Universidad de las Ciencias Informáticas
Facultad Regional Mártires de Artemisa

Título: Desarrollo de la base de datos para el módulo de la Dirección de Agricultura de la Administración Provincial de Artemisa.

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

Autor: Greydis Esther Vazquez Torralbas

Tutor: Ing. Yenisleydis Rodríguez Martínez

Co-Tutor: Ing. Ulises Socas Riesgo

Artemisa 2012

"Año del 54 aniversario de la Revolución."

"Queda prohibido no sonreír a los problemas, no luchar por lo que quieres, abandonarlo todo por miedo, no convertir en realidad tus sueños..."

Pablo Neruda.

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y consedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____	_____
Firma Autor	Firma Tutor
Greydis Esther Vazquez Torralbas	Ing. Yenisleydis Rodríguez Martínez

AGRADECIMIENTOS

A mis padres por su entrega incondicional y dedicación, por su apoyo y por el amor que siempre me han dado.

A mi familia, por apoyarme en todo momento.

A Oniel, por quererme como soy, apoyarme, ayudarme y aconsejarme siempre.

A Eylin, por estar siempre presente en los 5 años y medio que duró nuestra carrera.

A Diosmel, por creer siempre en mí.

A los profesores que me han impartido clases a lo largo de mi carrera, que de una forma u otra han contribuido en mi formación.

Agradezco el haber tenido la posibilidad de estudiar en esta facultad, porque he conocido a muchas personas maravillosas, y como dicen algunas personas, los amigos de la universidad son para toda la vida.

Un agradecimiento muy especial a todas las personas que esperaron junto a mí este gran momento, a aquellos que pasan por nuestra vida y dejan una huella imborrable, a los que viven muy lejos y a los que ya no están.

Le agradezco a todos, porque gracias a ustedes me he convertido en la profesional que soy hoy.

Gracias.

DEDICATORIA

A mis padres que son la luz que me ilumina.

A mi familia que son mi sostén y especialmente a mi abuela, que no se encuentra hoy conmigo.

RESUMEN

En la actualidad el avance que se produce en temas de informática y debido a sus enormes beneficios hace que las instituciones busquen alternativas para que su desempeño sea más eficiente y cómodo. Por lo que la Facultad regional de Artemisa desarrolla entre sus proyectos el Sistema informativo para el Gobierno (SIGOB), el cual gestionará todos los procesos de la Administración Provincial de Artemisa. Dichos procesos generan gran cantidad de datos, por lo que se necesita una base de datos para el manejo de la información que se gestiona, siendo el objetivo de esta investigación: realizar el diseño e implementación de la base de datos del Módulo de Agricultura de la Administración Provincial de Artemisa, que garantice el correcto almacenamiento y recuperación de los datos. Para dar cumplimiento al mismo, se analizaron los principales conceptos referentes a las bases de datos, los enfoques de los Sistemas Gestores de Bases de Datos (SGBD), así como su evolución y estructura. Se tuvo en cuenta el proceso de normalización como técnica de diseño de bases de datos, obteniéndose el modelo de datos, el cual se creó a partir del estudio de los requisitos funcionales. También se implementaron en la capa de acceso a datos, los métodos necesarios para tener acceso a los datos una vez implantada la base de datos en el SGBD PostgreSQL. Así mismo se validó la integridad de los datos, la calidad del dato y el alto rendimiento de la solución propuesta.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el avance vertiginoso del comercio, surge la administración de datos, cuando en Sumeria, tuvieron la necesidad de crear un sistema para registrar sus transacciones comerciales. Al inicio grababan con un punzón de caña, en tablas de arcilla, sencillas representaciones de objetos, denominadas pictografías. Los datos importantes se conservaban en tablas cocidas al horno.

Desde el principio de la civilización se ha vivido la importancia de organizar y almacenar grandes cantidades de información para después recuperarla y utilizarla de forma eficaz.

El nacimiento de las Bases de Datos (BD) se manifiesta con la invención de la Máquina Perforadora Herman Hollerith (1860 - 1929) quien fue denominado el primer ingeniero estadístico de la historia, ya que creó la primera computadora llamada “Máquina Automática Perforadora de Tarjetas”, con el objetivo de llevar a cabo el censo de Estados Unidos correspondiente al año 1880. (Daixauli, y otros)

La comercialización de las BD iniciaron a partir del año 1960, pero su disponibilidad fue limitada. Antes de su aparición existía un elevado nivel de redundancia en los sistemas de procesamiento de datos.

Desde la primera aplicación de las computadoras para el manejo de datos en los sesenta, la cantidad de datos recolectados así como la generación de información, ha crecido a pasos acelerados. El empleo de computadoras en el manejo de BD se ha generalizado desde grandes corporaciones a los pequeños negocios incluso en el manejo del hogar.

Esto se debe principalmente a la reducción del costo de las computadoras y a la disponibilidad de software fácil de usar, donde se puede observar la información en forma de gráficas, mapas y otras imágenes además del despliegue tradicional de texto y tablas.

Una Base de Datos (BD) es una colección de informaciones que están almacena-

das en una computadora.

Cada BD tiene una estructura, que es el modelo básico o formato en que guarda la información, que tiene un propósito particular en el conjunto de los datos. Permite recuperar cualquier clase de información: referencias, documentos textuales, imágenes, datos estadísticos, etc. (AARAON, 2011)

En el futuro de las BD, más que de su evolución como alternativa, se habla de la evolución de sus sistemas gestores, ya que la BD en sí, es la manera en que se introducen los datos mientras que los Sistemas Gestores de Base de Datos (SGBD) son los que acceden a ellas y las implementan, desde fuera. Estas están más orientadas al ámbito científico, industrial y comercial.

En tal sentido se puede afirmar que en el mundo actual existe cada vez una mayor demanda de datos, demanda que ha sido patente en empresas y sociedades, pero en la actualidad se ha disparado aún más, debido al acceso multitudinario a la Internet, por ello las BD se reconocen como una de las principales aplicaciones de la informática.

Nuestro país en el afán de estar acorde a las exigencias tecnológicas del mundo actual, creó la UCI (Universidad de las Ciencias Informáticas) en la que, específicamente en la Facultad Regional “Mártires de Artemisa”, se desarrollan diversos proyectos que vinculan la docencia y la producción, con el fin de su comercialización dentro y fuera del país, y para el beneficio interno de la Universidad.

El departamento de Aplicaciones Informáticas se encuentra enfrascado en el desarrollo de uno de ellos, nombrado “Sistema informativo para el Gobierno”(SIGOB), el cual se basa en el desarrollo de un producto para gestionar toda la información de la Administración Provincial de Artemisa que cuenta con 32 direcciones provinciales entre las cuales se encuentran la Dirección de Agricultura. La misma se encarga de gestionar toda la información con respecto a la existencia,

la venta, la preparación de tierras por temporadas y la demanda de productos esenciales para la alimentación de la población como lo son las viandas, las hortalizas, los granos, las frutas, los cítricos y la leche de cada entidad de la provincia. Además en cuanto a la ganadería se tramitan reportes sobre el plan de los mataderos y sobre la siembra para el ganado vacuno y porcino.

El proceso de almacenamiento de toda esta información se realiza de forma manual, en formato duro o digital, lo que trae consigo demora en la tramitación de los datos, provocando que en ocasiones no se tenga constancia real del cumplimiento de la planificación diaria de la venta de todos los alimentos agrícolas que se producen en la provincia y de la relación verdadera de cada cosecha por temporada, así como su pérdida y duplicado. Los reportes estadísticos que se generan son poco fiables por lo difícil que resulta recopilar un gran volumen de información al transcurrir largos períodos de tiempo. Todas estas dificultades ocasionan problemáticas para la generación de reportes inmediatos y para la búsqueda de los datos referente a los pastos del MINAG y de los cultivos varios, así como su entrega al Ministerio de la Agricultura, en el tiempo de entrega determinado y con la calidad requerida. La entrega incorrecta y tardía de esta información afecta la toma de decisiones para la Provincia Artemisa con respecto a la alimentación agrícola. Todo lo expuesto anteriormente engloba la **situación problemática.**

Mediante el análisis de la situación planteada anteriormente surge como **problema de la investigación** ¿Cómo contribuir a la integridad y disponibilidad de la información en los procesos de almacenamiento de datos, en la Dirección de Agricultura de la Administración Provincial de Artemisa?

A raíz del problema de la investigación se define como **objeto de estudio** bases de datos relacionales, delimitando como **campo de acción** base de datos relativa a los procesos de la Agricultura en el Sistema Informativo de la Administración Provincial de Artemisa.

INTRODUCCIÓN

El **objetivo general** de la investigación es**:** Desarrollar una base de datos que contribuya a elevar los grados de integridad y disponibilidad de los datos en el proceso de almacenamiento de estos en la Dirección de Agricultura de la Administración Provincial de Artemisa.

Para complementar este objetivo general se definen los siguientes **objetivos específicos:**

1. Elaborar la Fundamentación Teórica de la investigación.
2. Diseñar la base de datos para el módulo de la Dirección de Agricultura.
3. Implementar la capa de acceso a datos para el módulo de la Dirección de Agricultura.
4. Validar los resultados obtenidos.

Para guiar la presente investigación se formula como **idea a defender**: Con el desarrollo de la base de datos para el módulo de la Dirección de Agricultura de la Administración Provincial de Artemisa, se contribuirá a elevar los grados de integridad y disponibilidad de los datos.

Se identifica como **variable independiente** base de datos para el módulo de la dirección de Agricultura y como **variables dependientes** aumentar los grados de integridad y disponibilidad de los datos.

Para dar solución a los objetivos específicos se plantean las siguientes **tareas de la investigación:**

1. Definición del marco teórico y elaboración del estado del arte de la investigación.
2. Definición de la metodología, las herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta.
3. Diseño de la base de datos para el módulo de la Dirección de Agricultura.
4. Implementación de la capa de acceso a datos para el módulo de la Dirección de Agricultura.

5. Realización de pruebas funcionales para validar los resultados obtenidos.

Para guiar el proceso de desarrollo de la investigación se utilizaron los siguientes **métodos teóricos:**

- **Análisis histórico-lógico:** permitió analizar las características de los SGBD que se llevan a cabo para la creación de los distintos modelos existentes, realizar un estudio histórico de los mismos y conocer las tendencias actuales en cuanto al diseño e implementación de una capa de acceso a datos, lo que permitió valorar y determinar lo más factible a usar para la presente investigación.

- **Inductivo-deductivo:** a partir del planteamiento de la idea a defender y siguiendo la lógica de deducción se llega a nuevos conocimientos y predicciones que son sometidos a verificaciones. Se deduce que con la realización del diseño e implementación de la capa de acceso a datos para la gestión de la información, se garantizará la integridad de los datos y el acceso a la totalidad de la información gestionada en la Dirección de Agricultura del Consejo de la Administración Provincial Artemisa.

- **Modelación:** este método se utilizó para diseñar la BD, pues se debe definir un modelo de datos que cumpla con los requisitos necesarios.

- **Analítico – sintético:** Permite dividir el fenómeno a estudiar y unir las partes previamente analizadas. Se analizaron los aspectos más importantes para administrar una base de datos para posteriormente sintetizar y exponer los resultados obtenidos.

Dentro de los **métodos empíricos** se utilizó:

- **Análisis documental:** Permitió el estudio y análisis de todos los documentos e informes de la dirección de Agricultura, para obtener un mejor entendimiento de los procesos de la misma.

INTRODUCCIÓN

La **población** seleccionada para esta investigación, es la Dirección de Agricultura de la Administración provincial de Artemisa que cuenta con un total de 12 personas. Para una **muestra** de 12 persona, lo que representa un 100%.

Se pretende tener como **aporte práctico:** Una BD para el módulo de la Dirección de Agricultura, la cual brindará grandes beneficios a la Administración Provincial de Artemisa, ya que almacenará toda la información requerida para la gestión de los procesos en dicha dirección. La BD permitirá el acceso a la información en todo momento, para manejar los datos y actualizar los cambios que se realicen.

La BD en conjunto con la aplicación web, llevarán a cabo el cumplimiento eficiente de todas las funcionalidades que se manejan en cada uno de los procesos de la dirección de Agricultura.

El presente documento está estructurado en tres capítulos los cuales contienen la siguiente información:

Capítulo 1. Fundamentación teórica: Se hace un estudio de los diferentes sistemas de gestión de información, se definen y caracterizan herramientas y la metodología que se utilizarán en la propuesta de solución. Contempla los conceptos fundamentales del tema y también se mencionan los gestores de BD más populares.

Capítulo 2. Diseño y arquitectura de la base de datos: En este capítulo se brinda información acerca de las técnicas de diseño de BD, se analiza el modelo lógico y físico de la BD para la gestión de la información de la agricultura y para definir las características del sistema: se realiza el diseño con metáforas.

Capítulo 3. Implementación y validación de la capa de acceso a datos: En este capítulo se describe la implementación del diseño de la capa de acceso a datos y cómo se realiza la integración con la aplicación en la capa de persistencia. Se valida el diseño realizado de forma teórica teniendo en cuenta aspectos como la integridad, la normalización, la seguridad y cómo responde el sistema a las pruebas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

Introducción

En el presente capítulo la autora realiza un breve estudio de los sistemas de gestión de información. Define y caracteriza las herramientas y la metodología que se utilizarán en la propuesta de solución. Contempla los conceptos fundamentales del tema y también se mencionan los gestores de bases de datos más populares.

1.1. Conceptos fundamentales del tema

Bases de Datos

Las BD son identificadas por muchos autores por diferentes definiciones:

1. *¨Es posible considerar a la base de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos.¨* (Christopher J. Date, 2003)
2. *¨ Las BD son un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo.¨* (Rosa M. Matos, 2007)

La autora se afilia al concepto dado por Rosa M. Matos. En fin se puede decir que una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. A continuación se presentan algunos conceptos relacionados, definidos por Christopher J. Date.

Integridad: Conservar la seguridad en un sistema en que se permite a múltiples usuarios el acceso al sistema y compartir la BD.

Normalización: Es la transformación de vistas de usuarios complejas y almacenes

de datos a un conjunto de estructuras de datos estables más pequeñas además de ser más simples y más estables, las estructuras de datos normalizadas son más fáciles de mantener.

Redundancia: La redundancia se obtiene cuando las unidades de datos (palabras, bloques, etc.) se expanden para proporcionar más información que la estrictamente necesaria. Puede verificarse la consistencia interna de las unidades de datos, si se encuentran que están en error, pueden corregirse.

Consistencia: La ejecución aislada de la transacción (es decir, sin transacción que se ejecute concurrentemente) conserva la consistencia de la BD.
Se refiere a la cantidad de cambios que puede haber en la BD, para ello dichos cambios no deben de afectar a la BD para que haya consistencia.

Migración: Tiene una orientación de ruta por el hecho de que la tecnología puede pasar de una etapa o nivel a otro sin revisión.

Independencia: Significa que la tecnología no tiene que depender de otras instalaciones para operar y que esté libre de restricción técnica.

Seguridad: Describe la protección contra los sistemas de seguridad y su contenido. Se toma muy en cuenta la seguridad física porque sin ella no se llega a ninguna parte.

Entidad: La entidad es un objeto que existe y que se distingue de otros por sus características. Las entidades pueden ser de dos tipos:

- Tangibles: Son todos aquellos objetos físicos que podemos ver tocar o sentir.
- Intangibles: Todos aquellos eventos u objetos conceptuales que no podemos ver, aun sabiendo que existen.

Atributo: Se le llama así a las características que tienen las entidades en BD.

Relación: Es la característica que tienen las entidades en BD de asociarse o relacionarse con más entidades a través de las relaciones.

Llaves: La distinción de una entidad entre otra se debe a sus atributos, lo cual lo hacen único. Una llave primaria es aquel atributo el cual se considera clave para la identificación de los demás atributos que describen a la entidad.

Claro que puede haber más de un atributo que pueda identificarse como llave primaria en este caso se selecciona la que se considera más importante, los demás atributos son denominados llaves secundarias.

Capa de acceso a datos.

Una capa de acceso de datos debe de ser rápida, eficiente y transparente al desarrollador de aplicaciones.

¨Sirve como puente entre la capa lógica de negocio y el proveedor de datos. Esta capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Para que si cambia el proveedor de datos solo se necesitará cambiar en una sola capa el proveedor de datos.¨ (Manolo Herrera, 2007)

La capa de acceso a datos incluye todo el código específico del origen de datos subyacente. Incluye código que crea una conexión a la base de datos y que emite los comandos Select, Insert, Update y Delete. Normalmente, la capa de acceso a datos contiene clases que implementan los métodos para tener acceso a los datos de la base de datos subyacente.

La capa de presentación no trabaja directamente con datos. En su lugar, invoca clases y métodos en la capa de acceso a datos para todas las solicitudes de datos. Puede personalizar estas clases con su propia lógica de negocios.

En resumen de la capa de datos podemos decir:

- No guarda el estado (stateless).
- Sus parámetros deben de ser Entidades de Negocio (Clases que si tiene el valor de cada propiedad, ejemplo: Cliente, Factura, nomina, etc.).

- Los métodos más comunes se denominan CRUD (Crear, Leer, Actualizar o Eliminar) y realizan operaciones elementales en el proveedor de los datos.
- No debe de existir ninguna validación ni operaciones complejas en esta capa solo las pertinentes para operar con el proveedor de los datos, como es el manejo de los nulos en los parámetros y en los datos que se reciben del proveedor. (Herrera, 2007)

Sistema Gestor de Bases de Datos

¨Se puede afirmar que el software que permite la utilización y/o actualización de los datos en una o varias BD, desde diferentes puntos de vista a la vez, por uno o varios usuarios, se denomina Sistema Gestor de Base de Datos.¨ (Date, 2003)

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

Control de la redundancia: la redundancia puede tener efectos negativos o no deseados, pues puede provocar inconsistencias en los datos o duplicar el trabajo al actualizar.

Independencia: la independencia de los datos consiste en la capacidad de modificar el esquema lógico o físico de una BD sin tener que realizar cambios en las aplicaciones que sirven de ella.

Seguridad: se debe garantizar que la información se encuentra segura, estableciendo los permisos de acceso y autorización a los usuarios y grupos correspondientes.

Abstracción de la información: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una BD ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Redundancia mínima: Un buen diseño de una BD logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad: La información almacenada en una BD puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD

debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias. (CAVSI, 2010)

1.2. Integridad de datos

El término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos se modifican con sentencias *INSERT*, *DELETE* o *UPDATE*, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.

Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna un vendedor a una oficina no existente. Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía. Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender.

Una de las funciones importantes de un sistema gestor de base de datos (SGBD) relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

1.2.1. Tipos de restricciones para lograr integridad

- **Datos Requeridos**: establece que una columna tenga un valor no NULL. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.
- **Chequeo de Validez**: cuando se crea una tabla cada columna tiene un tipo de datos y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

- **Integridad de entidad**: establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla; si no, la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El SGBD comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallará.
- **Integridad referencial**: asegura la integridad entre las llaves foráneas y primarias (relaciones padre/hijo). Existen cuatro actualizaciones de la base de datos que pueden corromper la integridad referencial:
  - La inserción de una fila hijo se produce cuando no coincide la llave foránea con la llave primaria del padre.
  - La actualización en la llave foránea de la fila hijo, donde se produce una actualización en la clave ajena de la fila hijo con una sentencia UPDATE y la misma no coincide con ninguna llave primaria.
  - La supresión de una fila padre, con la que, si una fila padre -que tiene uno o más hijos- se suprime, las filas hijos quedarán huérfanas.
  - La actualización de la llave primaria de una fila padre, donde si en una fila padre, que tiene uno o más hijos se actualiza su llave primaria, las filas hijos quedarán huérfanas.

1.3. Disponibilidad de datos

Disponibilidad se refiere a la habilidad de la comunidad de usuarios para acceder al sistema, someter nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema se dice que está no disponible.

Disponibilidad de datos, que es el grado para el cual las bases de datos y otros sistemas de almacenamiento de la información que registran y reportan fielmente transacciones del sistema. Especialistas de gestión de la información

frecuentemente enfocan separadamente la disponibilidad de datos para determinar perdida de datos aceptable o actual con varios eventos de fracasos. Algunos usuarios pueden tolerar interrupciones en el servicio de aplicación pero no perdida de datos.

La disponibilidad de los datos en la empresa es muy importante ya que si no están disponibles, las aplicaciones no pueden funcionar y le ocasiona problemáticas a la empresa. Es trabajo del administrador de base de datos (ABD) que la Base de Datos este en línea y operando. (2012)

La disponibilidad es la condición donde un recurso dado puede ser accedido por sus consumidores. Además es el porcentaje de tiempo que un sistema puede ser usado en trabajo productivo. Si el rendimiento de una BD es demasiado pobre que los usuarios no pueden realizar su trabajo, la BD no está disponible. (2012)

La disponibilidad tiene 4 características. (2012)

Manejabilidad: la capacidad de crear y mantener un entorno eficaz, que proporciona servicio a los usuarios.

Recuperabilidad: la capacidad para restablecer el servicio en caso de un fallo o error de componente.

Fiabilidad: la capacidad para prestar el servicio en los niveles especificados por un período determinado.

Facilidad de servicio: la capacidad de determinar la existencia de problemas, diagnosticar su causa (s), y la reparación de los problemas.

Causas de Indisponibilidad

La pérdida del centro de datos: Obviamente, los datos no estarán disponibles si el centro de datos se pierde debido a un desastre natural o algún otro tipo de catástrofe, éste es el peor de los problemas que se pueden suscitar. Para solucionar este problema se debe:

- Recrear el Centro de Datos.
- Actualizar datos.
- Conectar usuarios o aplicaciones.

Problemas de Red: Si un servidor de BD está en un conmutador de red única y está sujeto a un interruptor de parada, la base de datos no estará disponible.

Considerar la implementación de conmutadores de red redundante para evitar tales interrupciones. También se puede dar problemas de hardware en la red como tarjetas de red o switchs, para ésto se debe tener repuestos disponibles. Igualmente puede haber problemas en el Software de red. El ABD debe minimizar donde está el problema y llamar al orden de red. El ABD debe coordinar con otros especialistas entre ellos los de redes. (2012)

La pérdida de hardware del servidor: El hardware básico del servidor consiste en el CPU[11], memoria y los subsistemas de disco. Si la CPU se daña todo el sistema no funciona, una solución es un clúster[12]. Otra solución puede ser la replicación a un servidor secundario. La rapidez con que se soluciona el problema depende del hardware y software. Cuando la RAM falla el sistema no funciona y si el daño es parcial el rendimiento es afectado directamente, una solución para esto es reponer las memorias RAM. (2012)

Si el servidor de Base de Datos se pierde o daña se debe reemplazar el servidor, cargar el Sistema Operativo (SO), SGBD, BD y configurarlos. El ABD se asegura de tener fuentes alternativas de energía como UPS[13], repuestos, etc. (2012)

Garantizar la disponibilidad

Frente a presupuestos reducidos y recursos, y un volumen cada vez mayor de datos para la gestión, las organizaciones de TI necesitan para evaluar sus necesidades críticas y poner en práctica una serie de pasos clave estratégicos para asegurar la disponibilidad. Una buena estrategia podría incluir medidas para:

- Realizar mantenimiento de rutina mientras que los sistemas siguen funcionando.
- Automatizar las funciones de ABD.
- Aprovechar las características del SGBD que promuevan la disponibilidad.
- Aprovechar las tecnologías de hardware.

Realizar mantenimiento de rutina mientras que los sistemas siguen funcionando Los ABD's necesitan herramientas que reducen el tiempo de mantenimiento de horas a minutos o ninguno en absoluto, mientras que a los usuarios permitiendo el acceso continuo a los datos que necesitan para hacer su trabajo. Algunos SGBD proporcionan funciones integradas para realizar algunas tareas de mantenimiento mientras la base de datos está disponible. (2012)

Realizar mantenimiento de rutina mientras que los sistemas siguen funcionando Existen herramientas de terceros que pueden realizar las siguientes tareas en caliente.

- Reorganizar la Base de Datos.
- Backups.
- Recuperación de Datos.
- Carga y descarga de datos.
- Análisis estadístico.
- Chequeo de integridad.

1.4. Tipos de Bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

1.4.1. Según la variabilidad de los datos almacenados

Bases de datos estáticas

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almace-

nar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones. (EcuRed, 2010)

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.(EcuRed, 2010)

1.4.2. Según el contenido

Bases de datos bibliográficas

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no, estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras. (EcuRed, 2010)

Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas. (EcuRed, 2010)

Directorios.

Un ejemplo son las Guías telefónicas en formato electrónico. (EcuRed, 2010)

Bases de datos o "bibliotecas" de información de química o biológica.

Son BD que almacenan diferentes tipos de información proveniente de la Química, las Ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- Las que almacenan secuencias de Nucleótidos o Proteínas.
- Las bases de datos de rutas metabólicas.
- Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de Biomoléculas.
- Bases de datos clínicas.
- Bases de datos bibliográficas (Biológicas, Químicas, Médicas y de otros campos): PubChem, Medline, EBSCOhost.(EcuRed, 2010)

1.5. Clasificación de los SGBD

Jerárquico

Éstas son BD que almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol, en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Una de las principales limitaciones de este modelo, es su incapacidad de brindar eficientemente una solución a la redundancia de datos. (Hansen, 1997)

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

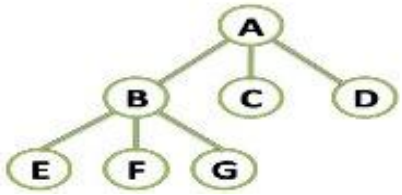

Figura 1. Estructura de una base de datos jerárquica.

En red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). (Hansen, 1997)

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una BD de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

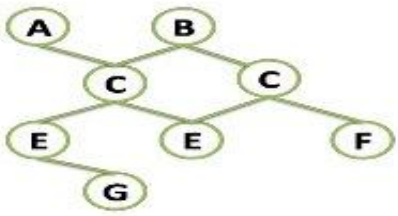

Figura 2. Estructura de una base de datos en red.

Relacional

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de entidades (tablas), compuestas por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla). (Hansen, 1997)

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

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

la información.

Figura 3. Estructura de una base de datos relacional.

Por su Independencia física y lógica, flexibilidad, uniformidad y sencillez, será el modelo relacional el que se utilizará en el presente trabajo.

1.6. Arquitectura de los sistemas de base de datos

La arquitectura de un sistema de bases de datos está influenciada en gran medida por el sistema informático subyacente en el que se ejecuta, en particular por aspectos de la arquitectura de la computadora como la conexión en red, el paralelismo y la distribución. Ejemplo de esto, la arquitectura ANSI/SPARC.

El objetivo principal de la arquitectura ANSI/SPARC es definir un SGBD con el máximo grado de independencia, separando las aplicaciones de usuario y la base de datos física. Para ello se utilizan tres niveles de abstracción conocidos como:

- **nivel físico**: es el nivel más bajo de abstracción y el nivel real de los datos almacenados. Este nivel define cómo se almacenan los datos en el soporte físico, ya sea en registros o de cualquier otra forma, así como los métodos de acceso. Este nivel lleva asociada una representación de los datos, que es lo que denominamos Esquema Físico.

- **nivel conceptual**: es el correspondiente a una visión de la BD desde el punto de visto del mundo real. Es decir se trata con la entidad u objeto representado, sin importar como está representado o almacenado éste. Es la representación de los datos realizada por la organización, que recoge los datos parciales de los requerimientos de los diferentes usuarios y aplicaciones parciales. Incluye la definición de los datos y las relaciones entre ellos. Este nivel lleva asociado un Esquema Conceptual.

- **nivel de visión**: son partes del esquema conceptual. El nivel conceptual presenta toda la base de datos, mientras que los usuarios, por lo general, sólo tienen acceso a pequeñas parcelas de ésta. El nivel visión es el encargado de dividir estas parcelas. Un ejemplo sería el caso del empleado de una organización que tiene acceso a la visión de su nómina, pero no a la de sus compañeros. El esquema asociado a éste nivel es el Esquema de Visión.

Otros autores utilizan la denominación de nivel interno, nivel conceptual y nivel externo, para referirse a estos mismos niveles:

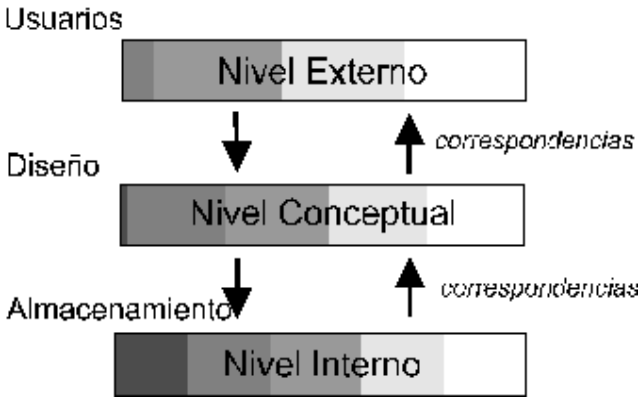

Figura 4. Niveles de la arquitectura de un sistema de base de datos.

Este modelo de arquitectura permite establecer el principio de independencia de los datos, ya se trate de una independencia lógica o física. La independencia lógica significa que los cambios en el esquema lógico no deben afectar a los esquemas externos que no utilicen los datos modificados; la independencia física significa que el esquema lógico no se va a ver afectado por los cambios realizados en el esquema interno, correspondientes a modos de acceso, etc. (Lapuente, 2010)

1.7. Sistemas Gestores de Base Datos más destacados

Rapidez, efectividad en los procesos y los grandes flujos de información están como primera necesidad la hora de optimizar servicios y productos. Ante esta notable demanda de soluciones informáticas han surgido multitud de gestores de bases de

datos, siendo estos programas que permiten manejar la información de modo sencillo y que prestan servicios para el desarrollo y el manejo de bases de datos.

Es oportuno destacar que los SGBD que más se destacan son Oracle, Microsoft SQL Server y Borland Interbase que comercialmente son los más fuertes, sin embargo en el mundo del software libre, se aprecian opciones tan completas como MySQL y PostgreSQL. A continuación se exponen algunas de sus características. (Carrasco, 2011)

1.7.1. Software bajo licenciamiento comercial

Oracle

Sin duda alguna la actual unión entre Dell y Oracle constituye uno de los principales encuentros tecnológicos al servicio de las necesidades empresariales actuales, tras alcanzar más de 22.000 instalaciones de software Oracle en equipo Dell, las empresas han demostrado un sólido éxito en la tarea de entregar mayor beneficio empresarial a una amplia gama de clientes, entre ellos el Lighting Group de Acuity Brands, Electronic Arts, Menasha Corporation, el Centro Mercedes-Benz de Ayuda al Cliente y Precisión Response Corporación.

Las propuestas de precio fijo incluyen: servicios de migración para los clientes que proceden de UNIX a Linux, servicios de implementación para ayudar a los clientes a desplegar rápidamente Oracle Database con Real Application Clusters; afinamiento del rendimiento y de la capacidad, así como replicación en espejo de las bases de datos y planificación de la recuperación de emergencia. (Sedano, 2010)

Sql Server 2000

Para la autora es importante destacar por su importancia, que Sql Server es el sistema de gestión de base de datos representativa de la firma mundialmente conocida Microsoft. En la actualidad, las compañías demandan una clase diferente de solución de base de datos. El rendimiento, la escalabilidad y la confiabilidad son

esenciales y la anticipación al mercado es crítica. Aparte de estas cualidades empresariales fundamentales, SQL Server 2000 proporciona agilidad a sus operaciones de análisis y administración de datos al permitir a su organización adaptarse rápida y fácilmente para obtener ventaja competitiva en un entorno de cambios constantes.

1.7.2. Software libre

MySQL Server

Por otra parte, MySQL Server es la BD de código fuente abierto más usada del mundo desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

Su origen se debió a la búsqueda por parte de los fundadores de crear un manejador de BD que fuera "rápido", todavía más rápido que mSQL. Así surgió MySQL, primero como un producto de la empresa y después como software de dominio público.

El servidor MySQL fue desarrollado originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y ha estado siendo usado exitosamente en ambientes de producción sumamente exigentes por varios años. (Sedano, 2010)

Aunque se encuentra en desarrollo constante, el servidor MySQL ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet.

PostgreSQL

Con la salida al mercado de múltiples entornos de desarrollo la preocupación está en conocer las características, ventajas y desventajas de cada herramienta que ofrece el mercado, y en el caso específico del desarrollo de este trabajo se da a

conocer características generales de PostgreSQL.

PostgreSQL es un descendiente Open Source. Soporta una gran parte del SQL estándar y ofrece muchas características modernas:

- Consultas complejas.
- Claves foráneas.
- Disparadores.
- Vistas.
- Integridad transaccional.
- Control de concurrencia (multiversión).

También, PostgreSQL puede ser ampliado por el usuario de muchas formas, por ejemplo, añadiendo nuevos Tipos de datos, Funciones, Operadores, Métodos de indexación y Lenguajes procedurales.

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 afecta al resto y el sistema continúa funcionando.

El 12 de septiembre del 2011, el Equipo Global de Desarrollo de PostgreSQL da a conocer el lanzamiento de PostgreSQL 9.1. Esta última actualización de la BD de código abierto líder ofrece tecnología innovadora, extensibilidad sin igual, y nuevas características como replicación sincrónica, método de indexado "K-Nearest Neighbor", y conectores de datos foráneos. (PostgreSQL, 2011)

La versión 9.1 ofrece muchas características que los usuarios han estado solicitando por años, retirando obstáculos para el despliegue de aplicaciones nuevas o migradas en PostgreSQL. Estas incluyen:

- **Replicación Sincrónica**: permitiendo alta disponibilidad con consistencia sobre múltiples servidores.
- **Regionalización por columna**: soportando correctamente el ordenamiento

por lenguaje en las bases de datos, tablas o columnas.

- **Tablas unlogged**: importante incremento del rendimiento para datos efímeros.

También incluye muchas características que son nuevas en la industria de las BD, como por ejemplo:

- **Indexamiento de los K vecinos más próximos (K-Nearest-Neighbor)**: índices sobre "distancia" para consultas rápidas de ubicación y búsquedas de texto.
- **Nivel de Aislamiento Serializable a través de "Snapshots"**: mantiene consistentes multiples transacciones concurrentes sin el uso de bloqueos, usando "verdadera serialización"
- **Writeable Common Table Expressions**: ejecuta actualizaciones multi-fases complejas en una simple consulta
- **Security-Enhanced Postgres**: despliega seguridad de nivel militar y control de acceso mandatorio

La extensibilidad de PostgreSQL permite a los usuarios añadir nuevas funcionalidades en bases de datos que corren en producción, y usar estas para tareas que no puede realizar ningún otro sistema de base de datos. La versión 9.1 añade nuevas herramientas de extensibilidad, incluyendo:

- **Conectores de datos foráneos**: añada y consulte fuentes de datos externas desde PostgreSQL
- **Extensiones**: cree, cargue, y administre fácilmente nuevas características de la base de datos.

Ventajas:

- Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas.

- Usa una arquitectura proceso por usuario cliente-servidor: hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectarse a PostgreSQL.
- Altamente extensible: soporta los tipos de datos base y otros como los de fechas, monetarios, elementos gráficos, datos sobre redes (MAC, IP) y cadenas de bits. Además operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Integridad referencial: es utilizada para garantizar la validez de los datos de la BD.
- Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Además tiene habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
- Control de Concurrencia Multi-Versión: es usado para evitar bloqueos innecesarios, permite la lectura sin que sea bloqueada por los que escriben y actualizan registros.
- Incrementa la dependencia de la BD al registro de cambios antes de que estos sean escritos en la BD. Esto garantiza que en caso de que la BD se caiga, exista un registro de las transacciones a partir del cual se pueda restaurar la BD desde el punto en que se quedó. (Rojas Santiesteban & Boloy Boado, 2010)

Desventaja:

- Consume muchos recursos y carga con facilidad el sistema.

Luego de realizar un estudio de los diferentes SGBD, se evidencia que PostgreSQL es el adecuado para el trabajo con la BD para la Dirección de Agricultura del Consejo de la Administración Provincial Artemisa. Esta elección se basa en que SQL Server no es multiplataforma y al igual que Oracle, para su uso es necesario pagar importes por su licencia, lo cual no cumple con las políticas de la Universidad de migrar al software libre. (PostgreSQL, 2011)

1.8. Herramientas seleccionadas

A continuación una breve descripción de las herramientas que fueron seleccionadas para desarrollar la capa de acceso a datos del Módulo de Agricultura de la Dirección Provincial de Artemisa.

1.8.1 Power Architect 9.15

El SQL Power Architect herramienta de modelado de datos que fue creada por los diseñadores de almacenamiento de datos y tiene muchas características únicas dirigidas específicamente para el arquitecto de almacenamiento de datos. Permite a los usuarios de la herramienta ingeniería inversa de BD existentes, realizar perfiles de datos en bases de datos de origen y generar automáticamente los metadatos de ETL.

Algunas Características.

- Permite acceder a las bases de datos a través de JDBC(es una herramienta JAVA).
- Puedes conectarte a múltiples bases de datos al mismo tiempo.
- Compara modelos de datos y estructuras de bases de datos e identifica las discrepancias.
- Drag-and-drop de las tablas origen y las columnas en el área de trabajo.
- Ingeniería directa/inversa para PostgreSQL, Oracle, MS SQL Server y muchas más.
- Todos los proyectos se guardan en formato XML.

SQL Power Architec edición Community es gratuita bajo licencia Open Source GPL v.3. Es una herramienta ideal para grupos de desarrollo donde se pude realizar el modelado de datos y poder así tener documentado el modelo de datos de todas las aplicaciones que se desarrollan.

Está disponible para Windows, Linux y Mac OS X. (Community, 2010)

1.8.2. Hibernate 3.6

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos o anotaciones en los beans de las entidades que permiten establecer estas relaciones, esta última forma de notaciones es la que usaremos en nuestro proyecto. Hibernate se usa para manipular las consultas hacia la base de datos y para que un proyecto en Hibernate sea exitoso se tiene que tener en cuenta una buena Base de Datos, ya que las clases que se generan con Hibernate están muy ligadas con la base de datos. Y si hacemos un cambio en la base de datos también deberíamos de hacerlo en nuestro mapeo de las entidades que creamos. (Wong, 2011)

Características

- No intrusivo (estilo POJO). POJOs, que es cuando se crean las clases .java dentro del paquete de entidades y las clases .xml, después que se mapean los archivos de configuración y de ingeniería inversa junto con las tablas.
- Muy buena documentación (forums para ayuda, libro)
- Comunidad activa con muchos usuarios
- Transacciones, caché, asociaciones, polimorfismo, herencia, lazy loading, persistencia transitiva, estrategias de fetching.
- Potente lenguaje de consulta (HQL)
- Fácil testeo.
- No es estándar.

1.8.3. Spring

El Spring Framework (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. La primera versión fue escrita por Rod Jonhson, quien lo lanzó primero con la

publicación de su libro *Expert One-on-One Java EE Design and Development* (Wrox Press, octubre 2002). También hay una versión para la plataforma .NET, Spring .NET.

Es un framework muy usado con Hibernate para poder hacer la programación un poco más limpia y es el encargado de manejar las transacciones. (Wong, 2011) (Peñalver, y otros, 2010)

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al ser considerado como una alternativa y sustituto del modelo de Enterprise JavaBeans. (Wrox Press, 2005)

Funcionalidades de Spring con Hibernate. (Agüero, 2007)

1. Gestión y definición de recursos centralizada desde Spring.
2. Conversión de excepciones propietarias.
3. Acceso a métodos DAO desde HibernateTemplate.
4. Es la versión en Spring de la interface Session de Hibernate.
   - Garantiza que las instancias de Session sean abiertas, cerradas correctamente y participen automáticamente en transacciones.
   - Instancias de HibernateTemplate son thread-safey reusables.
   - Convierte las checked HibernateExceptions a unchecked DataAccessExceptions (excepciones traducidas por Spring)
   - Demarcado programático de transacciones.
5. Demarcado declarativo de transacciones.

1.8.4. NetBeans IDE

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está

escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios.

Se decide que el NetBeans en la herramienta a utilizar en la solución propuesta. Esta elección se basa en que es un entorno integrado para el desarrollo de aplicaciones web. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones. Proporciona extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación.

1.8.5. Otras herramientas

Visual Paradigm para UML (por sus siglas en inglés, Unified Modeling Language, Lenguaje Unificado de Modelado) es fácil de usar y soporta la última notación *UML* 2.1, ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XML (del inglés eXtensible Markup Language, lenguaje de marcas extensible), generador de informes, editor de figuras, integración con MS Visio, plug-in[9], integración IDE[10] con Visual Studio, Eclipse, NetBeans y otros. Entre sus nuevas características se incluyen el modelado colaborativo con *CVS* y Sub-versión, e interoperabilidad con modelos UML a través de XML.

Como aplicación gráfica para administrar el SGBD PostgreSQL se hace uso del PgAdmin III, la herramienta más popular de código abierto de administración y pla-taforma de desarrollo de PostgreSQL.

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar BD complejas. La interfaz gráfica 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 TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante *SSL* para mayor seguridad (PgAdmin PostgreSQL Tools, 2008).

1.9. Metodología SXP

La Metodología SXP, es un híbrido cubano de metodologías ágiles que tiene como base las metodologías SCRUM y XP, que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.(Peñalver, y otros, 2010)

Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Basada completamente en los valores y principios de las metodologías ágiles expuestos en el Manifiesto Ágil. Como método de estimación se utiliza la opinión de expertos y constan con métricas o indicadores para lograr una eficiente calidad. (Peñalver, y otros, 2010)

Consta de 4 fases principales:

- Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega, puesta en marcha.
- Mantenimiento, donde se realiza el soporte para el cliente.

De cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implementación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario, es decir, es un flujo que se puede mover y utilizarlo en cualquier parte del ciclo de vida del proyecto. (Peñalver, y otros, 2010)

Ventajas:

- El líder de proyecto puede llevar un mejor control de las tareas y la planificación de las mismas.
- Involucra a los miembros del equipo de desarrollo en las decisiones sobre el proyecto y sus vías de desarrollo.
- El trabajo es ágil y mantiene al cliente dentro del equipo de desarrollo.
- Integración continua del sistema.
- Reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación.
- Todos los integrantes del equipo de trabajo conocen algo sobre todas las partes y muy bien aquellas en las que trabajan.
- Todo el código se escribe en parejas.
- Se realiza el trabajo de 1 persona en casi la mitad del tiempo y mejor.

Conclusiones parciales

Luego de estudiar los diferentes SGBD que existen se concluye que PostgreSQL es el más indicado para la realización de la BD para la Dirección de Agricultura del Consejo de la Administración Provincial Artemisa.

Se selecciona la metodología SXP porque está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. También se utilizarán herramientas como el Power Architect, Hibernate y NetBeans.

CAPÍTULO 2. DISEÑO Y ARQUITECTURA DE LA BASE DE DATOS.

Introducción

En este capítulo se brinda información acerca de las técnicas de diseño de bases de datos, se analiza el modelo lógico y físico de la base de datos para la gestión de la información de la agricultura y para definir las características del sistema: se realiza el diseño con metáforas (diagrama de paquetes).

2.1. Metodología para el diseño de bases de datos

La metodología para el diseño de la BD, consta de los siguientes pasos:

- Determinación de entidades y atributos.
- Normalización de entidades.
- Determinación de relaciones.
- Obtención del modelo lógico global de los datos.
- Diseño físico de la BD.

Cuando se realiza el diseño de la BD para un sistema determinado, es necesario establecer los datos a tomar en cuenta y las dependencias funcionales existentes entre ellos.

Rigurosamente, esto se obtiene luego de realizada la etapa de análisis del sistema y partiendo de lo obtenido en esta. (Mato García, 1999)

2.2. Técnicas para diseñar bases de datos relacionales

Existen dos escuelas que implican dos grupos de algoritmos a la hora de aplicar la Teoría de Normalización: los métodos de Análisis o descomposición y los Procedimientos de síntesis, que a continuación se detallan.

Diseño descendente, Análisis o descomposición

Este método es el que se emplea mediante un proceso denominado “normalización”. El mismo parte de una relación llamada Universal que contiene todos los atributos del universo del discurso y las dependencias funcionales existentes entre

ellas. Por medio de descomposiciones sucesivas y cumpliendo determinados principios de conservación de la información y de las dependencias, resultan esquemas de menor grado en formas normales cada vez más avanzadas, por lo que se puede garantizar que se reducen las anomalías. Se pueden aplicar estos métodos también a cada una de las relaciones obtenidas a partir de un esquema conceptual en un modelo de datos de alto nivel (Modelo Entidad-Relación) y luego transformar el esquema conceptual a un conjunto de relaciones, empleando un procedimiento de transformación. También se puede aplicar el método a cada una de las relaciones que se obtienen de la transformación del Diagrama Entidad-Relación al modelo relacional. El objetivo inicial que se pretende con estos métodos es separar la información referente a un objeto o entidad diferente.

Síntesis relacional o Procedimientos de síntesis

Este es un método de diseño alternativo a la descomposición, resulta ser un enfoque más purista. Implica contemplar el diseño de esquemas de BD relacionales estrictamente en términos de dependencias funcionales, y otros tipos especificados para los atributos de la BD. Aquí los esquemas de relación en tercera forma normal o forma normal de Boyce-Codd se sintetizan gradualmente agrupando los atributos apropiados. A partir de conjuntos de atributos y dependencias funcionales se obtienen relaciones. Recorre un camino inverso a la descomposición, es decir, busca agrupar atributos, a fin de tener en una relación toda la información correspondiente a un objeto o entidad. (EVA, 2010)

2.2.1. Normalización

La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización. El concepto de normalización fue introducido por *E.D. Codd* y fue concebido para aplicarse a sistemas relacionales. Sin embargo, tiene aplicaciones más amplias. La normalización es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para

describir la estructura lógica de los datos en un sistema de información.

Ventajas:

- Evita anomalías en la actualización.
- Mejora la independencia de los datos, permite realizar extensiones de la BD, y afecta muy poco, o nada, a los programas de aplicación existentes que accedan a la BD.

La normalización involucra varias fases que se realizan en orden. La realización de la segunda fase supone que se ha concluido la primera y así sucesivamente. Tras completar cada fase se dice que la relación está en:

- Primera Forma Normal (1FN): una relación está en dicha forma normal si los valores de sus atributos son atómicos, los dominios no tienen elementos que a su vez sean conjuntos y la relación no incluye ningún grupo repetitivo.

- Segunda Forma Normal (2FN): debe estar en 1FN y además define que los atributos no llaves son funcionales y completamente dependientes de la llave primaria, por lo general compuesta.

- Tercera Forma Normal (3FN): debe estar en 2FN y señala que no pueden existir dependencias transitivas entre los atributos no llaves, los que son independientes de cualquier otro atributo no llave primaria.

- Forma Normal de *Boyce-Codd (FNBC)*: además de cumplir con lo de las tres formas normales anteriores, debe satisfacer que cada determinante es una llave (candidata o primaria).

- Cuarta Forma Normal (4FN): debe estar en *FNBC* y que además existan tres o más atributos formando parte de la llave primaria. Debe cumplir que no existan dos o más atributos independientes con dependencia respecto a un conjunto de atributos, formando parte de la llave junto a dicho conjunto, es decir, no dependencia entre atributos llaves.

- Quinta Forma Normal (5FN): cumple que toda dependencia de agregación (*join dependency*) es implicada por las llaves candidatas. Diseñada para reducir la redundancia en relaciones que almacenan hechos multievaluados a través del aislamiento de relaciones semánticamente relacionadas. Una relación está en 5FN cuando su contenido no puede ser reconstruido a partir de un conjunto de tablas, se excluye la posibilidad de tablas con los mismos atributos llaves.

- FN de dominio/llave (*DKNF* por sus siglas en inglés): no depende de las FN anteriores. Declara de manera explícita todas las reglas de negocios como reglas de la BD. Con ella se evitan las anomalías no temporales a la hora de insertar y eliminar datos o entidades completas con relaciones de dependencias. Define como "llave" al identificador único de una tupla y como "dominio" a los valores de un atributo, que son permitidos (tanto física como semánticamente). Dentro de sus características fundamentales se define el proceso más simple: no se necesita saber que se elimina una dependencia transitiva. Sólo se deben aplicar las restricciones que surgen como consecuencia lógica de las definiciones de llaves y del dominio.

Para el proceso de la normalización, en la presente investigación, se comprueba que cada entidad cumpla con un conjunto de reglas basadas en la clave primaria, las dependencias funcionales parciales con respecto a la llave primaria y las dependencias transitivas de sus atributos. Luego de realizar el proceso de normalización se evidencia que el modelo de datos cumple con la normalización de la 3FN.

2.3. Tipos de datos

Tipo de datos "**serial**"

El tipo de datos "serial" es un tipo numérico que se incrementa automáticamente admitiendo el dinamismo para la generación de las claves primarias de las entidades, proporcionando números enteros consecutivos a cada tupla que se registra. En el caso de las claves primarias de las empresas, su uso obstruye el curso nor

mal de réplica de datos. Cuando se refiere a la replicación de información surge la tendencia a la duplicación de datos ya existentes, o a la no inclusión de los mismos por existir su llave en el nodo destino. A causa de ello se propone el uso de un mecanismo interno para la generación de las claves, que está basado en una "variante" del md5. Es decir, dado que el md5 es un algoritmo usado para la generación de un texto encriptado, se usa para generar las claves a partir de la información recogida en uno de los campos de la tabla en cuestión, o puede ser a partir de la combinación lineal de varios campos.

Tipo de datos **"integer"**

Se utiliza para los campos con valores enteros positivos.

Tipo de datos **"timestamp"**

Se utiliza para los campos donde es guardada la información referente a fechas y tiempos.

Tipo de datos **"date"**

Se utiliza para los campos donde es guardada la información referente a fechas.

Tipo de datos **"varchar(n)"**

Los tipos de datos varchar almacenan datos compuestos de caracteres en mayúsculas y minúsculas, como por ejemplo: a, b y C; números como 1, 2 y 3; y caracteres especiales como el signo de arroba (@), "y" comercial (&) y el signo de exclamación de cierre (!). Además este tipo de datos es de longitud variable, n puede ser un valor entre 1 y 8000. En el trabajo con la BD se tiene en cuenta lo siguiente:

- varchar (10) representa campos cortos.
- varchar (20) representa campos medianos.
- varchar (35) representa campos largos.

CAPÍTULO 2. DISEÑO Y ARQUITECTURA DE LA BASE DE DATOS.

2.4. Requisitos del sistema propuesto.

La captura de requisitos cumple un papel importante en la realización de lo que se-desea producir, pues describe el comportamiento del sistema a desarrollar.

2.4.1. Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen (Conferencia de Ingeniería de Software, 2011). Los requisitos funcionales del sistema, propuestos por los desarrolladores del lado del cliente y del servidor, fueron llevados a requisitos funcionales de la BD. A continuación aparecen los requisitos funcionales de la BD.

Prioridad muy alta

- Insertar información sobre la siembra de maíz
- Insertar datos sobre la Alimentación de Artemisa.
- Insertar datos sobre la siembra de cultivos varios.
- Insertar datos sobre la siembra para la Ganadería.
- Insertar datos sobre la Papa.
- Insertar datos sobre la siembra de frijol.

Prioridad alta

- Insertar datos sobre el Balance de Existencia Inicial y Final de los productos en nave.
- Insertar datos sobre la leche.

Prioridad media

- Insertar datos sobre el plan de los Mataderos.
- Insertar datos sobre el cumplimiento de la Demanda de la Provincia.
- Insertar datos sobre las Industrias de la Provincia.
- Insertar datos sobre la preparación de tierras en temporadas.

2.4.2. Requisitos no funcionales.

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

Se describen a continuación los requerimientos fundamentales relacionados con la BD definidos por el equipo de trabajo del módulo de Agricultura.

Seguridad

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

Usabilidad

- La información deberá estar disponible en todo momento, limitada solamente por las restricciones que esta tenga de acuerdo a las políticas de seguridad del sistema.

Soporte

- La aplicación recibirá mantenimiento según el tiempo determinado por el equipo de desarrollo.

Software

- El Sistema Gestor de Base de Datos a utilizar debe ser: PostgreSQL 9.1.
- Se debe trabajar sobre el Sistema Operativo Linux para lograr la compatibilidad con las herramientas de desarrollo.
- Navegador compatible o superior con Chromium Chrome 4, Opera 11, Firefox 4, o Safari 5.

Hardware

Servidor de Base de datos:

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

2.5. Diseño con metáforas

Debido a que SXP está basada en XP, y dicha metodología define un término llamado metáfora, lo cual según Martin Fowler es una historia compartida que describe cómo debería funcionar el sistema y define que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema.

El Diseño con metáforas es sencillamente el diseño de la solución más simple que pueda funcionar y ser implementado en un momento dado del proyecto; lo cual genera el artefacto conocido como Modelo de Diseño, que a su vez está compuesto por un diagrama de paquetes, el cual expone dicho diseño.

A continuación se representa el diagrama de paquetes para el sistema que se propone. (Ortíz Valmaseda, y otros, 2009)

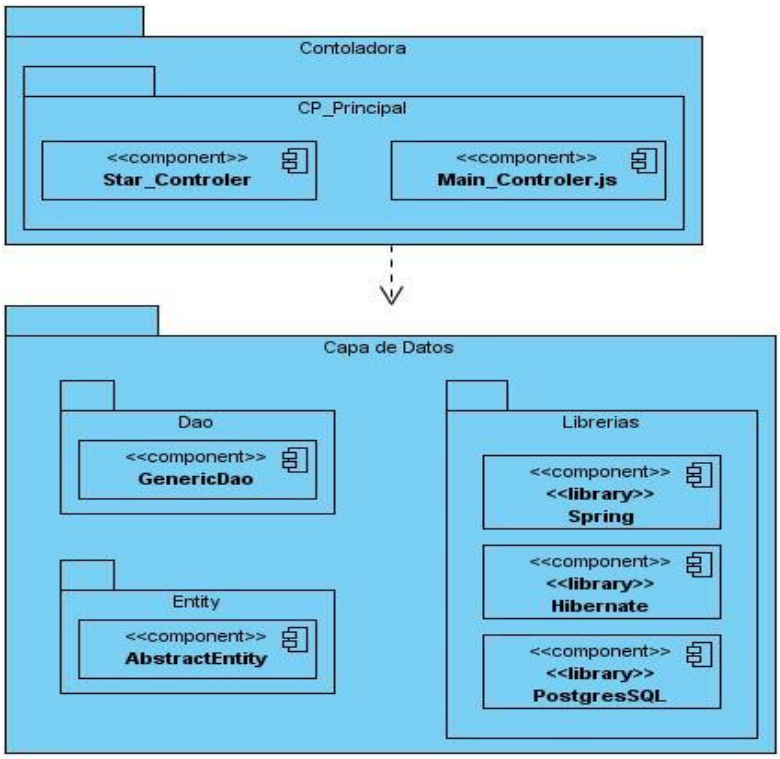

Figura 5. Diagrama de paquetes.

Descripción:

La capa Controladora integra todas las clases y funcionalidades relacionadas con el negocio y se relaciona con la capa de Acceso a Datos, posibilitando el acceso del cliente a los datos almacenados en la BD.

2.6. Diagrama de clases

La modelización semántica de los datos consiste en estudiar los datos que se pretenden almacenar en la BD antes de elegir el modelo lógico de datos concreto que se va a usar. A continuación, se representa una parte del diagrama semántico de los datos más conocido como diagrama de clases.

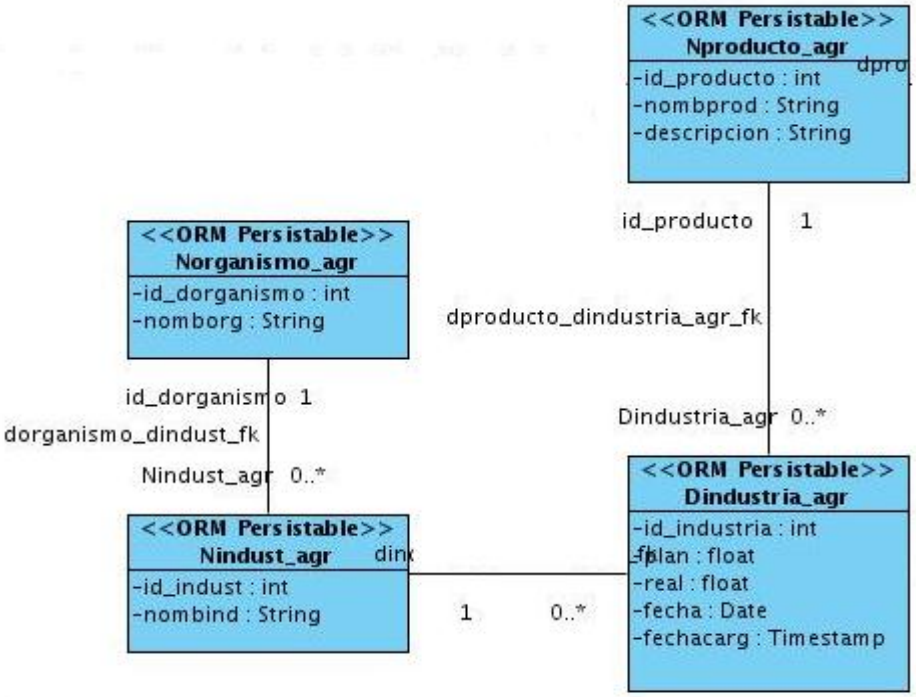

Figura 6. Diagrama de clases

2.7. Modelo Entidad Relación

Generalmente todo modelo tiene una representación gráfica, para el caso de datos el modelo más popular es el modelo entidad-relación o diagrama E/R.

Se denomina así debido a que precisamente permite representar relaciones entre entidades (objetivo del modelado de datos).

El modelo debe estar compuesto por:

- Entidades
- Atributos
- Relaciones
- Cardinalidad
- Llaves

CAPÍTULO 2. DISEÑO Y ARQUITECTURA DE LA BASE DE DATOS.

El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad, mediante un conjunto de representaciones gráficas y lingüísticas.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido.

A continuación, se representa una parte del diagrama del Modelo Entidad Relación.

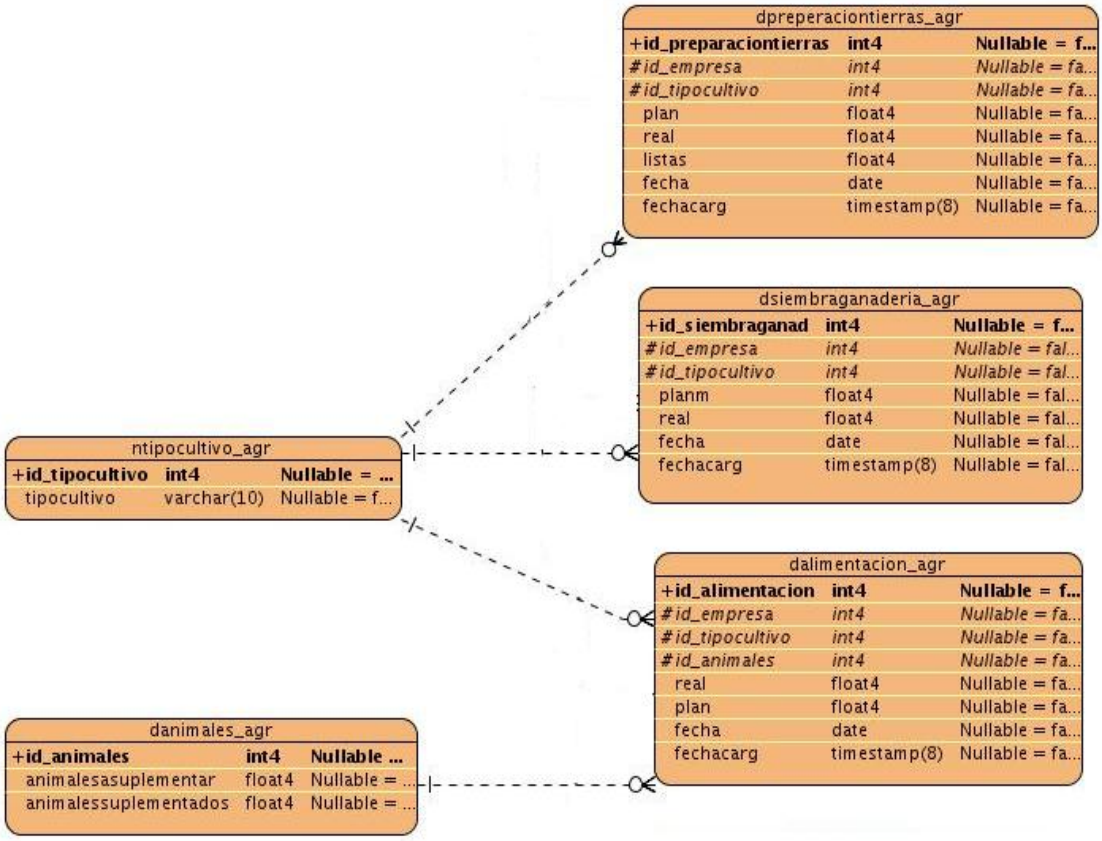

Figura 7. Modelo Entidad Relación

2.8. Modelo físico

El paso de convertir el modelo lógico a tablas hace que las entidades pasen a ser

tablas (más las derivadas de las relaciones) y los atributos se convierten en las columnas de dichas tablas.

Físicamente esta metáfora de una tabla se mapea al medio físico. A continuación se muestra una de las tablas principales.

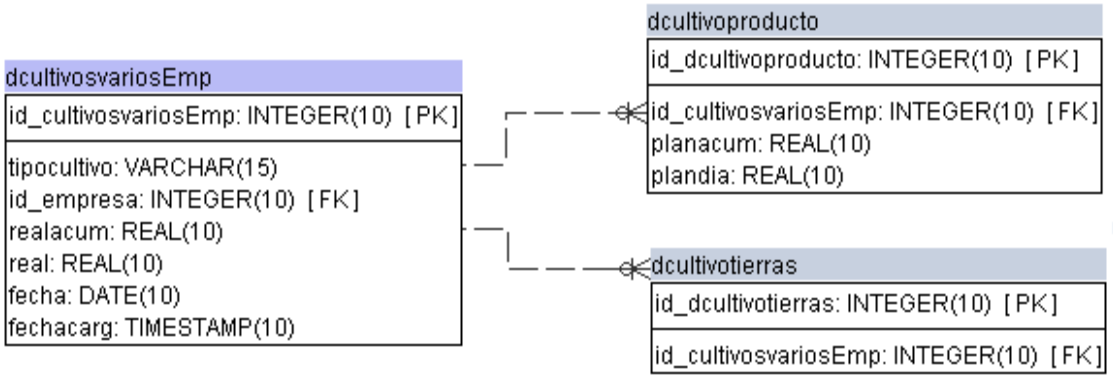

Figura 8. Modelo Físico

2.9. Descripción de las entidades

Una entidad no es más que un objeto al cual le asociamos atributos, es una forma física de visualizar la información. A continuación se muestra una de las entidades principales del módulo de Agricultura junto con su descripción, las restantes descripciones se encuentran ubicadas en el Modelo Canónico de Datos.

Entidad:	dcultivosvariosEmp_agr		
Descripción:	Esta entidad recogerá los datos de los cultivos varios.		
Relaciones:	dcultivoproducto_agr, dcultivotierra_agr, nempresas_agr		
Campos	**Tipo de Dato**	**Tamaño**	**Descripción**
id_cultivosvariosEmp	INTEGER	10	Este campo presenta el id del cultivo

id_empresa	INTEGER	10	Este campo presenta el id de la provincia
realacum	INTEGER	10	Este campo presenta el real acumulado
real	INTEGER	10	Este campo presenta la cantidad real de cultivos
fecha	DATE	10	Este campo presenta la fecha
fechacarga	TMESTAMP	10	Este campo presenta la fecha de carga

Conclusiones parciales

El modelado es la actividad más delicada e importante en la realización de una aplicación con base de datos. Al igual que en el desarrollo de un sistema, toda modificación al esquema de base de datos debe realizarse primero en el modelo conceptual, no en el lógico ni en el físico. La habilidad de crear buenos modelos es una cualidad que se adquiere con la experiencia.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DE LA CAPA DE ACCESO A DATOS.

Introducción

En este capítulo se describe la implementación del diseño de la capa de acceso a datos y cómo se realiza la integración con la aplicación en la capa de persistencia. Se valida el diseño realizado de forma teórica teniendo en cuenta aspectos como la integridad, la normalización, la seguridad y cómo responde el sistema a las pruebas.

3.1. Persistencia de Objetos

La tarea de persistir objetos Java en una base de datos relacional actualmente está siendo facilitada por un gran número de herramientas que permiten a los desarrolladores dirigir motores de persistencia para convertir objetos Java a columnas/registros de una base de datos y viceversa. Esta tarea implica serializar objetos Java estructurados en forma de árbol a una base de ratos relacional estructurada de forma tabular y viceversa. Esencial para este esfuerzo es la necesidad de mapear los objetos Java a columnas y registros de la base de datos de una manera optimizada en velocidad y eficiencia.

El marco de trabajo Hibernate se enfrenta al problema "objeto-java-a-base-de-datos" de forma tan elegante como cualquier otro marco de trabajo disponible. Hibernate funciona persistiendo y restaurando viejos objetos Java (POJOs) utilizando un modelo de programación muy transparente y poco exigente.

Hibernate es un marco de trabajo Java que proporciona mecanismos de mapeo objeto/relacional para definir cómo se almacenan, eliminan, actualizan y recuperan los objetos Java. Además, Hibernate ofrece servicios de consulta y recuperación que pueden optimizar los esfuerzos de desarrollo dentro de entornos SQL y JDBC.

3.2. Configuración de Hibernate

A continuación una breve explicación de cómo se mapea con Hibernate:

1. Primeramente, se crea un proyecto, y se le adicionan todas las librerías de Hibernate, de Spring Framework 3.0.2 RELEASE y la de Postgres SQL JDBC.

2. Luego, se hace la Configuración Wisard de Hibernate.

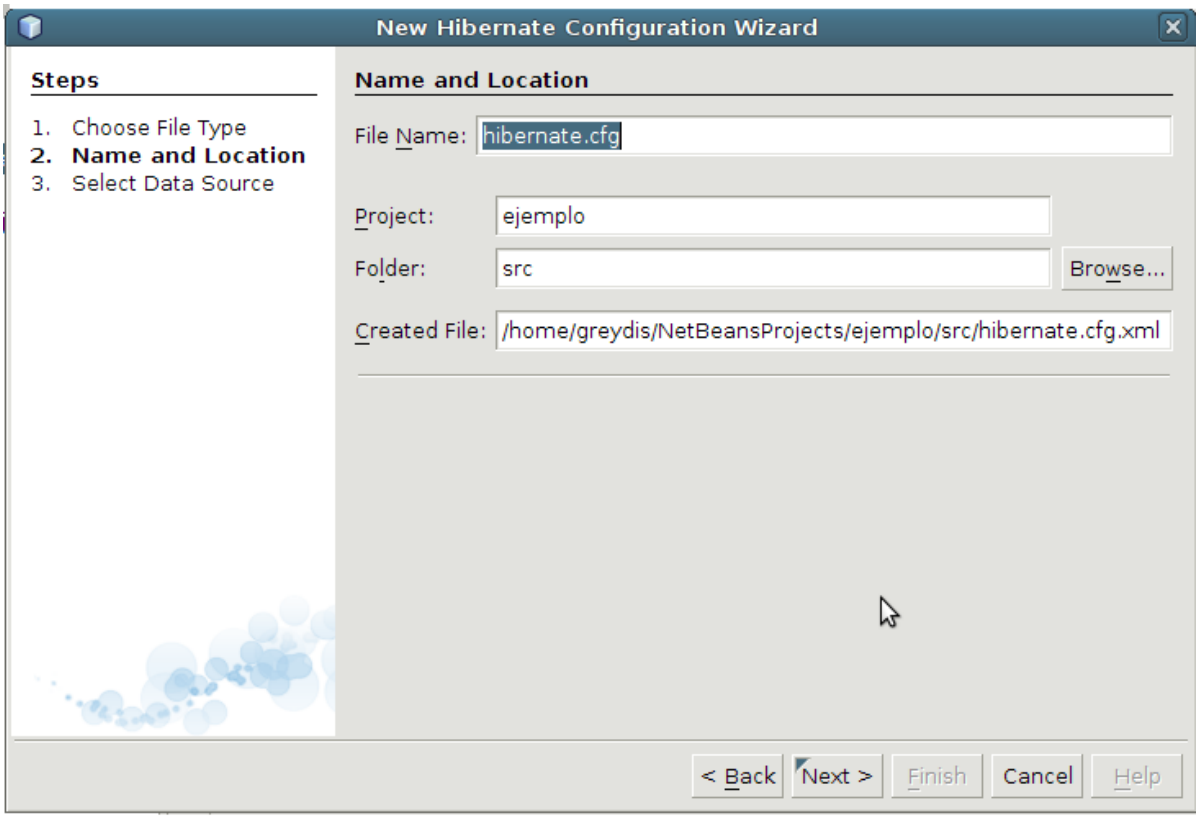

Figura 9. Configuración Wisard de Hibernate.

Seguido de esto se crea una conexión a la base de datos y luego se presiona el botón Finish.

De dicha configuración se crea el fichero hibernate.cfg.xml, el cual define la información sobre la conexión a la base de datos, la clase factoría de transacciones, los recursos de mapeo, etc.

3. Luego se hace la ingeniería inversa de la Configuración Wisard:

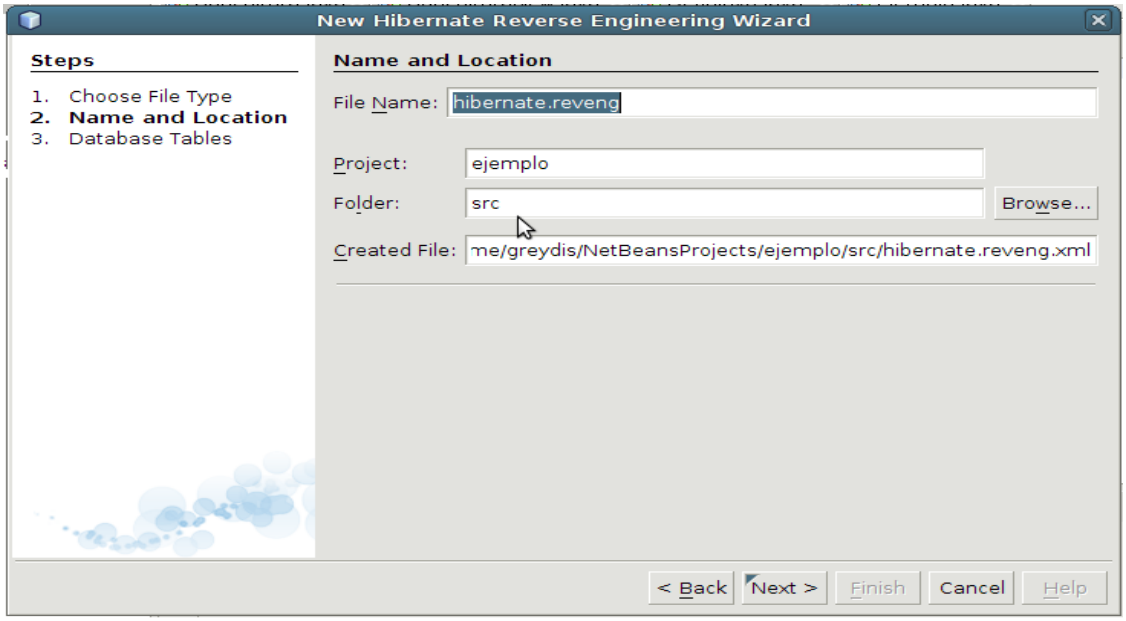

Figura 10. Ingeniería inversa de la Configuración Wisard.

Se presiona el botón Next y aparece una ventana, en la cual se muestran las tablas de la BD, como se muestra en la Figura 8.

Se seleccionan las tablas que se van a mapear. Y lo que debemos es adicionarlas a la columna derecha y así ya estarán listas para el mapeo.

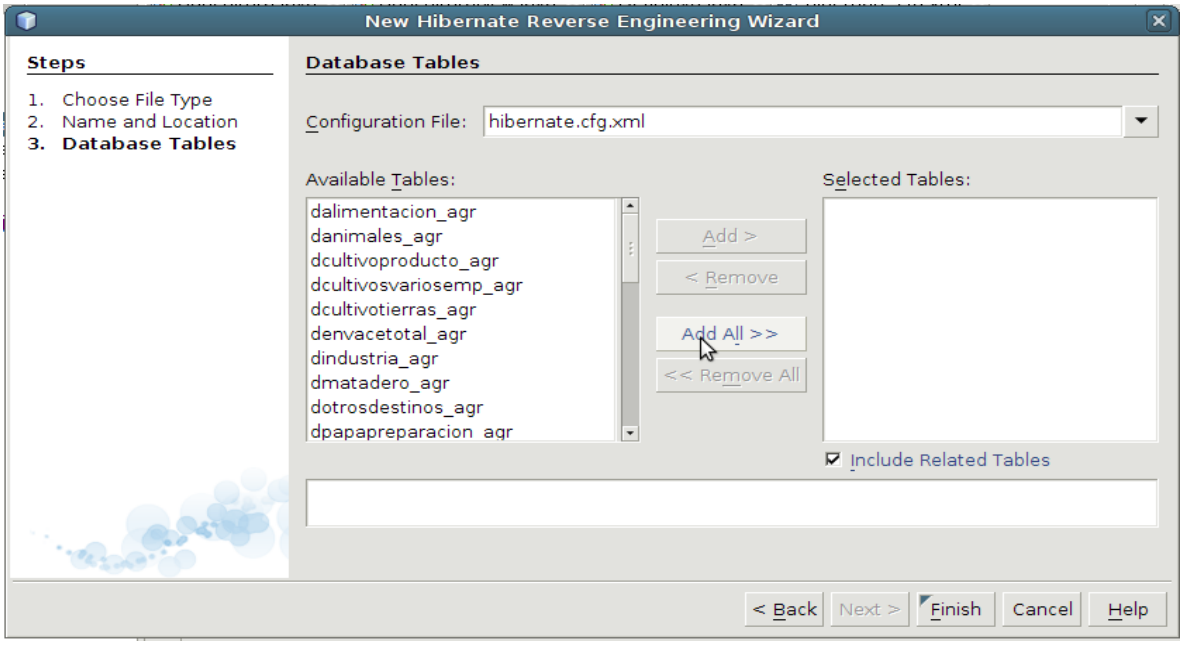

Figura 11. Selección de las tablas que se van a mapear.

Se crea la conexión con la base de datos.

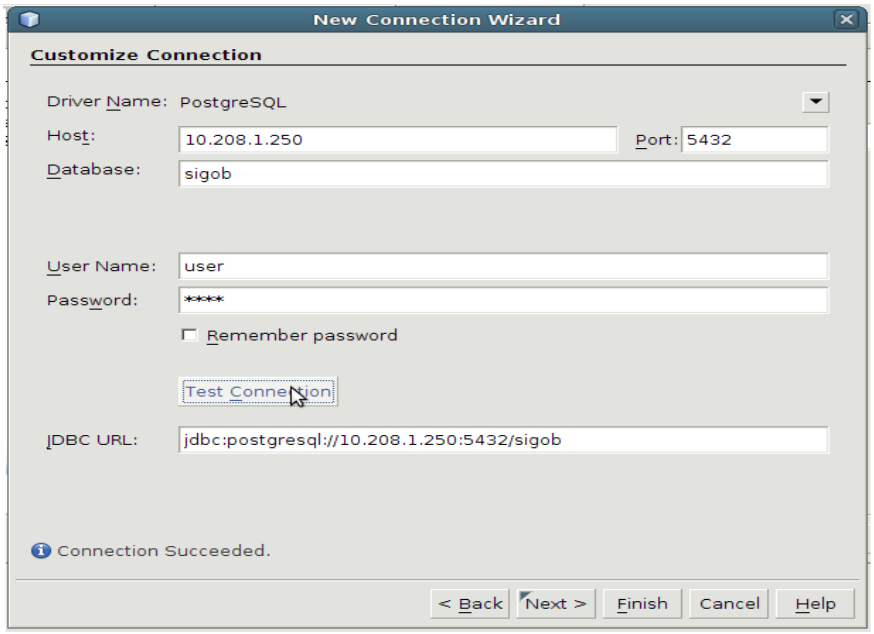

Figura 12. Creando la conexión con la base de datos.

Y se selecciona el esquema donde se encuentran las tablas, para la conexión.

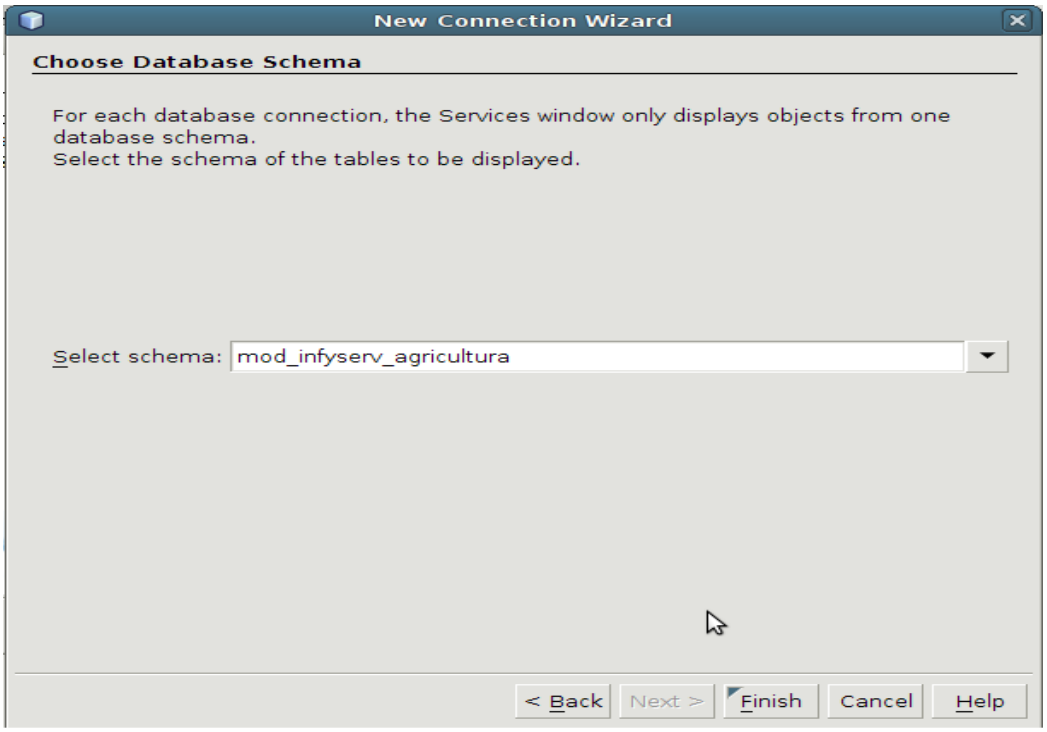

Figura 13. Seleccionando el esquema.

Se mapean las tablas y como resultado se crea el fichero hibernate.reveng.xml.

4. Para finalizar se mapean los archivos y los POJOs de la base de datos.

En este caso las clases se mapean con Anotaciones, por las ventajas que brindan las mismas, en especial ya que permiten al programador declarar en su código fuente cómo debe comportarse el software.

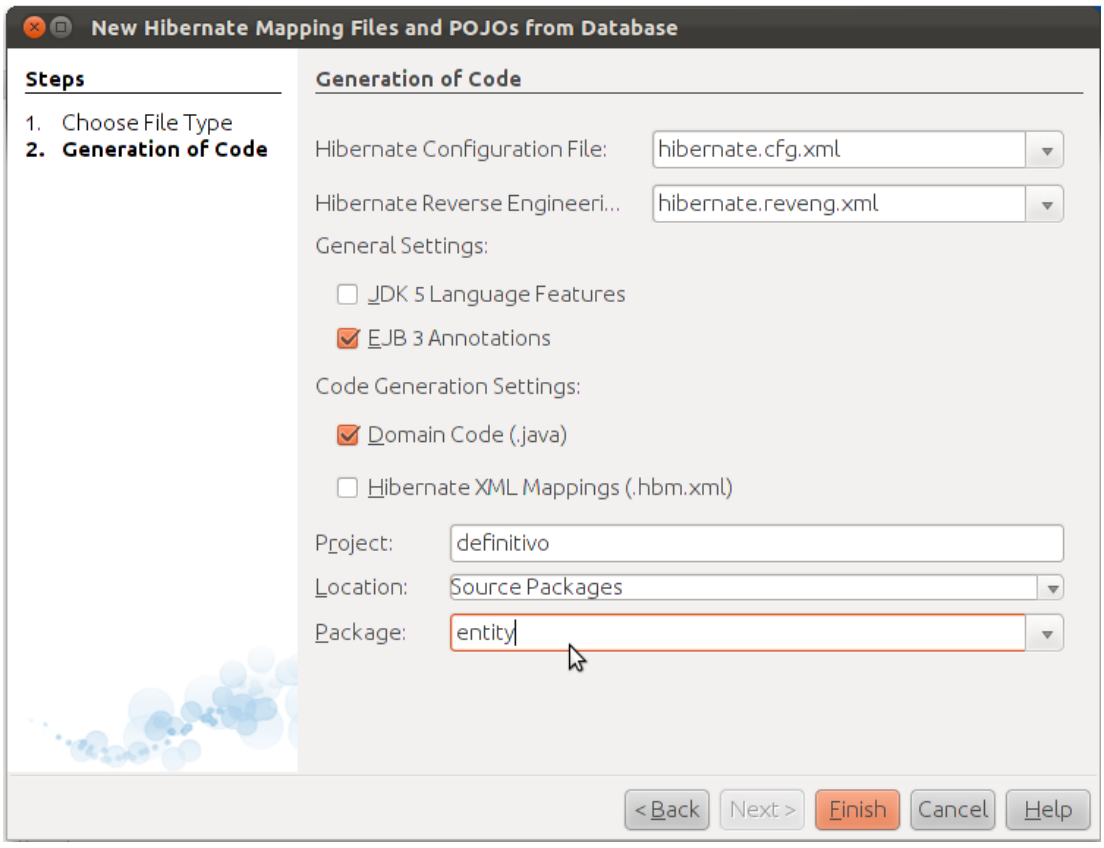

Figura 14. Hibernate Mapping Files and POJOs from Database.

Al dar clic en Finish ya todas las clases se encontraran en nuestro proyecto en el paquete entity, listas para trabajar con ellas.

3.3. Estructura del proyecto

El proyecto está compuesto por 4 paquetes, el default package, el paquete entity, paquete dao y por último el paquete AgriculturaUltimo.

3.3.1. Paquete default package

El paquete default package contiene los ficheros de la configuración del Hibernate Configuration Wizard y del Hibernate Reverse Engineering Wizard, explicados anteriormente.

3.3.2. Paquete entity

En este paquete se encuentran todas las clases mapeadas, en el mismo es donde se realiza el Hibernate Mapping Files and POJOs from Database. Dicho paquete presenta, además de las entidades mapeadas, una entidad con el nombre de AbstractEntity, la cual tiene la función de proporcionarle el atributo del id autoincrementable a cada una de las demás clases, ya que este es un atributo que tienen todas en común, por tanto, todas las demás clases heredan de ella. La clase AbstractEntity tiene la siguiente estructura:

```
package entity;
import java.io.Serializable;

public abstract class AbstractEntity implements Serializable {
	private static final long serialVersionUID = 5143168877262662951L;
	protected Long id;

// Este metodo es abstracto porque las anotaciones no son heredables
	public abstract Long getId();

/* Este método es protegido para evitar que un programador pueda poner un identificador en la instancia, ya que los identificadores deben ser gestionados por la capa de persistencia */
	protected void setId(final Long id){
		this.id=id;
	}
}
```

Como todas las clases tienen que heredar de la clase AbstractEntity, la estructura

de las restantes clases cambia. Uno de los cambios es que cada una de ellas va a heredar de la AbstractEntity, se les debe eliminar el atributo del id ya que este lo brindará la clase padre, luego se modifican los constructores, en este caso tenemos tres, uno vacío, otro con todos los atributos menos el id, y el último con todos donde el id será de tipo de dato Long, pues así es como está definido en la clase padre.

Después de esto, definimos tres Anotaciones para las Propertys del atributo id, donde se definirá que este atributo será autoincrementable, tomando para esto el valor de la secuencia de este atributo en la BD, y por último se sobrescribe el método abstracto en la clase padre. A continuación se muestra como queda después de estos cambios la clase Danimales:

```
@Entity
@Table(name="danimales_agr" ,schema="mod_infyserv_agricultura")

public class DanimalesAgr extends AbstractEntity  implements java.io.Serializable {

    private float animalesasuplementar;
    private float animalessuplementados;


  public DanimalesAgr() {
   }
  public DanimalesAgr( float animalesasuplementar, float animalessuplementados) {
     this.animalesasuplementar = animalesasuplementar;
     this.animalessuplementados = animalessuplementados;
  }
public DanimalesAgr(long id, float animalesasuplementar, float animalessuplementados) {
      this.id = id;
      this.animalesasuplementar = animalesasuplementar;
```

```
  this.animalessuplementados = animalessuplementados;
    }
@Id
@SequenceGenerator(name="id_animales",sequenceName="mod_infyserv_agricultura.danimales_agr_id_animales_seq")
@GeneratedValue(generator="id_animales")
@Column(name="id_animales", unique=true, nullable=false)
    @Override
    public Long getId() {
        return this.id;
    }
@Column(name="animalesasuplementar", nullable=false, precision=8, scale=8)
    public float getAnimalesasuplementar() {
        return this.animalesasuplementar;
    }
 public void setAnimalesasuplementar(float animalesasuplementar) {
        this.animalesasuplementar = animalesasuplementar;
    }
@Column(name="animalessuplementados", nullable=false, precision=8, scale=8)
    public float getAnimalessuplementados() {
        return this.animalessuplementados;
    }
 public void setAnimalessuplementados(float animalessuplementados) {
        this.animalessuplementados = animalessuplementados;
    }
}
```

3.3.3. Paquete dao

El paquete dao (Objeto de Acceso a Datos), es el que suministra la interfaz común que existe entre la aplicación y la BD. Dicho dao contiene la clase genérica GenericDao, la cual tiene la tarea de gestionar todas las entidades, y de esta forma tener la independencia de la capa de datos. La clase GenericDao hereda de la clase HibernateDaoSupport implementada en el framework Spring. El HibernateDaoSupport, brinda la ventaja de trabajar con las sesiones de Hibernate, de forma automática y sin ningún tipo de gestión por parte del administrador de BD. En la clase GenericDao se crearon las funciones necesarias para trabajar de forma general con todas las entidades, ahora si hiciera falta alguna funcionalidad para una entidad específica se implementa una nueva clase dao que herede del dao genérico.

A continuación se explican las funciones que tiene la clase GenericDao:

saveOrUpdate: Este método recibe como parámetro una clase abstracta, verifica que el objeto este almacenado en la BD, si ésta lo actualiza sino lo salva.

save: Este método recibe como parámetro una clase abstracta y salva este objeto en la BD.

update: Este método recibe como parámetro una clase abstracta, verifica que el objeto este almacenado en la BD, si ésta lo actualiza.

find: Retorna una lista de todas las tuplas de la BD mapeadas como objetos, se le pasa como parámetro el tipo de dato que se desea buscar.

findByPK: Busca y retorna un objeto a través de la llave primaria y el tipo.

delete: Elimina un objeto pasado por parámetro.

deleteAll: Elimina todas las tuplas de una tabla recibiendo como parámetro el tipo de esta.

3.3.4. Paquete AgriculturaUltimo

Por último se encuentra el paquete nombrado igual que la clase principal del proyecto, en este caso se llama AgriculturaUltimo, en dicha clase se probarán cada una de las funcionalidades anteriormente expuestas.

3.4. Configuración del archivo hibernate-persistence

Este archivo es de gran importancia pues en él se configuran las librerías de Hibernate y Spring para la configuración de la BD para conectarse a la aplicación web, es decir, este archivo es esencial para el manejo de todos los procesos del sistema.

3.5. Integridad de los datos

La integridad de los datos se define como un conjunto de reglas, que utilizan la mayoría de las BD relacionales para asegurarse que los registros de tablas relacionadas son válidos, y que no se borren o cambien datos relacionados de forma accidental, produciendo errores de integridad (Carralero Alcalde, 2010). Algunas de las restricciones presentes en el modelo propuesto se basan en la integridad referencial, integridad de entidad e integridad de dominio. Una restricción es una regla que limita los valores que pueden estar presentes en la BD.

Se realizan pruebas encaminadas a demostrar que un buen diseño evita malas prácticas y errores en los datos. Las pruebas de integridad verifican que los datos sean exactos, completos, coherentes y autorizados, indican si hay errores en los controles de entrada o procesamiento (Rojas Santiesteban & Boloy Boado, 2010).

3.5.1. Integridad de entidades

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

restricciones PRIMARY KEY (llave primaria).

En la validación de integridad de las entidades almacenadas, se comprueba que todas posean una clave principal, que define a cada registro de modo exclusivo respecto al resto de los mismos, es por ello que ninguna de las llaves primarias puede tomar valores repetidos o valores nulos, como se observa en las Figuras 15 y 16 respectivamente.

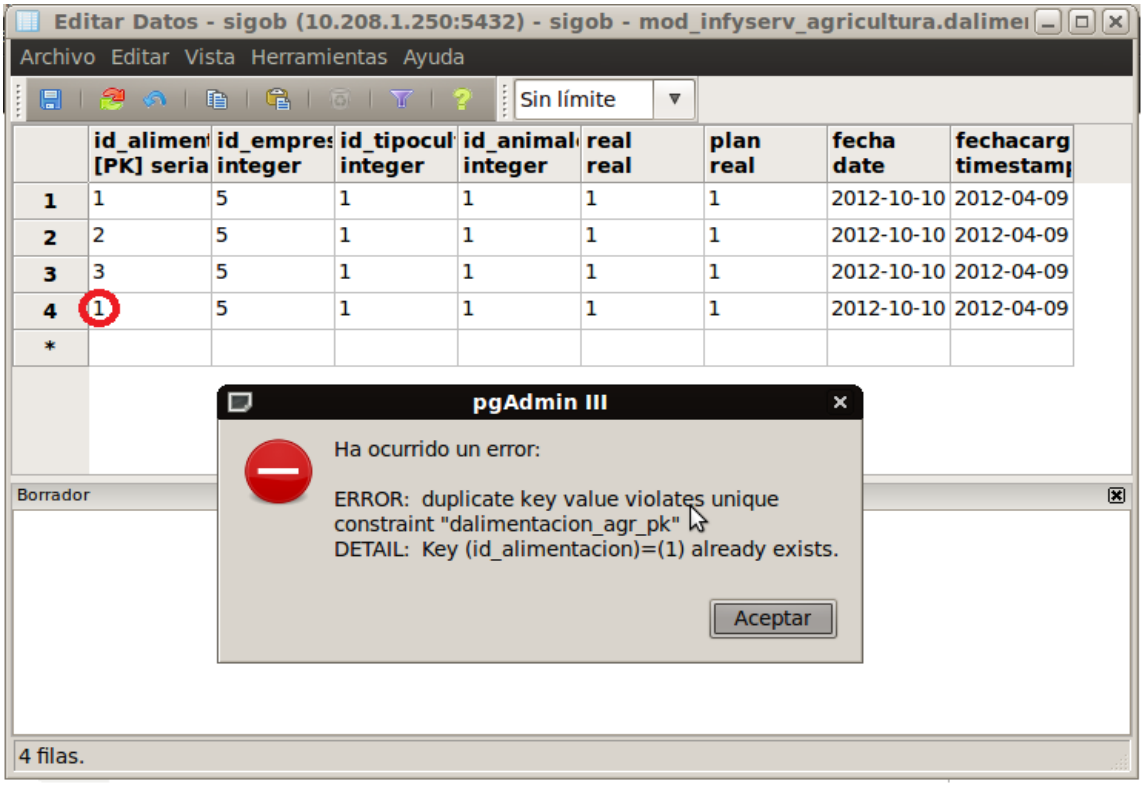

Figura 15. Ejemplo del error que emite el gestor al insertar una tupla duplicada.

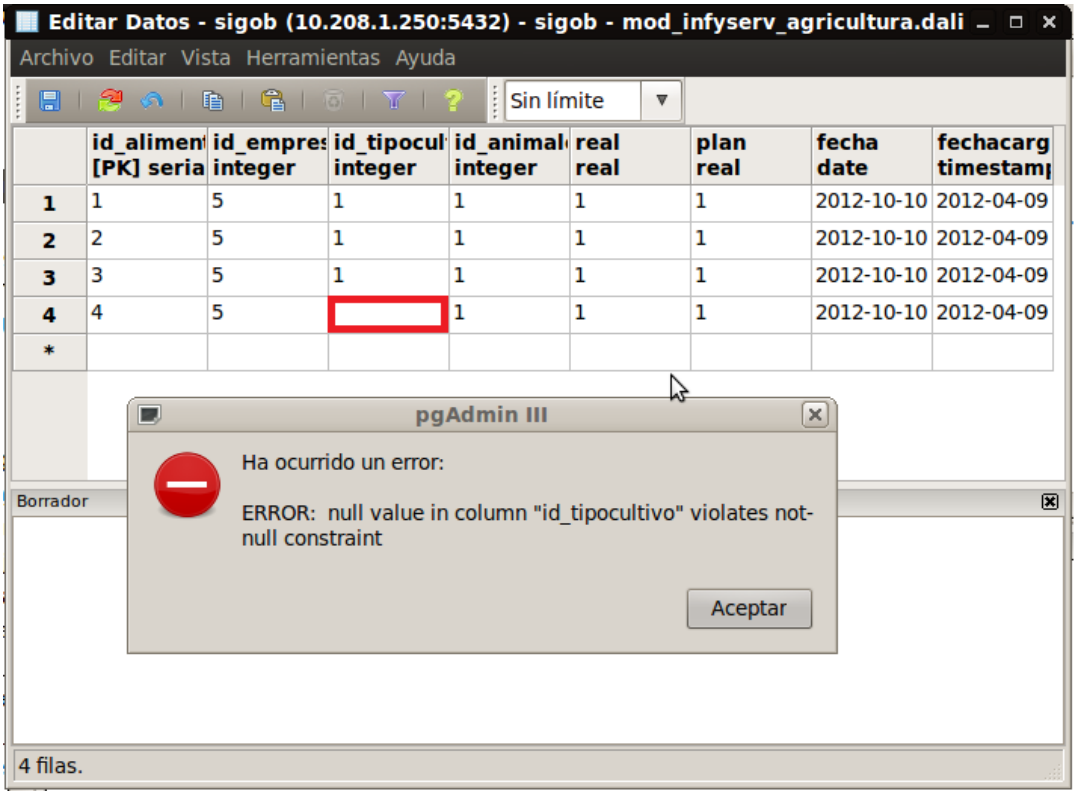

Figura 16. Ejemplo del error que emite el gestor cuando se insertan valores nulos no permitidos.

3.5.2. Integridad referencial

Contempla la integridad de las relaciones entre las entidades de la BD, específicamente cuando se crean, modifican o destruyen filas completas. Esta es la categoría más importante, dado que determina la consistencia de los datos existentes en cada tabla. Se logra a través de las restricciones de *FOREIGN KEY* y los modos de propagación de cambios.

Con las validaciones de integridad referencial se trata de asegurar que las filas relacionadas entre tablas no dejen de estarlo, o varíen esta relación cuando se modifiquen los datos. Con esta integridad se limita la actividad que puede realizar un usuario sobre la BD (Rojas Santiesteban & Boloy Boado, 2010).

Se toman algunas entidades para demostrar que no se permiten errores de inserción, por lo que se lanza un mensaje de error, tal como se muestra en la Figura 17.

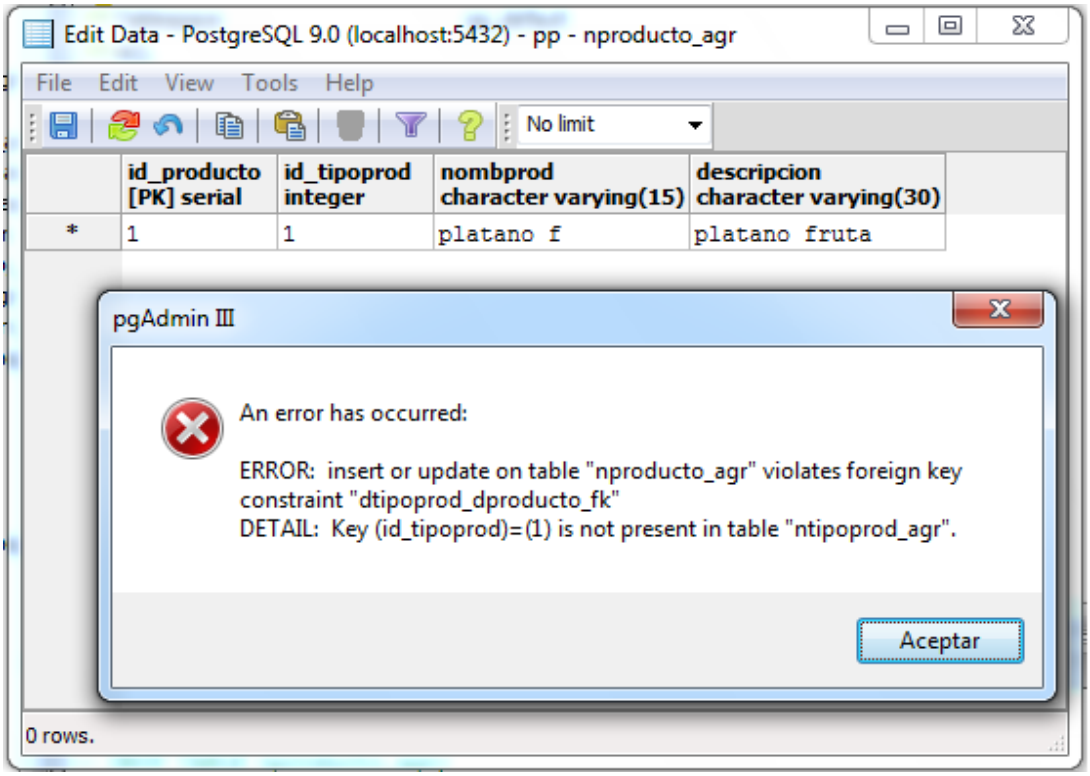

Figura 17. Ejemplo del error que emite el gestor cuando se viola la integridad referencial.

3.5.3. Integridad de dominio

La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Las pruebas de integridad de dominio sirven para controlar la información que se guarda en la BD, verificando que los datos son conformes con sus definiciones. Aseguran además que los datos tienen un valor legítimo. Las validaciones son a nivel de campo, es por ello que se toman varios atributos, todos referentes a distintos dominios de datos, y se intenta introducir valores de tipos diferentes a los definidos para obtener como resultado el mostrado en la Figura 18.

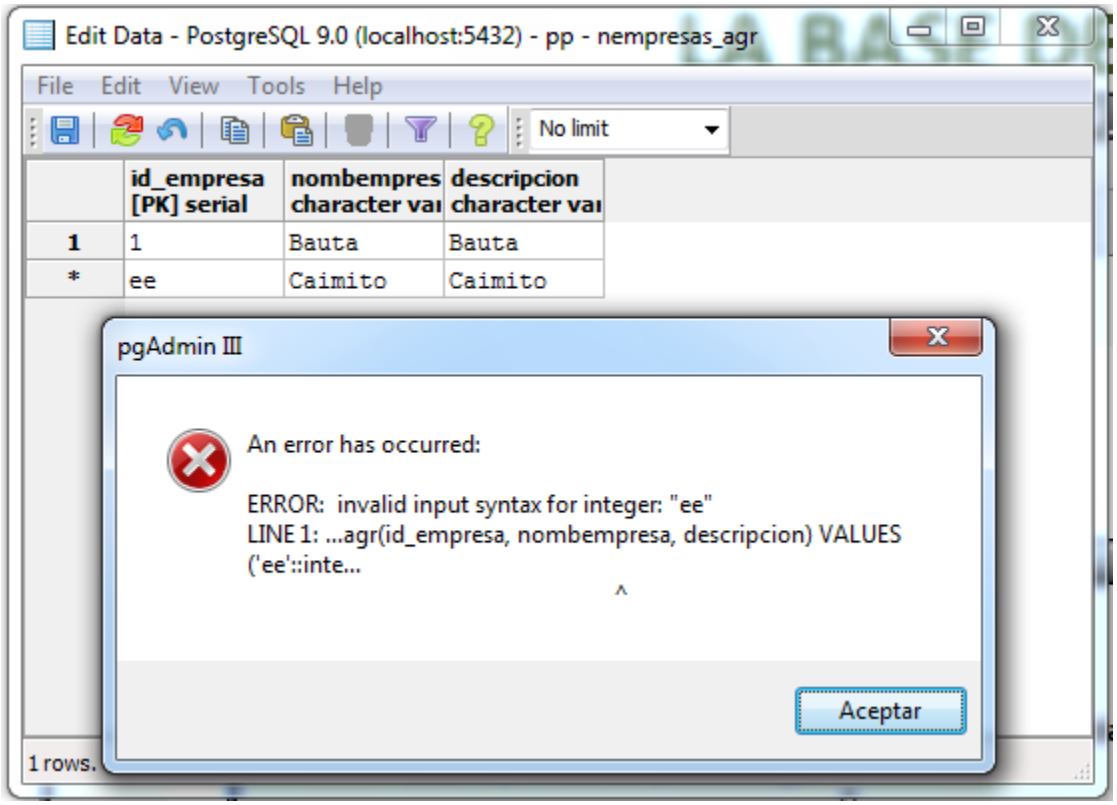

*Figura 18**.** Ejemplo del error que emite el gestor cuando se viola el dominio de un atributo.*

3.6. Calidad del dato

La calidad del dato es el grado en que los datos corresponden con la verdadera información. Las dimensiones de la calidad del dato vienen dadas por: **exactitud**, mide el grado en que la información refleja lo que está pasando con el evento, es decir, el dato es correcto para lo que está representando; **integralidad**, es el grado en que las BD cuentan con toda la información crítica requerida para el evento; **oportunidad**, que esté disponible cuando se requiere para tomar una decisión, o sea que el dato esté actualizado y la fecha de actualización es apropiada para la tarea a realizar; **consistencia**, el dato pasa todos los controles de aceptabilidad del formato, la longitud, entre otras características.

Después de analizar dichos factores, aterrizados al resultado de la investigación, se puede decir que se cumple con la calidad del dato. Para la realización del diseño y la implementación de la capa de acceso a datos del módulo de Agricultura del Consejo de la Administración provincial de Artemisa se hacen reuniones y entrevistas con los analistas del sistema, donde se tienen en cuenta los requisitos funcionales y las reglas del negocio, quedando bien identificados las tablas, relaciones y campos necesarios para el almacenamiento de toda la información con la precisión requerida. Actualmente, la aplicación en desarrollo trabaja con la capa de acceso a datos, lo que evidencia que la misma refleja los rasgos necesarios que aseguran la calidad del dato.

3.7. Valoración de resultados

Luego de concluir el diseño y la implementación de la capa de acceso a datos para la dirección de Agricultura del Consejo de la Administración provincial de Artemisa se obtuvieron los resultados siguientes:

- La capa de acceso a datos cumple con los requisitos funcionales.
- Las relaciones entre las tablas fueron construidas adecuadamente.
- Correcta normalización de la base de datos.
- La redundancia en la información se ha reducido en gran medida.

Conclusiones parciales

En el presente capítulo se explica brevemente cómo funcionan los métodos que se implementan en la capa de acceso a datos para tener acceso a los datos. Se realizaron pruebas de integridad al modelo de datos. Así se demostró la existencia de integridad referencial, de dominio y de entidad, lo cual evita la inconsistencia de los datos. También se demostró a través de las pruebas funcionales que se cuenta con una capa de acceso robusta y segura para brindar los servicios requeridos.

CONCLUCIONES GENERALES

Concluido el presente trabajo de investigación se puede afirmar que se ha cumplido con el objetivo del mismo a través del diseño e implementación de una capa de acceso a datos que permitió centralizar la información en la Dirección de Agricultura de la Administración Provincial de Artemisa.

- Durante la investigación se realizó un estudio acerca de los elementos teóricos principales relacionados con el diseño de base de datos que favoreció el desarrollo del trabajo permitiendo sentar las bases para un correcto diseño de la capa de acceso.

- Con el análisis bibliográfico y la caracterización del estado actual de los SGBD, se logró seleccionar el SGBD PostgreSQL que se utilizó para la manipulación de los datos relativos a los procesos de la agricultura.

- Con el estudio y desarrollo de procedimientos almacenados, funciones, disparadores, índices y llaves primarias, se logró implementar la capa de acceso a datos, de la Dirección de Agricultura, y en consecuencia, el acceso a la misma.

- Con la comprobación de la integridad de los datos y el análisis del diseño de la capa de acceso de la Dirección de Agricultura en cuanto a alto rendimiento y calidad del dato, se logró la optimización de la misma lo que permitió avalar la eficiencia del diseño.

De manera general se obtuvo una base de datos que permite el almacenamiento y la recuperación de la información requerida, permitiendo mayor organización y control en la dirección de Agricultura.

RECOMENDACIONES

Luego de haber cumplido los objetivos planteados, y a partir de los resultados obtenidos, se recomienda:

- Que se realice un estudio de las nuevas tendencias para realizar mejoras en el funcionamiento de la base de datos.
- Que se ponga en práctica la utilización de este diseño de capa de acceso.
- Proporcionar un mantenimiento y soporte regular a la capa de acceso enfocados a su mejor funcionamiento.
- Actualización periódica de la base de datos, logrando que se mantenga la integridad de los datos.

REFERENCIAS BIBLIOGRÁFICAS

AARAON, FRANCO. 2011. *BASE DE DATOS DE UNA TIENDA DE ABARROTES.* INSTITUTO TECNOLÓGICO SUPERIOR DE COALCOMÁN: buenastareas, 2011.

Badilla, Ricardo Montaño. 2011. Globomedia. [En línea] 26 de 6 de 2011. [Citado el: 10 de 1 de 2012.] http://cu.globedia.com/sistema-erp-definicion-funcionamiento-ventajas-desventajas.

Carrasco, Yailin Fundora. 2011. *Diseño e implementación de la base de datos del Sistema de Gestión Académica de Postgrado.* Universidad de las Ciencias Informáticas, La Habana: s.n., 2011.

CAVSI. 2010. CAVSI. [En línea] CAVSI, 2010. http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

Community, SQL Power Architec. 2010. SQL Power Architect herramienta de modelado de datos. [En línea] 19 de octubre de 2010. http://www.tuinformaticafacil.com/herramientas-desarrollo/sql-power-architect-herramienta-de-modelado-de-datos.

2011. Cosolig. [En línea] 14 de diciembre de 2011. http://www.cosolig.org/?s=Sistema+Garux.

Daixauli, Vicente y Grau, Noemí. SlideShare. [En línea] http://www.slideshare.net/da4equipo3/historia-de-las-bases-de-datos.

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

EcuRed. 2010. EcuRed. *EcuRed.* [En línea] EcuRed, 14 de mayo de 2010. http://www.ecured.cu/index.php/Bases_de_datos.

Garcia, Rosa María Mato. 1999. Diseño de Base de Datos. *Diseño de Base de Datos.* 1999.

Giráldez Reyes, Raudel, Díaz Pérez, Maidelyn y Armas Peñas, Dayron . 2008. *PROInTec: un software para el tratamiento inteligente de datos sobre patentes .* Universidad de Pinar del Río "Hermanos Saínz Montes de Oca" : s.n., 2008.

Herrera, Manolo. 2007. blogspot. [En línea] 6 de enero de 2007. http://jmhogua.blogspot.com/2007/01/programacin-en-capas-primera-parte-capa.html.

REFERENCIAS BIBLIOGRÁFICAS

Lapuente, María Jesús Lamarca. 2010. Hipertexto. [En línea] 2010. http://www.hipertexto.info/documentos/b_datos.htm.

Mena Torres, Dayrelis, Gómez González, Giselle y Estrella Romero, Diana Rosa. 2009. *Sistema Automatizado para el Perfeccionamiento de la Diagnosis utilizada en el Tanque T-55.* Universidad de Pinar del Río Hermanos Saíz Montes de Oca : s.n., 2009.

Nogales Cobas, Pedro Manuel y Galvan Rey, Magdanis. 2010. *PROCEDIMIENTO PARA LOGRAR LA INTEROPERABILIDAD ENTRE SISTEMAS ERP EN CUBA.* UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS (UCI) : s.n., 2010.

Ortíz Valmaseda, Marcos y Rodríguez Cervantes, Susany . 2009. *O2PMigration: Herramienta para la migración de bases de datos Oracle a PostgreSQL.* Universidad de las Ciencias Informáticas, C. Habana : s.n., 2009.

Peñalver, G, Meneces, A y S, García. 2010. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

Peñalver, G, Meneses, A y García, S. 2010. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba: s.n., 2010.

PostgreSQL. 2011. PostgreSQL. *PostgreSQLl.* [En línea] septiembre de 2011. http://www.postgresql.org/about/press/presskit91/es/.

Sedano, Sergio Cano. 2010. *Los Sistemas gestores de bases de datos relacionales.* 2010.

Wong, Henry. 2011. Programando con Café. [En línea] 20 de mayo de 2011. http://www.programandoconcafe.com/2011/05/java-aplicacion-web-hibernate-con.html.

Agüero, Martín. 2007. Introducción a Spring Framework. Universidad de Palermo Introducción, Buenos Aires, Argentina: Octubre, 2007.

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

Disponibilidad de datos. 2012. *Disponibilidad de datos* [En línea] mayo de 2012. http://www.slideshare.net/utnvirtualcuicyt/disponibilidad-de-datos

GLOSARIO DE TÉRMINOS

Base de datos: desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Claves foráneas: expresan relaciones entre objetos representados, incluyendo en el esquema de una relación atributos de otra. Utilizado para relacionar tablas.

Clave primaria: conjunto de atributos de su esquema que son elegidos para servir de identificador unívoco de sus tuplas.

Entidad: se refiere a cualquier concepto del mundo real con una existencia independiente.

Querys: consiste en una cadena de consulta, normalmente se utilizan para: insertar, actualizar o editar valores de la base de datos.

Servidor: en informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al computador en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos.

ACID: es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español. Es un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

savepoint (en español, punto de recuperación) es una forma de implementar subtransacciones dentro de un ORDBMS indicando un punto dentro de una transacción de BD que puede ser "rolled back" (devuelta) sin afectar a cualquier trabajo realizado en la transacción antes de que el punto de recuperación fuera creado.

plug-in o complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es eje

cutada por la aplicación principal e interactúan por medio de la API.

IDE (en inglés integrated development environment, entorno de desarrollo integrado) es un programa informático compuesto por un conjunto de herramientas de programación.

CPU: Abreviatura de Central Processing Unit (unidad de proceso central), se pronuncia como letras separadas. La CPU es el cerebro del ordenador. A veces es referido simplemente como el procesador o procesador central**,** la CPU es donde se producen la mayoría de los cálculos. En términos de potencia del ordenador, la CPU es el elemento más importante de un sistema informático.

Clúster: Un cluster es un conjunto de computadoras que utilizan componentes comunes y actúan como si se tratase de un solo sistema u ordenador. Se utilizan clusters en la configuración de servidores y bases de datos de alto rendimiento.

UPS: (Uninterrumpible Power Supply). Fuente de alimentación ininterrumpible. Energía de seguridad que se emplea cuando la energía eléctrica de la línea se interrumpe o baja a un nivel de voltaje inaceptable. Los pequeños sistemas UPS proveen energía de baterías durante sólo unos pocos minutos; los necesarios para apagar el computador de manera ordenada.