de seguridad.
- Excelentes Herramientas visuales y de texto para elaboración de consultas.
- Acceso al servidor PostgreSQL a través del protocolo HTTP.
- Conexión vía reenvío de puerto local a través del túnel SSH.
- Impresionantes opciones de exportación e importación de datos.
- Poderoso diseñador visual de base de datos.
- Asistentes fáciles de usar que realizan mantenimiento de tareas de PostgreSQL.

PgAdmin III

"PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows" (22).

"PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El 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, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad" (22).

1.10. Herramientas de pruebas

EMS Data Generator 2005 for PostgreSQL

EMS Data Generator 2005 for PostgreSQL es una poderosa herramienta que permite generar un gran volumen de datos de prueba a las tablas de bases de datos PostgreSQL. Posibilita la generación de datos para una o varias tablas a la vez, definiendo para cada campo el rango de valores admisibles, dependiendo del tipo, además de la cantidad de tuplas que se desean generar. Entre sus principales características se encuentran (23):

- Generación de datos a varias tablas desde diferentes bases de datos en un solo ordenador anfitrión (host).
- Soporte para todos los tipos de datos de PostgreSQL, incluyendo los tipos array (vector), network address (dirección de red) y geometric (geométricos).
- Diferentes tipos de generación por cada campo, incluyendo lista, azar (random), generación incremental de datos y más.
- Capacidad para usar resultados de consultas SQL como lista de valores para la generación de datos.
- Control automático sobre integridad referencial para la generación de datos a tablas vinculadas.
- Amplia variedad de generación de parámetros para cada tipo de campo.
- Capacidad para configurar valores Null (nulos) para ciertos casos.
- Posibilidad de salvar todos los parámetros de generación, para comenzar la sesión del asistente actual.
- Utilitario de líneas de comandos para generar datos usando el archivo que contiene la plantilla.

Apache Jmeter

Jmeter es una herramienta Open Source realizada en java (es un proyecto de Apache) que se utiliza para realizar pruebas de rendimiento, normalmente de aplicaciones web. Jmeter permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga. “Las características principales de Apache Jmeter son las siguientes” (24):

- Permite cargar y realizar pruebas sobre distintos tipos de servidores: Web (HTTP, HTTPS), SOAP, Bases de datos (JDBC), LDAP, JMS, Email (POP3 e IMAP).

- Multiplataforma: Unix (Solaris, Linux, etc.), Windows (98, NT, XP, etc.), OpenVMS Alpha 7.3+.
- Testeo distribuido.
- Multihilo: permite realizar pruebas de forma concurrente.
- Interfaz gráfica: permite realizar las operaciones más rápido.
- Permite comprobar cómo se comportará una aplicación web ante multitud de usuarios y la velocidad de carga que tendrá.
- Extensible: personalización.

El JMeter brinda gran flexibilidad respecto a la configuración y muestra los resultados de las pruebas en una amplia variedad de informes y gráficas, con una gran cantidad de variables diferentes que permiten interpretar los resultados desde diferentes puntos de vista.

Conclusiones parciales

Con el resultado de la investigación parcial se describieron las principales características de las herramientas, modelos y gestores de bases de datos posibles a utilizar para el desarrollo del modelo de datos de SIC.

Capítulo 2: Descripción y Análisis de la Solución Propuesta

Introducción

En el presente capítulo se definen actividades importantes para la construcción de la base de datos, la selección del modelo de base de datos a usar así como las herramientas, la descripción de la arquitectura a utilizar en el proyecto y la selección y argumentación de los requisitos no funcionales del sistema propuesto, trazados desde el negocio y la fase de requerimientos. Además se confeccionan el diagrama de clases persistentes y el diagrama entidad relación de la base de datos, con el fin de generar las clases del dominio.

2.1. Soluciones Técnicas

Como modelo de base de datos para SIC se propone el uso del modelo relacional debido a que el mismo evita la duplicidad de registros mediante un identificador único, garantiza la integridad referencial, ya que elimina todos los registros dependientes a uno que haya sido eliminado anteriormente, y favorece la normalización por ser más comprensible y aplicable, evitando grandes dificultades en la actualización.

De las herramientas referenciadas en el capítulo anterior, la herramienta CASE escogida para realizar el diseño de la base de datos de SIC es Visual Paradigm porque es la que tiene mejores características en cuanto a desempeño y licencia.

Se ha seleccionado PgAdmin III como herramienta de administración de base de datos por ser la más completa y popular bajo licencia de software libre.

Como herramienta para realizar las pruebas de carga a la base de datos de SIC se ha escogido JMeter por tener licencia de software libre y por dejar interpretar los resultados desde diferentes puntos de vista.

Para la realización del proceso de ingeniería inversa a la base de datos se propone el uso de la herramienta GRAG, un generador objetos del dominio libre para el framework Grails, que permite generar Clases del Dominio Grails después de aplicar la ingeniería inversa a una base de datos existente.

2.2. Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener, o sea, las características que hacen a este sistema atractivo, usable, rápido y confiable. Forman una parte significativa de la especificación de requisitos de un proyecto, ya que en muchos casos, son

fundamentales en el éxito del producto. A continuación se describen los principales requerimientos relacionados con la base de datos.

2.2.1. Confiabilidad

- Integridad de datos del RIU: El sistema debe ser capaz de mantener la calidad del dato de manera que garantice su integridad durante su aplicación, procesamiento y almacenamiento en el RIU; además de cuando sean adquiridos desde estaciones de trabajo externas.

- Actualización automática del RIU: Cada vez que se detecta un cambio en el RIU se debe replicar de manera automática los cambios para asegurar que los equipos de respaldo se mantengan actualizados.

- Restablecer la comunicación con otros sistemas externos ante la presencia de fallos: El sistema debe permitir restablecer comunicación con otros sistemas externos tras la ocurrencia de una falla.

- Tiempo de recuperación ante fallas: El sistema debe restablecer sus funciones 30 segundos después de haber ocurrido la falla. No se desea perder la operatividad por períodos prolongados. El tiempo máximo que esperarán los usuarios para la recuperación es de 5 minutos.

2.2.2. Desempeño

- Garantizar procesos y transacción masiva de datos: Los procesos y transacción masiva de datos deben estar definidos hacia y desde todas las estaciones de trabajo y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema.

- Tiempos requeridos para realizar consultas al RIU de acuerdo a la metodología aplicada: Los tiempos de consulta al RIU deben ajustarse de acuerdo al criterio y/o metodología aplicada y deben responder a las funciones requeridas por los usuarios en lapsos de espera que permitan cumplir con la funcionalidad solicitada. El tiempo máximo de respuesta esperado por los usuarios es de 5 segundos.

- Concurrencia de usuarios: El sistema debe soportar mínimo diez usuarios concurrentes usando todas las funcionalidades del sistema.

- Acceso a los Datos en el RIU: El RIU debe permitir ser accedido, modificado y actualizado independientemente del número de datos que maneje. La frecuencia máxima de acceso a los datos se prevé semanal. La frecuencia de uso mínima, cada hora todos los días.

2.2.3. Restricciones en el diseño y la implementación

- Sistema Gestor de Base de Datos: Se debe utilizar el Gestor de Base de Datos PostgreSQL.

- Framework: En el desarrollo del sistema se debe emplear el framework Grails que conlleva a la aparición de GORM (Grails Object Relational Mapping) como la tecnología para controlar el ciclo de vida de las entidades.

- IDE de desarrollo: Se debe utilizar como IDE de desarrollo Spring Source Tool Suite (STS).

- Lenguaje de programación: El sistema debe ser implementado en el lenguaje Groovy.

2.2.4. Seguridad

- Confidencialidad: La información almacenada en la base de datos debe estar protegida de acceso no autorizado y divulgación.

- Integridad: La información almacenada en la base de datos debe estar protegida contra la corrupción y estados inconsistentes.

- Disponibilidad: El acceso a la información debe garantizarse a los usuarios autorizados (autenticados por dominio y según su rol).

2.2.5. Hardware

- Servidor de Base de Datos: El servidor de base de datos debe tener como mínimo una PC Pentium 4, 2.4 GHz, 512 MB de memoria RAM y 160 GB de disco duro.

2.3. Descripción de la arquitectura y fundamentación

La arquitectura constituye la base fundamental en el desarrollo de un software. Se concentra en requerimientos no funcionales, que se satisfacen mediante los modelos y diseños de la aplicación. En el marco del trabajo con base de datos se puede decir que constituye el conjunto de reglas y normas a respetar para facilitar el trabajo de desarrollo y mantenimiento de los objetos en la base de datos. Un diseño correcto de la arquitectura del sistema es esencial para el éxito o fracaso del proyecto.

El sistema a construir estará desarrollado bajo la plataforma Java, la cual define estándares para desarrollar y ejecutar aplicaciones en el lenguaje de programación Java, empleando arquitecturas que definen un modelo multicapa y que se apoyan en componentes de software modulares. Las aplicaciones desarrolladas en esta plataforma tienden a ser portables, escalables, robustas y seguras a la vez que son integrables con tecnologías anteriores. De acuerdo a estas características el sistema será basado en el estilo arquitectónico Cliente-Servidor y este a su vez seguirá un estilo lógico en 3 capas, como se muestra en la figura 4:

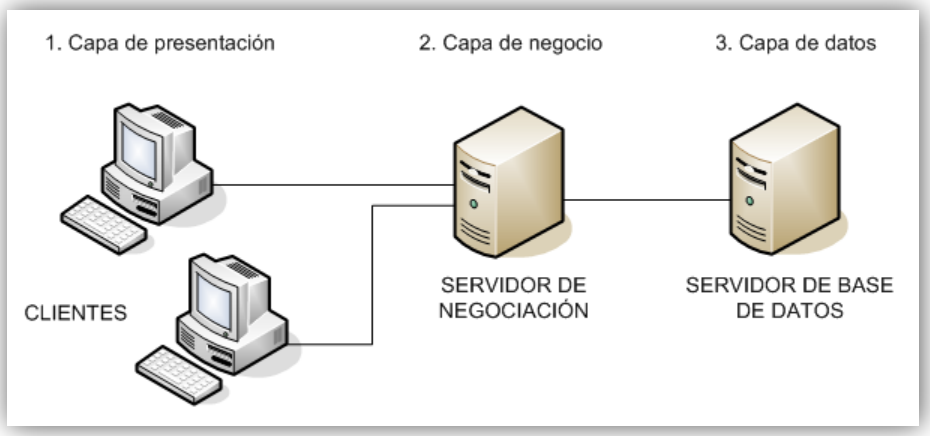

Figura 3. Arquitectura 3-Capas

La implementación del modelo de datos de SIC se encuentra dentro de la Capa de Acceso a Datos, que es la encargada de hacer persistente toda la información, además de suministrar y almacenar la información para el nivel de negocio. Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación y está dividida en tres subcapas las cuales facilitan el acceso de forma efectiva a la base de datos desde un contexto orientado a objetos.

- **Subcapa de ORM (Object Relational Mapping):** Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia, que en nuestro caso es **GORM** (Grails Object Relational Mapping), un gestor de persistencia escrito en Groovy, para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos dinámicos, los cuales se crean en tiempo de ejecución para cada entidad y facilitan enormemente las búsquedas.

  GORM está construido sobre Hibernate, una herramienta de mapeo objeto-relacional, que se encarga de relacionar las entidades de una clase con las tablas de una base de datos, y las propiedades de las entidades con los campos en las tablas. Entonces cada una de las

operaciones que se realicen sobre los objetos del modelo de datos, serán traducidas por Hibernate en las sentencias SQL necesarias para quedar reflejadas en la base de datos.

- **Subcapa física de la Base de Datos:** Contiene los componentes físicos de la base de datos, es decir los archivos de datos que residen en el disco.
- **Subcapa lógica de la Base de Datos:** Contiene las estructuras que mapean los datos hacia los componentes físicos.

2.4. Proceso para la creación de Clases del Dominio

El modelo de datos en una aplicación Grails está compuesto por las Clases de Dominio. El proceso de creación de las Clases del Dominio contempló que se realizaran 4 subprocesos, que se ejecutan secuencialmente, siendo necesario haber terminado con uno para empezar el otro. Concluir los 4 subprocesos satisfactoriamente implica realizar el proceso de diseño e implementación del modelo de datos de manera exitosa y cumplir así con los objetivos del presente trabajo.

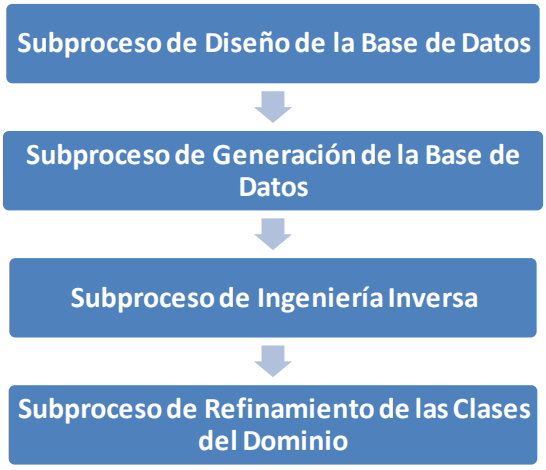

Figura 4. Subprocesos para la creación de Clases del Dominio

2.4.1. Subproceso de Diseño de la Base de Datos

El diseño de la base de datos depende en gran medida de una correcta selección de las entidades que formarán parte la misma, de la información que contendrá cada una de dichas entidades, de las relaciones que se establezcan entre las mismas y de un proceso de normalización correcto.

2.4.1.1. Diagrama de Clases Persistentes

Para el diseño de la base de datos del sistema, se parte de un diagrama de clases persistentes, definiendo la persistencia como la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. El diagrama de clases persistentes se utiliza para modelar la estructura lógica de la

base de datos, donde las clases son representadas por tablas y los atributos de clase por columnas. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos.

A continuación se muestra el diagrama de clases persistentes de SIC, realizado a través de la herramienta Visual Paradigm, donde se representa las clases persistentes con sus atributos y las relaciones que existen entre dichas clases.

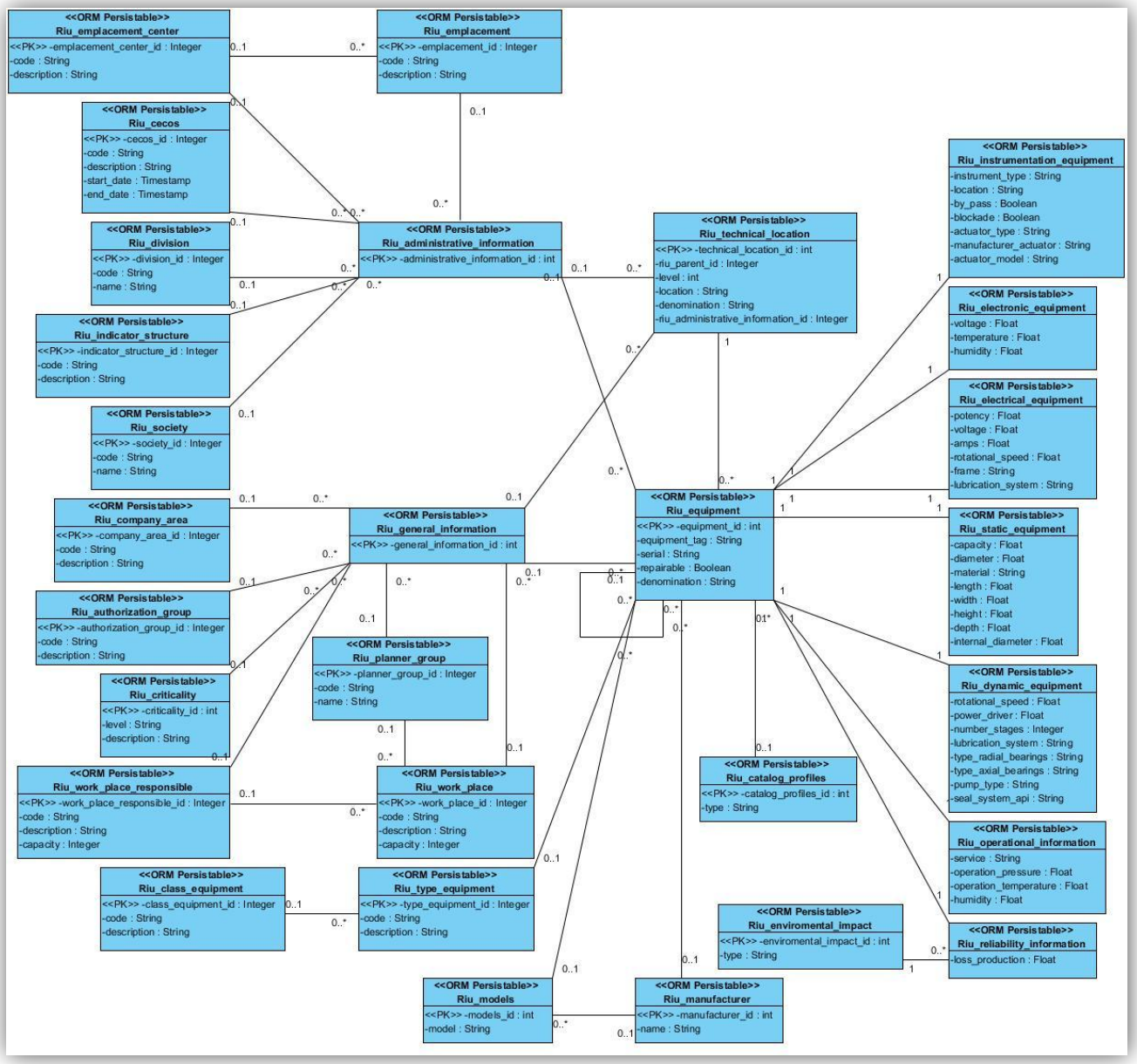

Figura 5. Diagrama de Clases Persistentes SIC

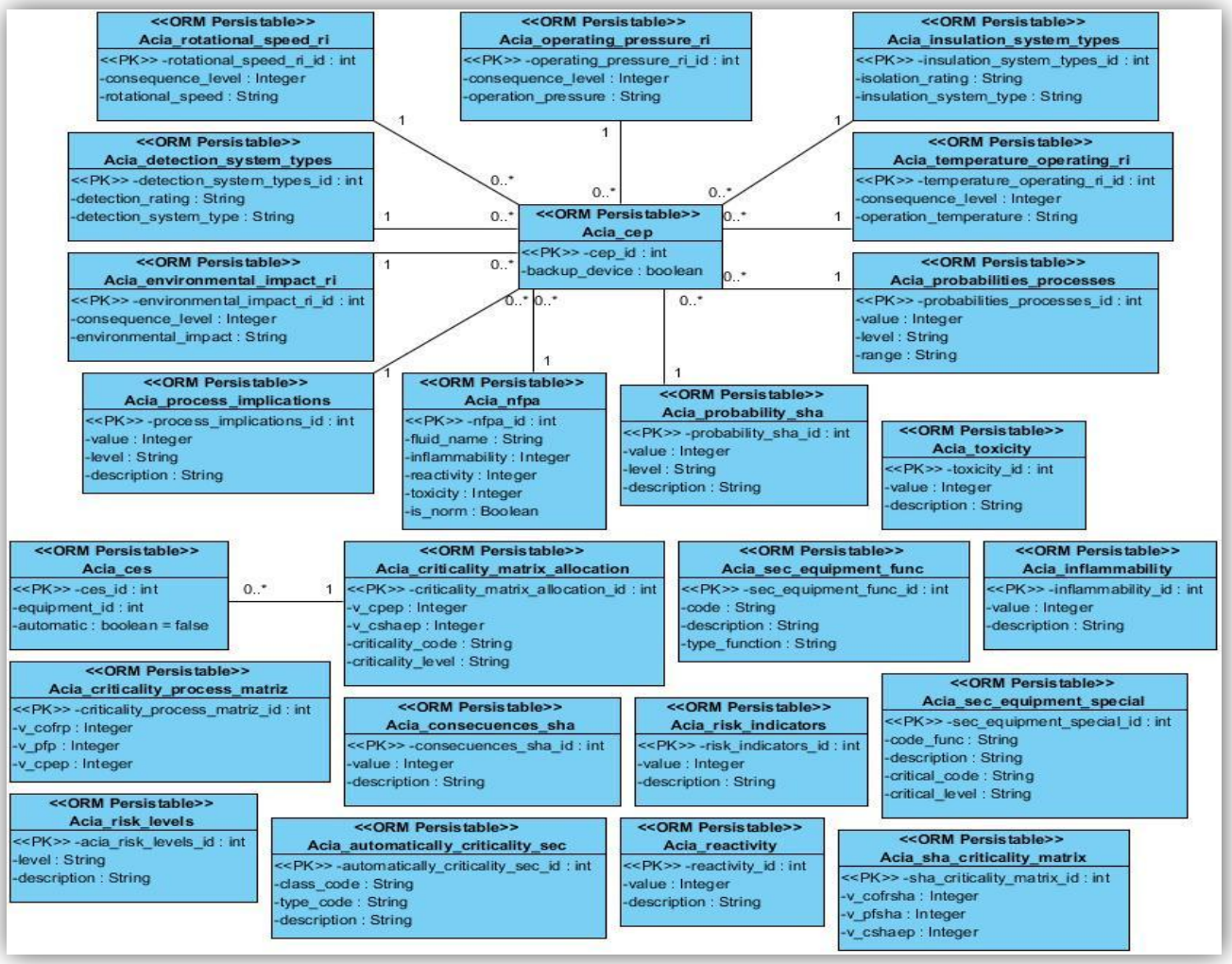

Figura 6. Diagrama de Clases Persistentes SIC

2.4.1.2. Patrones de diseño de Bases de Datos

"Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software" (25). Un patrón de diseño "(…) describe una estructura común y recurrente de componentes interrelacionados, que resuelve un problema general de diseño dentro de un contexto particular" (26).

Durante la realización del Diagrama de Clases Persistentes se usaron varios patrones que hicieron que el diseño fuera más fácil. A continuación se describen algunos de los patrones utilizados:

Árboles fuertemente codificados

"Este es un patrón de diseño de base de datos asociado al patrón árbol. Un árbol es un conjunto de nodos conectados en la estructura de hijo a padre. Un nodo pude tener muchos nodos hijos pero solo un padre, con excepción del nodo raíz y no existen los ciclos por lo que un camino solo

conecta a dos nodos" (27). En el patrón árboles fuertemente codificados a cada nivel del árbol se le asocia una entidad y admite tantos niveles como requiera la jerarquía que se vaya a representar.

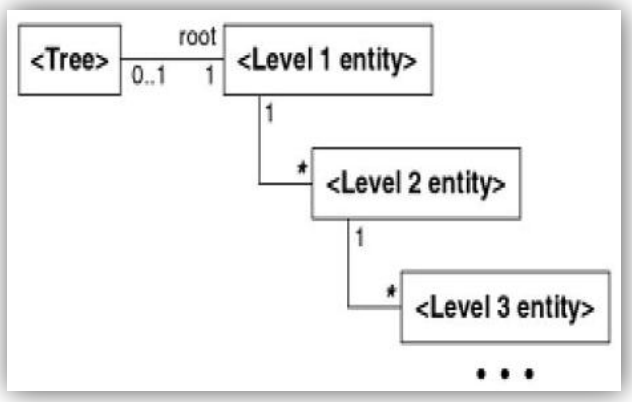

Figura 7. Árboles fuertemente codificados

Se hizo uso de este patrón en la entidad *riu_technical_location*, para representar la estructura organizativa de las ubicaciones técnicas de los equipos.

Árboles simples

Este patrón es normalmente utilizado cuando el árbol es la representación de una estructura de datos. Los elementos a almacenar son del mismo tipo, es decir pueden ser almacenados en la misma entidad. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre (27).

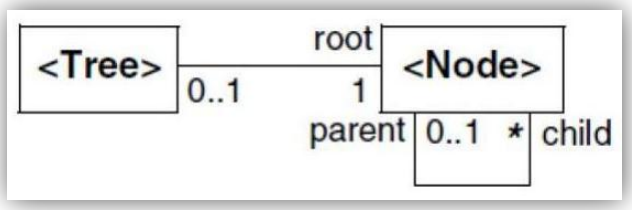

Figura 8. Árboles simples

Este patrón fue usado en la entidad *riu_equipment*, para representar que un equipo puede formar parte de otro.

Llaves subrogadas

El propósito de este patrón es generar una llave primaria única para cada entidad en vez de usar un atributo identificador en el contexto dado. Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas.

En el caso de la base de datos en cuestión se usaron tipos de datos enteros en columnas identity, las cuales contienen valores generados por el sistema que identifican en exclusiva cada fila de una tabla.

Patrón Control de acceso basado en roles (RBAC)

El empleo de este patrón se realizó con el objetivo de conservar la seguridad de la base de datos de SIC. El patrón plantea que se deben asignar permisos para ejecutar operaciones a roles específicos, que luego serán asignados a usuarios del sistema en dependencia de las funciones de trabajo. Así el manejo de los permisos de cada usuario se convierte en una cuestión de simplemente asignar los roles apropiados al usuario.

2.4.1.3. Diagrama Entidad Relación

Para la realización del diagrama entidad relación se hizo uso de la funcionalidad *Sincronizar a Diagrama Entidad-Relación* que brinda Visual Paradigm.

Las siguientes figuras representan el diagrama entidad relación correspondiente a la base de datos de SIC.

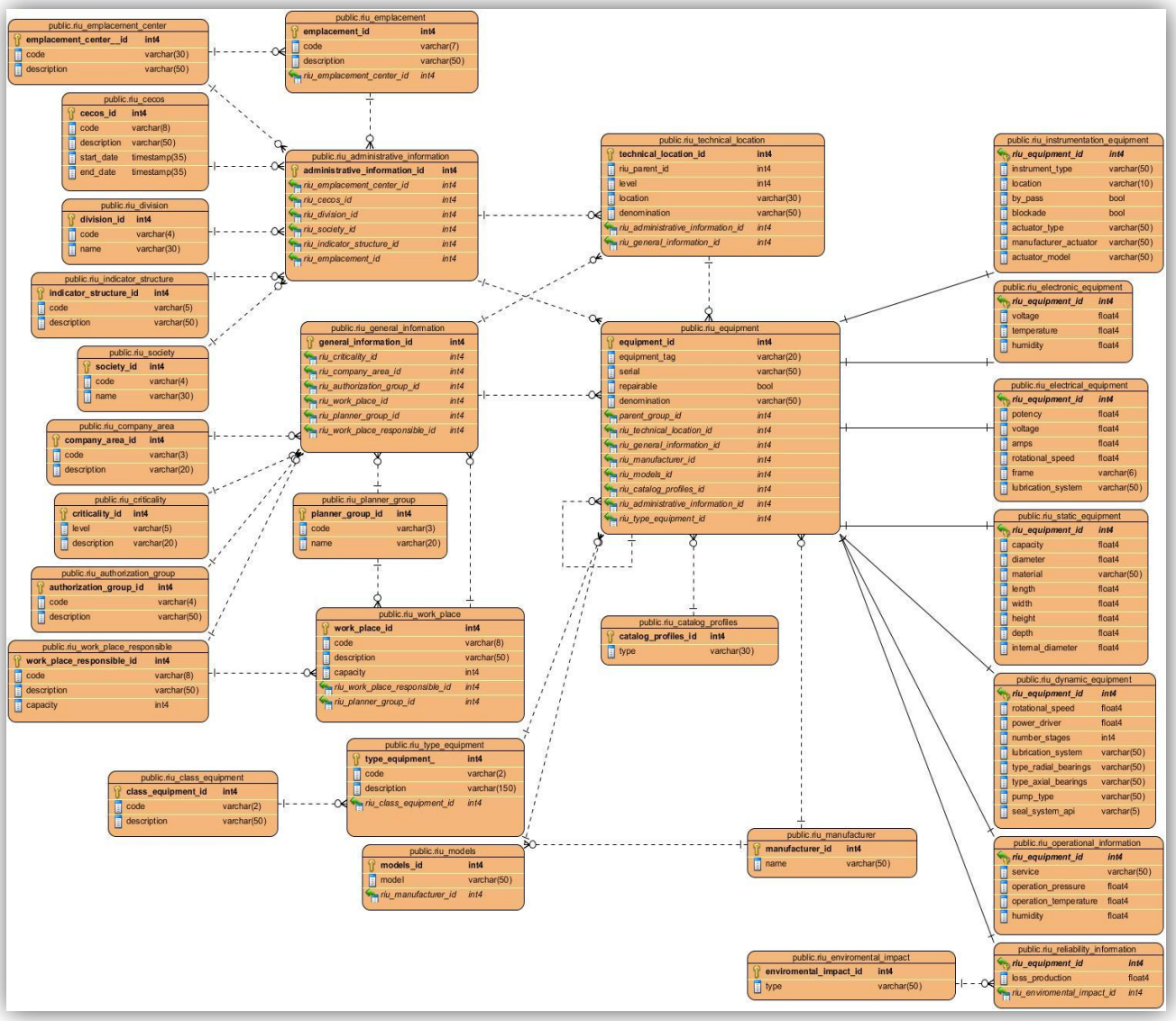

Figura 9. Diagrama Entidad Relación SIC

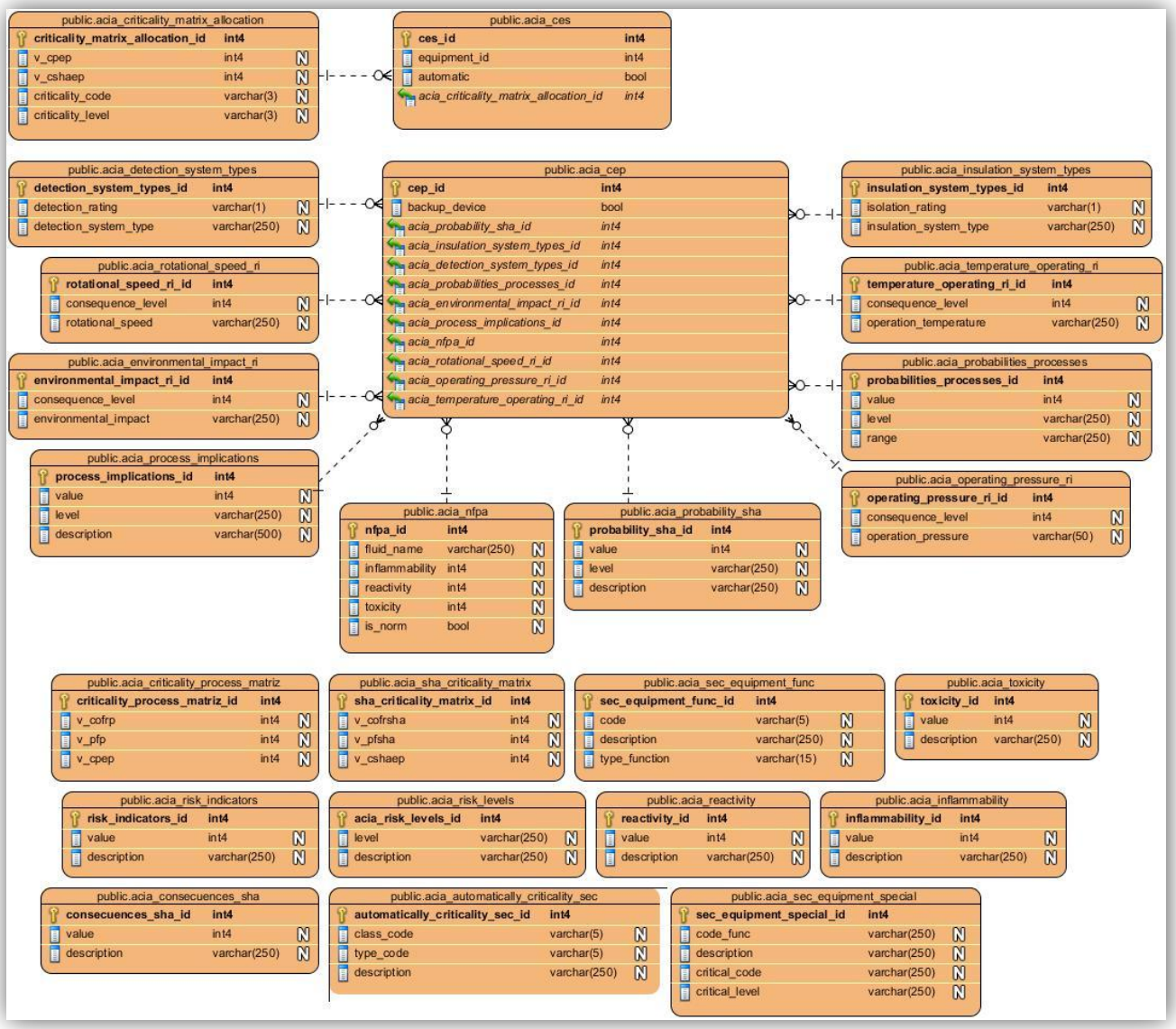

Figura 10. Diagrama Entidad Relación SIC

2.4.1.4. Descripción de las tablas del diagrama Entidad Relación

En esta sección se describen de forma general los datos que se almacenan en las principales tablas del diagrama entidad relación de la base de datos de SIC, al igual que se explica brevemente lo que representan todos sus atributos.

Tabla 1. Descripción de la tabla riu_technical_location

Nombre: riu_technical_location		
Descripción: Almacena las ubicaciones técnicas en donde se encuentran los equipos.		
Atributo:	**Tipo:**	**Descripción:**
riu_technical_location_id		Identificador de la entidad.

riu_parent_id	integer	Identificador de la ubicación técnica a la que pertenece.
level	integer	Nivel de la ubicación técnica.
location	varchar	Nombre de la ubicación técnica.
denomination	varchar	Breve descripción de la ubicación técnica.
riu_general_information_id	integer	Llave foránea de la tabla riu_general_information.
riu_administrative_information_id	integer	Llave foránea de la tabla riu_administrative_information.

Tabla 2. Descripción de la tabla riu_equipment

Nombre: riu_equipment		
Descripción: Almacena la información común de los equipos.		
Atributo:	**Tipo:**	**Descripción:**
equipment_id	integer	Identificador de la entidad.
equipment_tag	varchar	Etiqueta de identificación del equipo.
serial	varchar	Serial del equipo.
repairable	bool	Indica si el equipo tiene reparación.
denomination	varchar	Breve descripción del equipo.
parent_group_id	integer	Llave foránea de la tabla riu_equipment.
riu_technical_location_id	integer	Llave foránea de la tabla riu_technical_location.
riu_general_information_id	integer	Llave foránea de la tabla riu_general_information.
riu_manufacturer_id	integer	Llave foránea de la tabla riu_manufacturer.
riu_type_equipment_id	varchar	Llave foránea de la tabla riu_type_equipment.
riu_models_id	integer	Llave foránea de la tabla riu_models.
riu_catalog_profiles_id	integer	Llave foránea de la tabla riu_catalog_profiles.
riu_administrative_information_id	integer	Llave foránea de la tabla riu_administrative_information.

Tabla 3. Descripción de la tabla riu_dynamic_equipment

Nombre: riu_dynamic_equipment

Descripción: Almacena la información específica del equipo dinámico.		
Atributo:	**Tipo:**	**Descripción:**
riu_equipment_id	integer	Identificador de la entidad.
rotational_speed	float	Velocidad de rotación del equipo principal en revoluciones por minuto (rpm).
power_driver	float	Potencia de la sub-unidad de equipo que genera la fuerza utilizada por el equipo dinámico expresada en Kilowattios.
number_stages	integer	Indica el número de etapas del equipo dinámico.
lubrication_system	varchar	Sistema de lubricación utilizado por el equipo dinámico.
type_radial_bearings	varchar	Tipos de cojinetes radiales del equipo dinámico.
type_axial_bearings	varchar	Tipos de cojinetes axiales del equipo dinámico.
pump_type	varchar	Tipo de bomba según la norma API 610.
seal_system_api	varchar	Sistema API de sello de la bomba según la norma API 682.

Tabla 4. Descripción de la tabla riu_ electrical_equipment

Nombre: riu_electrical_equipment		
Descripción: Almacena la información específica del equipo eléctrico.		
Atributo:	**Tipo:**	**Descripción:**
riu_equipment_id	integer	Identificador de la entidad.
potency	float	Potencia del equipo eléctrico expresada en kilowattios
voltage	float	Voltaje de trabajo del equipo eléctrico en voltios.
amps	float	Corriente de trabajo del equipo eléctrico en amperios.
rotational_speed	float	Velocidad de rotación generada por el motor eléctrico en revoluciones por minuto (rpm).
frame	varchar	Frame del motor eléctrico.
lubrication_system	varchar	Sistema de lubricación utilizado por el motor eléctrico.

Tabla 5. Descripción de la tabla riu_static_equipment

Nombre: riu_static_equipment		
Descripción: Almacena la información específica del equipo estático.		
Atributo:	**Tipo:**	**Descripción:**

riu_equipment_id	integer	Identificador de la entidad.
capacity	float	Capacidad de almacenamiento del equipo estático en metros cúbicos.
diameter	float	Diámetro del equipo estático en metros.
material	varchar	Material de construcción del equipo estático.
length	float	Longitud del equipo estático en metros.
width	float	Ancho del equipo estático en metros.
height	float	Altura del equipo estático en metros.
depth	float	Espesor del equipo estático en milímetros.
internal_diameter	float	Diámetro interno del equipo estático en pulgadas.

Tabla 6. Descripción de la tabla riu_instrumentation_equipment

Nombre: riu_instrumentation_equipment		
Descripción: Almacena la información específica del equipo de Instrumentación.		
Atributo:	**Tipo:**	**Descripción:**
riu_equipment_id	integer	Identificador de la entidad.
instrument_type	varchar	Descripción del tipo de instrumento.
location	varchar	Ubicación del instrumento.
by_pass	boolean	Indica si la válvula posee by pass (vía alterna) para facilitar el mantenimiento.
blockade	boolean	Indica si la válvula posee bloqueo aguas abajo y aguas arriba en el proceso para realizar mantenimiento.
actuator_type	varchar	Indica el tipo de actuador de la válvula.
manufacturer_actuator	varchar	Fabricante del actuador de la válvula.
actuator_model	varchar	Modelo del actuador de la válvula.

Tabla 7. Descripción de la tabla riu_electronic_equipment

Nombre: riu_electronic_equipment		
Descripción: Almacena la información específica del equipo electrónico.		
Atributo:	**Tipo:**	**Descripción:**

riu_equipment_id	integer	Identificador de la entidad.
voltage	float	Voltaje de trabajo del equipo electrónico en voltios.
temperature	float	Temperatura del equipo electrónico en grado Celsius.
humidity	float	Humedad del equipo electrónico en porciento.

Tabla 8. Descripción de la tabla riu_operational_information

Nombre: riu_operational_information		
Descripción: Almacena la información operacional del equipo.		
Atributo:	**Tipo:**	**Descripción:**
riu_equipment_id	integer	Identificador de la entidad.
service	varchar	Tipo de fluido manejado por el equipo principal del grupo de equipos.
operation_pressure	float	Presión de operación normal del equipo en Psi.
operation_temperature	float	Temperatura de operación normal del equipo en grados Celsius.
humidity	float	Humedad del ambiente donde se encuentra el equipo electrónico en %. Este campo solo aplica para equipos electrónicos

Tabla 9. Descripción de la tabla riu_administrative_information

Nombre: riu_administrative_information		
Descripción: Almacena la información administrativa común para ubicaciones técnicas y equipos.		
Atributo:	**Tipo:**	**Descripción:**
administrative_information_id	integer	Identificador de la entidad.
riu_society_id	varchar	Llave foránea de la tabla riu_society.
riu_division_id	varchar	Llave foránea de la tabla riu_division.
riu_indicator_structure_id	varchar	Llave foránea de la tabla riu_indicator_structure.
riu_emplacement_id	varchar	Llave foránea de la tabla riu_emplacement.
riu_emplacement_center_id	varchar	Llave foránea de la tabla riu_emplacement_center.
riu_cecos_id	varchar	Llave foránea de la tabla riu_cecos.

Tabla 10. Descripción de la tabla riu_society

Nombre: riu_society		
Descripción: Almacena la información referente a la sociedad.		
Atributo:	**Tipo:**	**Descripción:**
society_id	integer	Identificador de la entidad.
code	varchar	Código de la sociedad.
name	varchar	Nombre de la sociedad.

Tabla 11. Descripción de la tabla riu_division

Nombre: riu_division		
Descripción: Almacena la información referente a la división.		
Atributo:	**Tipo:**	**Descripción:**
división_id	integer	Identificador de la entidad.
code	varchar	Código de la división.
name	varchar	Nombre de la entidad.

Tabla 12. Descripción de la tabla riu_emplacement_center

Nombre: riu_emplacement_center		
Descripción: Almacena la información referente al centro de emplazamiento.		
Atributo:	**Tipo:**	**Descripción:**
emplacement_center_id	integer	Identificador de la entidad.
code	varchar	Código del centro de emplazamiento.
description	varchar	Descripción del centro de emplazamiento.

Tabla 13. Descripción de la tabla riu_indicator_structure

Nombre: riu_indicator_structure		
Descripción: Almacena la información del indicador de estructura de la ubicación técnica.		
Atributo:	**Tipo:**	**Descripción:**
indicator_structure_id	integer	Identificador de la entidad.

code	varchar	Código del indicador de estructura.
description	varchar	Descripción del indicador de estructura.

Tabla 14. Descripción de la tabla riu_cecos

Nombre: riu_cecos		
Descripción: Almacena la información de los Centros de Coste.		
Atributo:	**Tipo:**	**Descripción:**
cecos_id	integer	Identificador de la entidad.
code	varchar	Código del Centro de Coste.
description	varchar	Descripción del Centro de Coste.
start_date	timestamp	Código de inicio del Centro de Coste.
end_date	timestamp	Código de fin del Centro de Coste.

Tabla 15. Descripción de la tabla riu_emplacement

Nombre: riu_emplacement		
Descripción: Almacena la información referente al emplazamiento.		
Atributo:	**Tipo:**	**Descripción:**
emplacement _id	integer	Identificador de la entidad.
code	varchar	Código del emplazamiento.
description	varchar	Descripción del emplazamiento.
riu_emplacement_center_id	integer	Llave foránea de la tabla riu_emplacement_center.

Tabla 16. Descripción de la tabla riu_general_information

Nombre: riu_general_information		
Descripción: Almacena la información general común para ubicaciones técnicas y equipos.		
Atributo:	**Tipo:**	**Descripción:**
general_information_id	integer	Identificador de la entidad.
riu_criticality_id	integer	Llave foránea de la tabla riu_criticality.
riu_company_area_id	varchar	Llave foránea de la tabla riu_company_area.

riu_planner_group_id	varchar	Llave foránea de la tabla riu_planner_group.
riu_work_place_id	varchar	Llave foránea de la tabla riu_work_place.
riu_authorization_group_id	varchar	Llave foránea de la tabla riu_authorization_group.
riu_work_place_responsible_id	varchar	Llave foránea de la tabla riu_work_place_responsible.

Tabla 17. Descripción de la tabla riu_company_area

Nombre: riu_company_area		
Descripción: Almacena la información de cada una de las áreas de la empresa.		
Atributo:	**Tipo:**	**Descripción:**
company_area_id	integer	Identificador de la entidad.
code	varchar	Código del área de la empresa.
description	varchar	Descripción del área de la empresa.

Tabla 18. Descripción de la tabla riu_criticality

Nombre: riu_criticality		
Descripción: Almacena la información referente al nivel de criticidad que posee un objeto técnico.		
Atributo:	**Tipo:**	**Descripción:**
criticality_id	integer	Identificador de la entidad.
level	varchar	Nivel de criticidad.
description	varchar	Descripción de la criticidad.

Tabla 19. Descripción de la tabla riu_authorization_group

Nombre: riu_authorization_group		
Descripción: Almacena la información referente a los grupos de autorizaciones.		
Atributo:	**Tipo:**	**Descripción:**
authorization_group_id	integer	Identificador de la entidad.
code	varchar	Código de los grupos de autorizaciones.
description	varchar	Descripción de los grupos de autorizaciones.

Tabla 20. Descripción de la tabla riu_work_place_responsible

Nombre: riu_work_place_responsible		
Descripción: Almacena la información referente al supervisor de mantenimiento.		
Atributo:	**Tipo:**	**Descripción:**
work_place_responsible_id	integer	Identificador de la entidad.
code	varchar	Código del responsable del puesto de trabajo.
description	varchar	Descripción del responsable del puesto de trabajo.
capacity	integer	Capacidad del puesto de trabajo.

Tabla 21. Descripción de la tabla riu_planner_group

Nombre: riu_planner_group		
Descripción: Almacena la información referente a los grupos de planificación de mantenimiento.		
Atributo:	**Tipo:**	**Descripción:**
planner_group_id	integer	Identificador de la entidad.
code	varchar	Código del grupo de planificación.
description	varchar	Descripción del grupo de planificación.

Tabla 22. Descripción de la tabla riu_work_place

Nombre: riu_work_place		
Descripción: Almacena la información referente al puesto de trabajo.		
Atributo:	**Tipo:**	**Descripción:**
work_place_id	integer	Identificador de la entidad.
code	varchar	Código del puesto de trabajo.
description	varchar	Descripción del puesto de trabajo.
capacity	integer	Capacidad del puesto de trabajo.
riu_planner_group_id	varchar	Llave foránea de la tabla riu_planner_group.
riu_work_place_responsible_id	varchar	Llave foránea de la tabla riu_work_place_responsible.

Tabla 23. Descripción de la tabla riu_ type_equipment

Nombre: riu_type_equipment		
Descripción: Almacena la información referente al tipo de equipo.		
Atributo:	**Tipo:**	**Descripción:**
type_equipment_id	integer	Identificador de la entidad.
code	varchar	Código del tipo de equipo.
description	varchar	Descripción del tipo de equipo.
riu_class_equipment_id	varchar	Llave foránea de la tabla riu_class_equipment.

Tabla 24. Descripción de la tabla riu_class_equipment

Nombre: riu_class_equipment		
Descripción: Almacena la información referente a la clase de equipo.		
Atributo:	**Tipo:**	**Descripción:**
class_equipment_id	integer	Identificador de la entidad.
code	varchar	Código de la clase de equipo.
description	varchar	Descripción de la clase de equipo.

Tabla 25. Descripción de la tabla riu_models

Nombre: riu_models		
Descripción: Almacena la información referente al modelo del equipo.		
Atributo:	**Tipo:**	**Descripción:**
model_id	integer	Identificador de la entidad.
description	varchar	Descripción del equipo.
riu_class_equipment_id	varchar	Llave foránea de la tabla riu_class_equipment.

Tabla 26. Descripción de la tabla riu_manufacturer

Nombre: riu_manufacturer		
Descripción: Almacena la información referente al fabricante del equipo.		
Atributo:	**Tipo:**	**Descripción:**

manufacturer_id	integer	Identificador de la entidad.
name	varchar	Nombre de la empresa, entidad o persona.

Tabla 27. Descripción de la tabla riu_catalog_profiles

Nombre: riu_catalog_profiles		
Descripción: Almacena la información referente al perfil de catálogo del equipo.		
Atributo:	**Tipo:**	**Descripción:**
catalog_profiles_id	integer	Identificador de la entidad.
type	varchar	Tipo de catálogo.

2.4.2. Subproceso de Generación de la Base de Datos

Para generar la base de datos diseñada anteriormente se realizaron una serie de pasos que se explican a continuación:

1. A través de la funcionalidad *Generar Base de Datos* que brinda Visual Paradigm se configura la base de datos especificando: el gestor a utilizar que en nuestro caso es PostgreSQL, la dirección del controlador JDBC, la dirección del servidor de base de datos, así como el nombre, puerto, usuario y contraseña de la misma.

2. Se genera el script de la base de datos a partir del Diagrama Entidad Relación.

3. Se crea una base de datos en blanco llamada SICO a través de PgAdmin III.

4. Haciendo uso de PgAdmin III se ejecuta el script en la base de datos SICO, creándose en la misma todas las tablas del Diagrama Entidad Relación.

2.4.3. Subproceso de Ingeniería Inversa

Una vez creada la base de datos SICO es necesario realizarle el proceso de ingeniería inversa a la misma con el objetivo de crear las Clases del Dominio de Grails. Para ello se utiliza la herramienta GRAG, realizando los siguientes pasos:

1. Se crean los parámetros de conexión a la base de datos de SICO, ejemplificando de la misma: el tipo de base de datos, el driver, el usuario y la contraseña.

2. Se especifica dónde se va a guardar la aplicación que se va a generar.

3. Se procede a la generación de la aplicación.

2.4.4. Subproceso de Refinamiento de las Clases del Dominio

Ya creadas las clases del dominio, se procede a refinarlas manualmente ya que la maquinaria de GORM no es tan exacta en cuanto a las estructuras de tablas heredadas: claves primarias sin una columna id o columnas de fecha que no son de tipo date sino enteros. Para ello se establece la conexión de la base de datos con Grails:

1. Se copia el driver de postgreSQL para Java en el directorio *lib.*

2. Se configura el archivo *DataSource.groovy* que se encuentra en el directorio *grails-app/conf* con el objetivo de introducir los parámetros de conexión con la base de datos: nombre de la clase del controlador JDBC, nombre del usuario para establecer la conexión JDBC y su contraseña.

Una vez creada la conexión a la base de datos y refinadas las clases del dominio en el lenguaje Groovy, se ejecuta el comando *run-app* para generar nuestra aplicación.

Conclusiones parciales

En este capítulo se plasmó una propuesta del diseño de la base de datos de SIC que recoge cada uno de los requerimientos del sistema. En dicha propuesta se definió el soporte arquitectónico que tendría la base de datos, así como se eliminó la redundancia en la información debido a que ya no existen datos repetidos innecesariamente, aumentando la integridad de los mismos. Como resultado del actual diseño se logró obtener una base de datos donde existe integridad, organización e independencia de los datos cumpliendo con las exigencias y objetivos planteados en la investigación.

Capítulo 3: Validación del diseño realizado

Introducción

En el presente capítulo se realiza una validación de la propuesta de solución para el diseño de la base de datos de SIC. Se definen aspectos tan importantes como son la integridad, redundancia de la información y seguridad de la base de datos. Además se tratan aspectos como las pruebas realizadas a la base de datos para comprobar su correcto y eficiente funcionamiento.

3.1. Validación teórica del diseño

La etapa de diseño constituye un punto crucial en el proceso de creación de la base de datos. Aspectos como la integridad de los datos, la normalización del diseño, la redundancia de información y la seguridad en la base de datos son de vital importancia no solo en la confección de un buen diseño de base de datos, sino para garantizar la consistencia e integridad de la información, asegurando la calidad de almacenamiento y disponibilidad de los datos.

3.1.1. Integridad

"La integridad en una base de datos se refiere a la corrección y exactitud de la información contenida. Una base de datos determinada podría estar sujeta a cualquier cantidad de restricciones de integridad de una complejidad arbitraria" (28). Cuando se utilizan sentencias INSERT, DELETE o UPDATE, la integridad de datos puede verse afectada; ya que se puede añadir datos no válidos o modificarse datos existentes de forma incorrecta, con lo que la integridad podría no cumplirse. "Las restricciones de integridad proporcionan un medio de asegurar que las modificaciones hechas a la base de datos por los usuarios autorizados no provoquen la pérdida de la consistencia de los datos. Por tanto, las restricciones de integridad protegen a la base de datos contra los daños accidentales" (29).

La integridad de los datos se contempla en diferentes niveles. Las restricciones de dominio, transacciones y entidades definen las reglas para el mantenimiento de la integridad de las relaciones individuales. Las relaciones de integridad referencial aseguran que se mantienen las asociaciones necesarias entre las relaciones. Las restricciones de integridad de la base de datos gobiernan la misma como un todo y las restricciones de integridad de transacciones controlan la forma en que se manipulan los datos, dentro de una o entre múltiples bases de datos.

3.1.1.1. 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" (30).

La integridad de dominio controla la información que se almacena en la base de datos. Estas normas o reglas de integridad de dominio permiten indicar cuáles son los campos que requieren tener obligatoriamente valores, para que la base de datos no tenga datos sin conectar en el caso de tener relaciones o dependencias entre tablas. Cuando se aumenta el número de limitaciones, mejora el funcionamiento de la base de datos.

Las restricciones de los dominios son la forma más simple de restricción de integridad. Además de permitir la verificación de los valores que se encuentran en la base de datos, chequean las consultas para asegurarse de que tengan sentido las comparaciones que hagan.

Para lograr la integridad de dominio en la base de datos se precisaron correctamente cada uno de los tipos de datos para cada uno de los atributos. Además se revisan los valores que no pueden clasificarse de nulos haciendo uso de la restricción NOT NULL.

3.1.1.2. Integridad Referencial

"La integridad referencial es una funcionalidad disponible en las bases de datos relacionales. Protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas y garantiza que los valores de clave sean coherentes en las distintas tablas. Para conseguir esa coherencia, es preciso que no haya referencias a valores inexistentes y que, si cambia el valor de una clave, todas las referencias a ella se cambien en consecuencia en toda la base de datos" (30).

La regla de integridad referencial define que la base de datos no puede contener valores de claves foráneas sin concordancia. Esta regla se aplica a las claves foráneas. Si en una relación hay alguna clave foránea, entonces sus valores deben coincidir con los valores de la clave primaria a la que hace referencia, o bien, puede ser completamente nulo.

La integridad referencial chequea que se cumplan ciertas reglas como no poder introducir datos en la tabla relacionada sin antes haberlos introducidos en la tabla principal, el impedimento de eliminar un registro de una tabla principal si existen registros coincidentes en la tabla relacionada y la denegación de cambiar un valor de la clave primaria en la tabla principal si la tupla tiene registros relacionados.

Los conceptos de actualizar registros en cascada y eliminar registros en cascada están asociados a la integridad referencial. El primero hace referencia a que cuando se cambie un valor del campo clave de la tabla principal, automáticamente cambiará el valor de la clave foránea de los registros relacionados en la tabla secundaria. El segundo indica al SGBD que cuando se elimina un registro de la tabla principal automáticamente se borran también los registros relacionados en la tabla secundaria.

Para mantener esta integridad referencial en la base de datos se utilizaron llaves foráneas, que obligan a los valores introducidos en las columnas marcadas por esta restricción a que correspondan con los valores en las tablas referenciadas. Además se definió que la realización de la actualización y eliminación de datos sería en cascada.

3.1.1.3. Integridad de Entidades

"La integridad de entidad define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY" (30).

"Las restricciones de entidades aseguran la integridad de las entidades que son modeladas por el sistema. En el nivel más simple, la existencia de una clave principal es una restricción de entidad que impone la regla "cada entidad debe estar identificada de forma única". En esta no está permitido que algún componente de clave primaria acepte valores nulos" (28).

En el caso de la base de datos que ese está analizando, cada entidad tiene definida su llave primaria, que no puede ser nula ni se puede repetir. Definir correctamente el identificador principal para cada registro, es de vital importancia, ya que esto representa la principal vía que utiliza el servidor de base de datos para seleccionar la información que se necesite.

3.1.1.4. 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" (30).

3.1.2. Análisis de redundancia de información

La redundancia de información se refiere a aquellos datos que se encuentran almacenados varias veces en la misma base de datos. Esto genera que cuando se inserten o actualicen datos haya que hacerlo en todos los lugares a la vez, provocando inconsistencia en los datos ya que como la información está duplicada se actualizan algunos datos y otros no. Además el espacio de almacenamiento se ve afectado ya que los mismos datos se encuentran almacenados en lugares distintos.

Los procesos de normalización mejoran en buena medida el comportamiento de este parámetro. En el caso de la base de datos de SIC, no cuenta con redundancia de información ya que la misma se encuentra normalizada hasta la 3FN. Además de la normalización se aplicó el uso de nomencladores para almacenar los datos comunes.

3.1.3. Análisis de la seguridad de la base de datos

La seguridad es un tema sumamente importante a tratar en este capítulo ya que garantiza el acceso autorizado a los datos, para interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intensión. La mayoría de los ataques que se realizan a las aplicaciones web tienen que ver con el uso incorrecto de las técnicas que se aplican en la generación de consultas a la base de datos, ya que al construir consultas SQL, introduciendo parámetros de la petición HTTP sin procesar, la aplicación está expuesta a ataques de inyección SQL. El ORM de Grails, GORM, implementa mecanismos de seguridad que protegen la aplicación de estos ataques tan comunes. El uso de sentencias preparadas (*prepared statements*) permite pasar cada parámetro de las consultas por separado y no concatenado, evitando así la inyección SQL. También con el uso de GORM en la capa de acceso a datos, se generan consultas que se requieren para validar a un usuario en la base de datos, buscar los roles con los que cuenta un usuario, así como las acciones específicas que puede realizar un usuario en dependencia de sus roles.

3.2. Validación funcional

Para comprobar el correcto funcionamiento de la base de datos, es necesario validarla funcionalmente. Para ello se realizan un conjunto de pruebas que tienen como objetivo simular una carga de producción real y posteriormente observar el comportamiento que tiene la base de datos bajo dicha carga. Esto permite solucionar los problemas de rendimiento antes de poner la base de datos en marcha.

3.2.1. Pruebas de rendimiento

En la Ingeniería del Software, las pruebas de rendimiento son aquellas que son realizadas para determinar qué tan rápido un sistema realiza una tarea bajo ciertas condiciones pre-planificadas de trabajo. Estas pruebas también son utilizadas para validar y verificar diferentes aspectos de la calidad de software, como por ejemplo, escalabilidad, fiabilidad y el buen uso de los recursos. “Las pruebas de rendimiento constituyen un subconjunto de la Ingeniería de Pruebas, la cual se esfuerza en mejorar el rendimiento, basándose en el diseño y la arquitectura de un sistema, antes de la realización del proceso de codificación” (31).

Las pruebas de rendimiento son de gran utilidad siempre y cuando sean usadas a través de herramientas que permitan visualizar los detalles técnicos y gráficos altamente informativos de los resultados de las pruebas. Entre los tipos de pruebas de rendimiento se encuentran las pruebas de volumen y las pruebas de carga.

3.2.1.1. Prueba de volumen

Las pruebas de volumen centran su trabajo en poblar a la base de datos con volúmenes de datos similares a los esperados en la explotación real del sistema para determinar si existen límites de almacenamiento, identificándose así la carga máxima que puede manejar la base de datos en un período dado.

Para la realización de este tipo de prueba se utilizó el plugin Build-Test-Data para Grails, el cual permite generar de forma automática datos de prueba validando la integridad referencial, ya que genera los datos que provienen de otras tablas para evitar errores.

En estas pruebas se pobló la base de datos con poco más de 500 000 tuplas, donde a la mayoría de las tablas se le insertaron alrededor de 10 000 tuplas.

Después de realizado el llenado a la base de datos, la misma no presentó problemas ya sea de límite de capacidad o volumen de datos, desbordamientos de columnas, atributos o tipos de datos. Esto garantiza que el diseño de las estructuras de la base de datos y el gestor utilizado para el desarrollo de la misma soporta el almacenamiento de los niveles de información requeridos para el comienzo del funcionamiento de la base de datos.

3.2.1.2. Prueba de carga

Este es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Si la base de datos, el servidor de aplicaciones, y otros componentes también se monitorizan, entonces esta prueba puede mostrar cuál es el cuello de botella en la aplicación.

Para la realización de estas pruebas se empleó la herramienta JMeter 2.4, quedando la configuración de la misma de la siguiente forma:

- Primero se creó un grupo de hilos donde se definieron la cantidad de usuarios que realizarían peticiones al servidor. (Anexo 1, figura 12)
- Seguidamente se configuró el acceso a la base de datos detallando el nombre de la variable que será usada para las peticiones, la dirección URL a través de la cual se conecta a la base de datos, con que usuario y contraseña se accede a la misma. (Anexo 1, figura 13)

- Luego se definió la petición que se le realizaría a la base de datos en el archivo Petición JDBC. (Anexo 1, figura 14)

Las pruebas se realizaron a la consulta que se emplea en la búsqueda avanzada de información de equipos en el RIU, pero con una cantidad de usuarios conectados diferentes, por lo que arrojaron resultados distintos que se presentan a continuación a través de cuatro variables que son muy importantes a tener en cuenta.

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

Consulta: Obtener de todos los equipos el número, el tag, la ubicación técnica, el código del tipo, el código de la clase y el perfil de catálogo.

```
SELECT
riu_equipment.equipment_id, riu_equipment.equipment_tag, riu_technical_location.location,
riu_type_equipment.code, riu_catalog_profiles.type, riu_class_equipment.code
FROM
public.riu_equipment JOIN public.riu_technical_location ON
riu_technical_location.technical_location_id = riu_equipment.technical_location_id JOIN
public.riu_type_equipment ON riu_type_equipment.type_equipment_id =
riu_equipment.type_equipment_id JOIN public.riu_catalog_profiles ON
riu_catalog_profiles.catalog_profiles_id = riu_equipment.catalog_profiles_id JOIN
public.riu_class_equipment ON riu_class_equipment.class_equipment_id =
riu_type_equipment.class_equipment_id;
```

Las tablas involucradas en dicha consulta son riu_equipment, riu_technical_location, riu_type_equipment, riu_catalog_profiles y riu_class_equipment para un total de 50 000 tuplas.

Tabla 28. Resultados de las pruebas de carga.

Cantidad de usuarios concurrentes	Media (ms)	Mediana (ms)	Mínimo (ms)	Máximo (ms)
7	133	125	125	156
40	641	593	203	1219

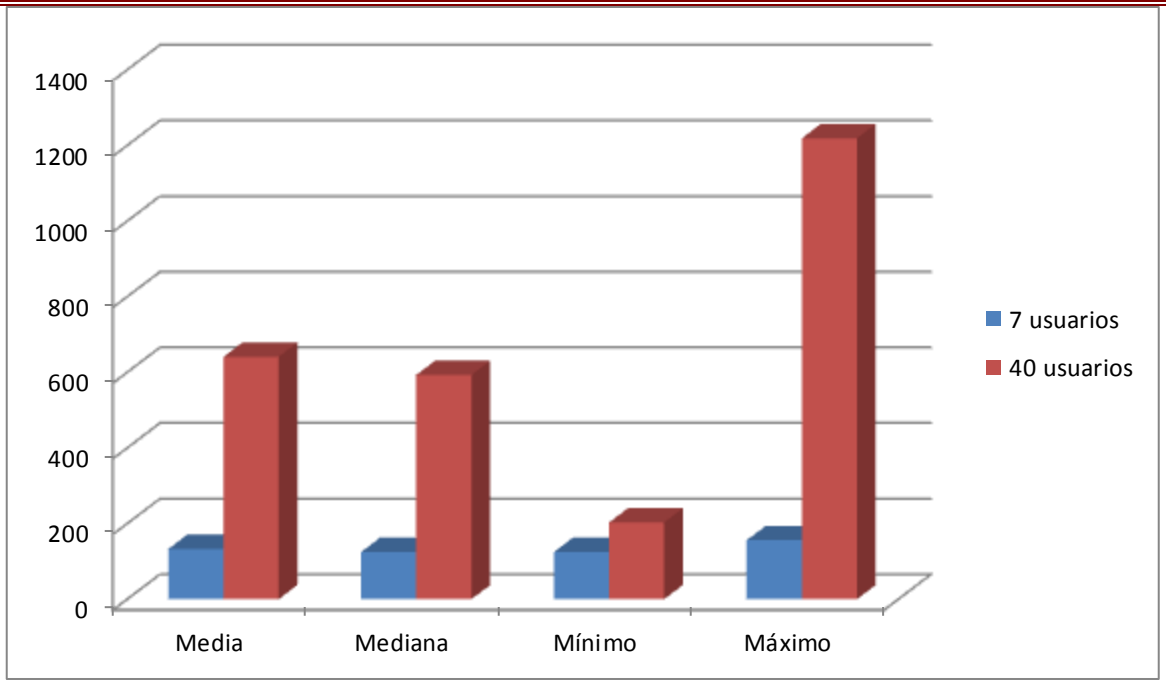

Figura 11. Gráfica de resultados de las pruebas de carga.

En la figura 11 se representa gráficamente la prueba de carga realizada. En el eje horizontal, se representan las cuatro variables a tener en cuenta en el resultado final, en el eje vertical los tiempos de respuestas en milisegundos y las barras representan la conexión concurrente de los usuarios.

Luego de realizadas las pruebas de carga se concluyó que las mismas arrojaron resultados satisfactorios tanto para 40 como para 7 usuarios concurrentes, ya que los tiempo de respuesta se encuentran dentro de los establecidos para la aplicación, evidenciándose de esta forma la eficiencia de la base de datos.

Conclusiones parciales

En este capítulo el diseño propuesto se validó teórica y funcionalmente para demostrar que el mismo ha sido construido correctamente y cumple con los requerimientos del sistema.

Con la validación teórica se comprobaron un conjunto de aspectos fundamentales como la integridad y la redundancia de información. Además se analizaron e implementaron mecanismos de seguridad a la base de datos.

Para la validación funcional se realizaron las pruebas de rendimiento, específicamente las pruebas de carga y volumen, permitiendo conocer el comportamiento de la base de datos ante una sobrecarga de información y grandes conexiones de usuarios concurrentemente.

Conclusiones

Con la realización de la presente investigación se ha dado cumplimiento al objetivo propuesto, arribando a la siguiente conclusión:

- El diseño y la implementación del modelo de base de datos del Sistema Integral de Confiabilidad Operacional favorece la integridad, organización e independencia de la información almacenada.

Recomendaciones

Al culminar el desarrollo del trabajo de diploma se recomienda:

- Realizar el diseño y la implementación del modelo de datos correspondiente a cada metodología restante del Sistema Integral de Confiabilidad Operacional, basándose en la propuesta de solución de la presente investigación.

Referencias Bibliográficas

1. *¿Qué es Confiabilidad Operacional?* **Durán, José Bernardo.** 2, s.l. : Revista Club Mantenimiento, 2000.

2. **Mota Herranz, Laura , Casamayor Ródenas, Juan Carlos and Celma Giménez, Matilde.** Bases de datos relacionales: teoría y diseño. 1994.

3. **Date, C. J.** Introducción a los Sistemas de bases de datos. 1993.

4. **Pérez Valdés , Damián.** Maestros del Web. [Online] 2007. [Cited: 10 22, 2011.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

5. **Mendoza Pacheco, Henry Jesus.** [Online] 2 10, 2008. [Cited: 10 22, 2011.] http://www.plusformacion.com/Recursos/r/Sistemas-gestion-bases-datos-0.

6. **Matos García, Rosa María.** Diseño de Base de datos. 1999.

7. **Gil, Fidel , Albrigo, Javier and Do Rosario, Javier.** Sistemas de Gestión de Bases de Datos. Valencia : s.n., 2005.

8. **Instituto Tecnologico de Villahermosa.** [Online] [Cited: 11 3, 2011.] http://www.mitecnologico.com/Main/Dise%F1oBasesDeDatos.

9. **Garcia, Edgar Ursulo.** [Online] 2008. [Cited: 11 2011, 5.] http://www.mitecnologico.com/Main/DefinicionModeloDeDatos.

10. **Rob, Peter and Coronel, Carlos.** Sistemas de bases de datos: diseño, implementación y administración. 2006.

11. Ecured. [Online] 11 7, 2011. http://www.ecured.cu/index.php/Base_de_datos#Bases_de_datos_jer.C3.A1rquicas.

12. **Anzaldo, J.** Breve historia de las bases de datos. [Online] 2005. [Cited: 11 10, 2011.] http://janzaldo.wordpress.com/2005/12/06/breve-historia-de-las-bases-de-datos.

13. **Pons, Olga, et al., et al.** Introducción a las bases de datos: el modelo relacional. 2005.

14. **The PostgreSQL Global Development Group.** PostgreSQL. [Online] [Cited: 11 22, 2011.] http://www.postgresql.org.es/sobre_postgresql.

15. Tecnologías Libres. [Online] 2008. [Cited: 4 15, 2012.] http://www.tecnologiaslibres.com/portal/content/view/19395/52/.

16. Ecured. [Online] [Cited: 11 20, 2011.] http://www.ecured.cu/index.php/Oracle_Designer.

17. Ecured. [Online] [Cited: 3 20, 2012.] http://www.ecured.cu/index.php/Oracle.

18. Ecured. [Online] [Cited: 12 10, 2011.] http://www.ecured.cu/index.php/Microsoft_SQL_Server.

19. Embarcadero. [Online] [Cited: 11 24, 2011.] http://www.embarcadero.com.pl/produkty/er-studio/ersent85/wymagania.shtml.

20. Free Download Manager. [Online] [Cited: 11 26, 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

21. Free Download Manager. [Online] 11 25, 2011. http://www.freedownloadmanager.org/es/downloads/SME_Gerente_de_PostgreSQL_37536_p/.

22. Guia Ubuntu. [Online] 2008. [Cited: 11 26, 2011.] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

23. Free Download Manager. [Online] [Cited: 4 28, 2012.] http://www.freedownloadmanager.org/es/downloads/Generador_de_Datos_de_SME_2005_para_PostgreSQL_37202_p/.

24. No Solo Unix. [Online] 2010. [Cited: 5 4, 2012.] http://www.nosolounix.com/2010/02/herramientas-test-software.html.

25. **Tedeschi, Nicolás.** MSDN. [Online] [Cited: 3 28, 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

26. **Marquina, Ernesto .** Guía de Patrones, Prácticas y Arquitectura .NET. 2008.

27. Ecured. [Online] [Cited: 1 20, 2012.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

28. **Savedra, J.** Integridad Informática enfocada a las bases de datos. 2009.

29. **Silberschatz, Abraham.** Fundamentos de Bases de Datos. Cuarta. 2002.

30. **Microsoft.** [Online] [Cited: 4 2, 2012.] http://technet.microsoft.com/es-es/library/ms184276%28v=sql.105%29.aspx.

31. **Corporación Sybven.** [Online] 2012. [Cited: 5 5, 2012.] http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246.

Anexos

Anexo 1 Configuración de la herramienta JMeter

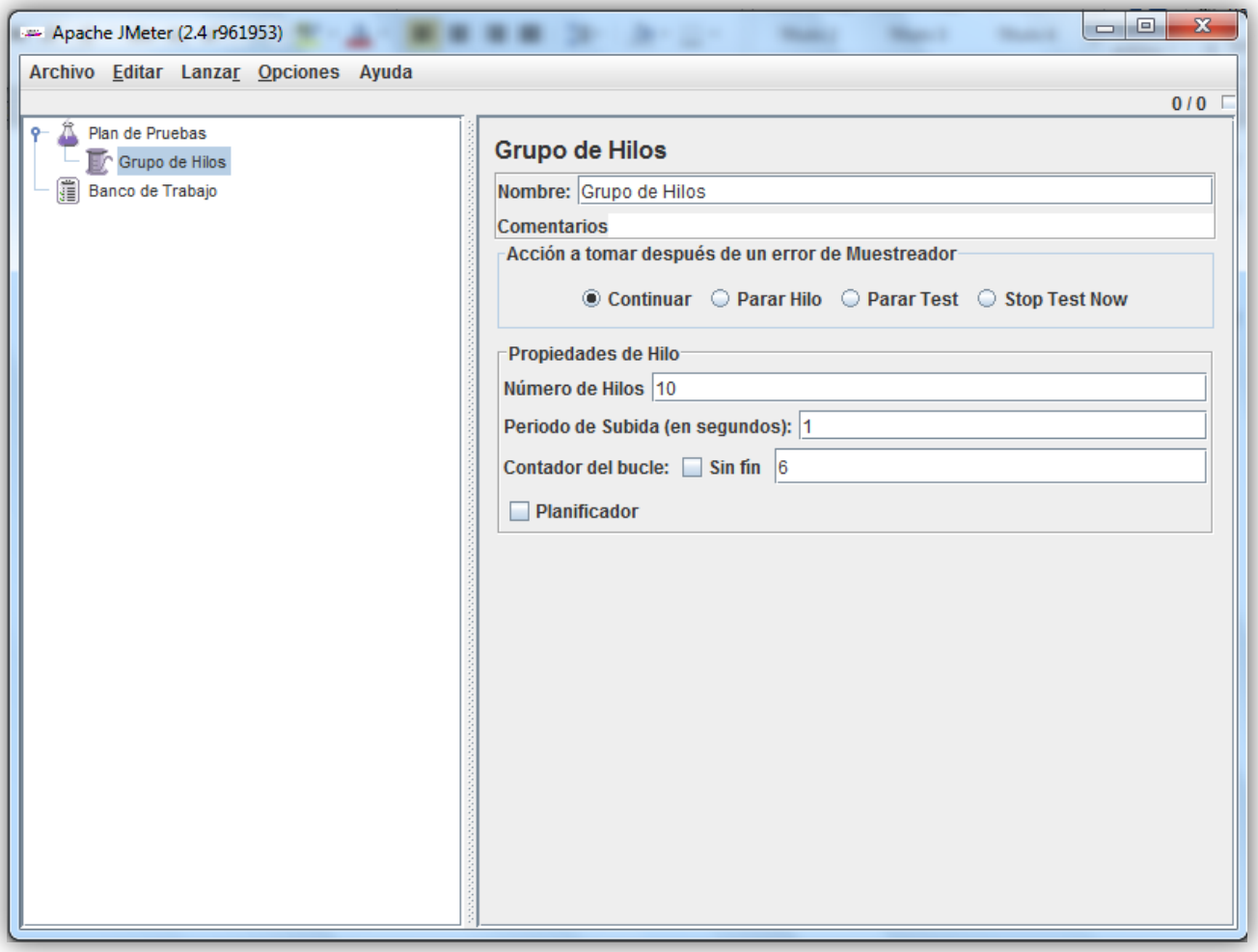

Figura 12. Creación de grupos de hilos

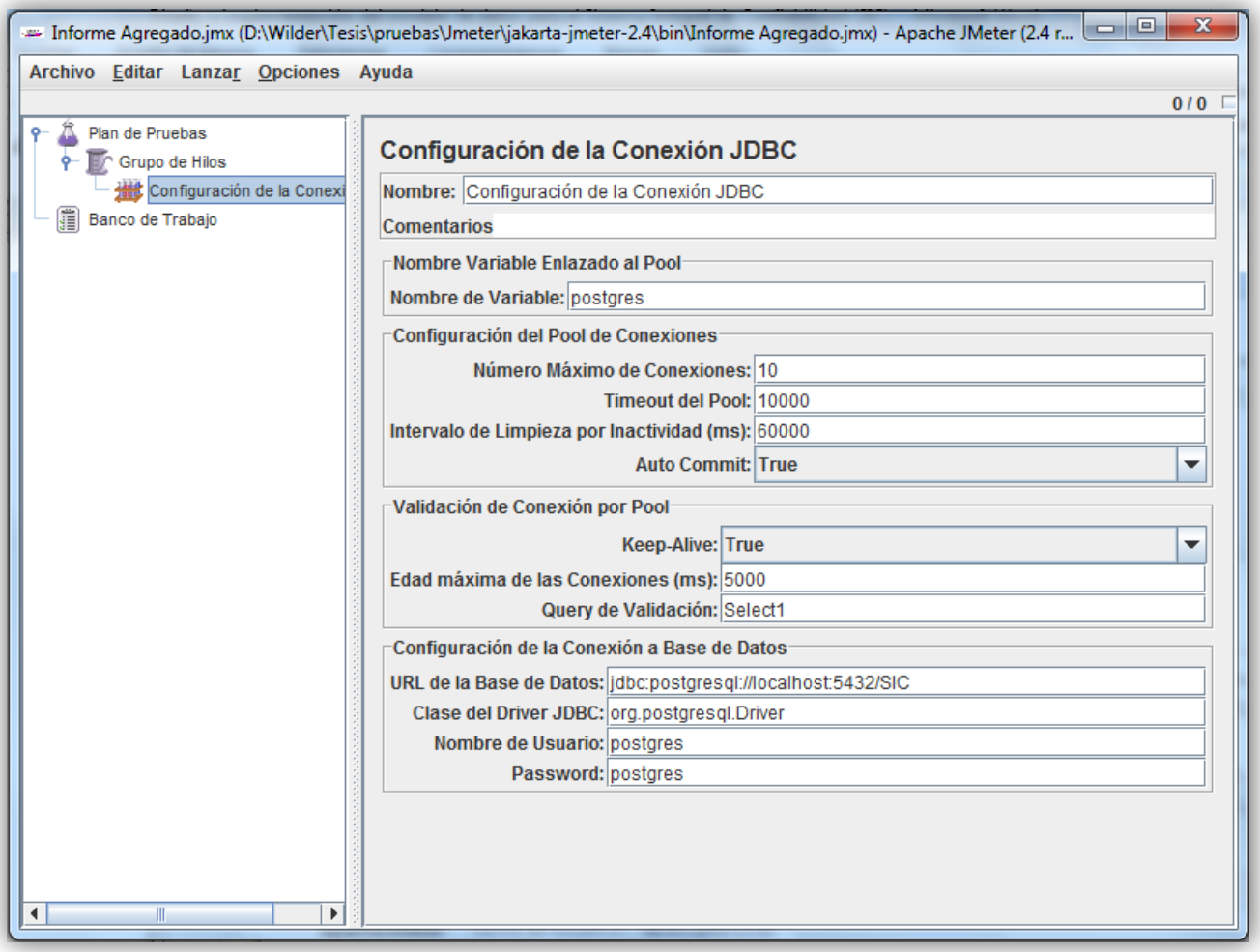

Figura 13. Configuración del acceso a la base de datos

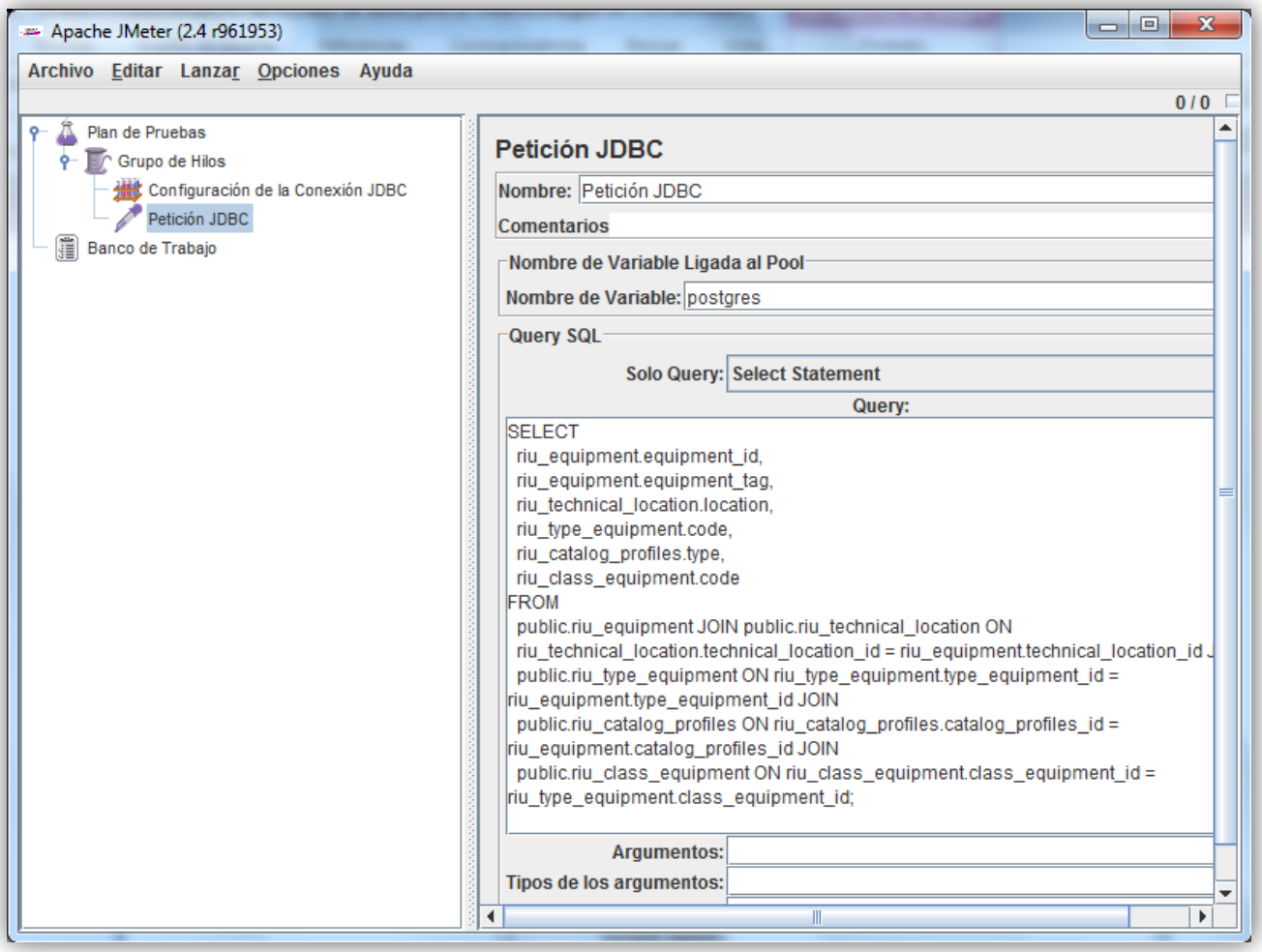

Figura 14. Configuración de la petición JDBC