UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

TÍTULO: Diseño de base de datos para el Sistema de Producción de la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO).

AUTOR: Henrry Maturell Andino.

TUTORAS: Ing. Yadeny Aquino Jiménez.

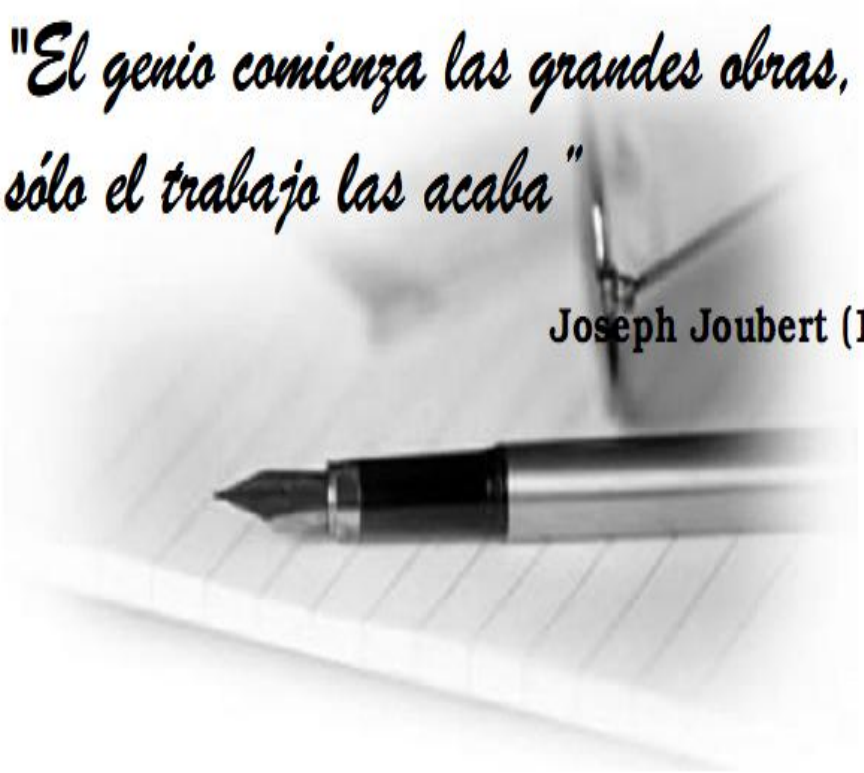
Ing. Yaquelin Cintra Almaguer.

Ciudad de La Habana, Junio del 2009

"Año 50 de la Revolución"

"El genio comienza las grandes obras, pero sólo el trabajo las acaba"

Joseph Joubert (1754-1824)



Dedicatoria

A Dios por no abandonarme jamás.

A mis padres, que siempre me han apoyado y que nunca han dejado de confiar en mí. A ellos que siempre están aquí, en las malas y en las buenas; por la educación y consejos sabios ofrecidos en los momentos oportunos.

A mis hermanos que me han soportado siempre, por todos los momentos vividos que han formado parte de las páginas de mi vida.

A mi esposa por su apoyo incondicional en cada momento de mi vida, por brindarme siempre todo su amor y por ser una persona tan especial en mi vida.

A los familiares de mi esposa, por su apoyo cuando lo necesitaba.

A mi amigos que siempre han estado cuando les he necesitado y nunca me han dado la espalda.

Agradecimientos

A mis padres por confiar en mí y darme siempre su ejemplo, por haberme ayudado a lograr la más grande meta de mi vida, sin ellos esto sería imposible.

A mis hermanos por haberme apoyado en cada momento de mi vida y brindarme siempre su ayuda.

A mi esposa que la quiero muchísimo, que siempre ha estado presente cuando la he necesitado, que me ha brindado su amor incondicionalmente y me ha enseñado a ver la luz en la oscuridad.

A la abuela de mi esposa que ya es como mi abuela también por ser tan sabia en sus consejos.

A mis suegros por saber que puedo contar con ellos y por ser parte ya de mi gran familia.

A mi padre Antonio y a toda su familia que aunque están un poco lejos no dejan de ser un factor fundamental en el desarrollo de mi carrera.

A Rafael por apoyarme en todo momento, brindarme su confianza y quererme como un hijo.

A mis familiares y los de mi esposa por siempre estar ahí cuando los necesité, dándome el aliento necesario para hacer realidad este sueño.

A mis amigos, pues aunque todos no estén presentes en este momento, me han acompañado en los buenos y malos momentos de la vida, sorteando dificultades y aprendiendo a ser mejor persona, especialmente a Toledo, Raúl, Lidier, Eliober, Yuri y el resto de mis compañeros que los recordaré siempre.

A los profesores Navarrete, Yoan, Gretchen, Pimentel, Dennis y Reynaldo con los que he podido contar en esta gran escuela.

A mis tutoras por haber dedicado parte de su preciado tiempo a orientarme y apoyarme en todo lo concerniente a la realización de este trabajo.

A mis profesores y entrenadores de la escuela deportiva por enseñarme a ser mejor persona.

En fin a todos los que de una forma u otra han hecho posible que lo que un día fue una utopía en mi vida, hoy se convierta en realidad. A todos gracias.....

Declaración de Autoría

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

Para que así conste firmamos la presente a los 17 días del mes de junio de 2009.

_____

Henrry Maturell Andino (Autor)

_____ _____

Ing. Yaquelin Cintra Almaguer (Tutor) Ing. Yadeny Aquino Jiménez (Tutor)

Datos de Contacto

Ingeniera: Yadeny Aquino Jiménez
Categoría docente: Adiestrada.
Universidad de Ciencias Informáticas, Habana, Cuba
E-mail: yaquino@uci.cu

Ingeniera: Yaquelin Cintra Almaguer
Categoría docente: Adiestrada.
Universidad de Ciencias Informáticas, Habana, Cuba
E-mail: ycintra@uci.cu

Licenciada: María Valdés Ramos.
Categoría docente: Instructor
Empresa de Perforación y Extracción de Petróleo de Occidente, Habana, Cuba
E-mail: mariavaldes@epepo.cupet

Actualmente la Informática constituye un factor muy importante para el desarrollo económico y social de todos los países. Como respuesta a la creciente complejidad de las aplicaciones informáticas y a las necesidades de almacenamiento, seguridad y gestión de la información que persiste han surgido en relativamente poco tiempo los Sistemas de Bases de Datos.

La Empresa de Perforación y Extracción de Petróleo de Occidente está orientada a la explotación eficaz de los yacimientos, la aplicación de tecnologías de avanzada y la satisfacción total de clientes y partes interesadas, además almacena gran cantidad de información relacionada con la producción y tienen como objetivos explorar y explotar yacimientos petrolíferos, para garantizar el incremento de las reservas de hidrocarburos, el crecimiento de la producción, la ampliación de los procesos de tratamiento y transportación de petróleo y gas, ofertando productos con calidad que satisfagan las necesidades del mercado nacional. Para ello utiliza herramientas ineficientes que tienen corto alcance para el almacenamiento de la información facilitando la repetición y el atraso en la entrega de la misma.

Debido a esta problemática la facultad 9 de la UCI creó un proyecto productivo con el objetivo de realizar un sistema para el control de la producción de dicha empresa. Éste trabajo de diploma es el resultado del diseño de una base de datos en PostgreSQL para el sistema a desarrollar. También se realiza un estudio relacionado con los Sistemas Gestores de Bases de Datos, los tipos de replicación, las clasificaciones de las Bases de Datos, los modelos de Bases de Datos, entre otros aspectos de interés para la investigación.

PALABRAS CLAVES

Diseño, Base de Datos, Sistema Gestor de Base de Datos.

Introducción 1
CAPÍTULO 1: Fundamentación Teórica. 5
1.1 Introducción 5
1.2 Surgimiento de las Base Datos. 5
1.3 Clasificaciones de las Bases de Datos. 6
1.4 Fases del diseño de Base de Datos. 7
1.5 Modelos de Base de Datos. 8
1.5.1 Base de Datos Jerárquicas. 8
1.5.2 Base de Datos de Red. 9
1.5.3 Base de Datos Relacionales. 10
1.5.4 Base de Datos Orientadas a Objetos. 11
1.6 Selección del Modelo de Base de Datos a utilizar en la EPEPO. 12
1.7 Diseño de la arquitectura de la Base de Datos 12
1.7.1 Arquitectura centralizada 13
1.7.2 Arquitectura descentralizada 13
1.8 Replicación de Datos. 14
1.8.1 Tipos de Réplica. 15
1.8.2 Técnicas de replicación. 16
1.8.3 Por qué el uso de replicación. 16
1.9 Análisis de los Sistemas Gestores de Base de Datos. 17
1.9.1 Oracle. 17
1.9.2 PostgreSQL. 18
1.9.3 MySQL. 20
1.10 Selección del Sistema Gestor de Base de Datos. 21
1.11 Lenguaje Unificado de Modelado 22
1.12 Metodologías de desarrollo 23
1.12.1 Programación Extrema. 24
1.12.2 Proceso Unificado de Desarrollo. 24
1.13 Herramienta para el Modelado 27
1.13.1 Visual Paradigm para UML 27
1.14 Herramienta de desarrollo EMS SQL Manager 2005 for PostgreSQL. 28

1.15 Conclusiones 28
CAPÍTULO 2: Descripción y análisis de la solución propuesta. 30
2.1 Introducción 30
2.2 Modelo de objeto del negocio. 30
2.3 Descripción de la arquitectura. 30
2.3.1 Arquitectura de PostgreSQL. 31
2.4 Selección de los requisitos. 32
2.4.1 Requisitos funcionales. 33
2.4.2 Requisitos no funcionales. 38
2. 5 Diagrama Entidad-Relación. 40
2.5.1 Descripción de las tablas. 41
2.7 Conclusiones 56
CAPÍTULO 3: Validación del diseño realizado 57
3.1 Introducción 57
3.2 Integridad de datos 57
3.3 Normalización y análisis de redundancia. 58
3.4 Descripción de la prueba realizada. 60
3.5 Trazabilidad de las acciones 61
3.6 Conclusiones 62
Conclusiones Generales 63
Recomendaciones 64
Bibliografía Citada 65
Bibliografía Consultada 68
Glosario de Términos 71
Anexos 73

Figuras

Figura 1: Enfoque orientado a los datos para el diseño de sistemas de información. **(Sánchez, 2004)** ..8
Figura 2: Ejemplo de estructura del Modelo Jerárquico. **(desarrolloweb.com, 2002)**.....9
Figura 3: Ejemplo de estructura del Modelo de Red.....9
Figura 4: Replicación maestro-esclavo. **(Pérez, y otros, 2008)**.....15
Figura 5: Replicación maestro-maestro. **(Pérez, y otros, 2008)**.....16
Figura 6: Flujos, fases e iteraciones de la Metodología RUP. **(Alma, 2006)**.....26
Figura 7: Arquitectura en capas.....31
Figura 8: Arquitectura de PostgreSQL **(Pérez, y otros, 2008)**.....32
Figura 9: Interacción entre FrondEnd y BackEnd **(Pérez, y otros, 2008)**.....32
Figura 10: Logs que se encuentran en archivos de texto en la carpeta pg_log.....62

Tabla 1: registro 41
Tabla 2: funcionalidad 41
Tabla 3: funcionalidad_rol 42
Tabla 4: rol 42
Tabla 5: usuario 42
Tabla 6: cod_capas 43
Tabla 7: capas_pozos 43
Tabla 8: cod_prod_fund 44
Tabla 9: cod_ciclo 44
Tabla 10: cod_tipo_pozo 44
Tabla 11: cod_metodos 45
Tabla 12: cod_categ_prod 45
Tabla 13: cod_pozos 46
Tabla 14: cod_tipo_instalacion 47
Tabla 15: cod_instalaciones 47
Tabla 16: cod_entidades 47
Tabla 17: cod_yacimientos 48
Tabla 18: cod_companias 48
Tabla 19: planes_prod 49
Tabla 20: cod_indicadores 49
Tabla 21: cod_tipo_destinos 49
Tabla 22: cod_destinos 50
Tabla 23: cod_meses 50
Tabla 24: cod_tipo_prod 51
Tabla 25: cod_afect_generacion 51
Tabla 26: afect_unidades_generales 51
Tabla 27: dosif_prod_q 52
Tabla 28: param_medicion 52
Tabla 29: param_sep_ent 53
Tabla 30: param_fwko 54
Tabla 31: param_pozo_cc 55
Tabla 32: trasiego 56
Tabla 33: Datos de prueba a la BD. 61

Introducción

Hoy día las aplicaciones informáticas que gestionan información son cada vez más utilizadas por las instituciones, debido a que permiten sustituir los procedimientos tradicionales de manipulación y control de la información por métodos automatizados de almacenamiento que proveen un ambiente de trabajo para la toma de decisiones y el manejo de resultados estadísticos, además permiten que la información se almacene con rapidez, eficiencia, seguridad, menos costos y más control para satisfacer las necesidades a la hora de optimizar servicios y productos.

Las empresas productoras de petróleo son un ejemplo de todo el proceso de informatización, las cuales debido al aumento del costo del petróleo a nivel internacional han tenido que establecer un control estricto de los yacimientos en sus distintos estados así como la producción y otros factores que propician una mayor ganancia en dichas empresas.

Los principales productores de petróleo a nivel mundial abogan por proyectos y actividades que hagan cada vez más eficientes sus métodos de extracción, exploración y control de la producción de petróleo.

El petróleo representa la principal fuente de energía primaria, más de la mitad se utiliza para el transporte (57.2%), la cuarta parte se destina a las actividades industriales (25.0%) y el resto a otros sectores económicos. **(Agency, Internacional Energy, 2004)** La existencia de petróleo resulta entonces estratégica para el crecimiento económico, el control sobre su producción y comercialización, ha sido históricamente motivo de disputas entre las naciones; especialmente porque los países que cuentan con las mayores reservas petrolíferas no son quienes más hacen uso de este combustible. **(Sagastume, 2007)**

En nuestro país existen dos Empresas de Perforación y Extracción de Petróleo correspondiente a la oficina de Exploración y Producción perteneciente a la Unión Cuba Petróleo (CUPET) vinculado al Ministerio de la Industria Básica (MINBAS), ellas son: la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO) y la Empresa de Perforación y Extracción de Petróleo de Centro (EPEPC), las cuales están emergidas en un proceso de informatización con el objetivo de brindar la

información en tiempo real y lograr que exista una homogeneidad en la información enviada por ambas empresas.

EPEPO es una organización de alto desempeño, con costos competitivos y un potencial humano competente, orientada a la explotación eficaz de los yacimientos, la ampliación de tecnologías de avanzada y la satisfacción total de clientes y partes interesadas, además almacena gran cantidad de información relacionada con la producción y tienen como objetivos explorar y explotar yacimientos petrolíferos, para garantizar el incremento de las reservas de hidrocarburos, el crecimiento de la producción, la ampliación de los procesos de tratamiento y transportación de petróleo y gas, ofertando productos con calidad, que satisfagan las necesidades del mercado nacional. Para ello utiliza herramientas ineficientes que tienen corto alcance para el almacenamiento de la información, facilitando la repetición y el atraso en la entrega de la misma. Estas herramientas son Microsoft Excel y Microsoft Access, esta última se utiliza en el área de yacimientos, por lo que los reportes generados no cubren todas las necesidades relacionadas con la producción de la empresa, dando lugar al **problema** que llevó a este trabajo de diploma, ¿Cómo lograr un adecuado diseño de la base de datos (BD) para el Sistema de Producción de la EPEPO?

Con vista a la solución del problema planteado, se ha propuesto como **objeto de estudio**: Diseño de BD y como **objetivo general**: Desarrollar la BD para el Sistema de Producción de la EPEPO.

En este contexto, quedarán definidos entonces, los siguientes **objetivos específicos**:

- Realizar un análisis de los métodos de diseño de BD existentes.
- Realizar el diseño de la BD del Sistema de Producción de EPEPO.
- Realizar la validación funcional del diseño de la BD.

Para un mayor control y evaluación del proceso investigativo, y además, para garantizar el cumplimiento de los objetivos propuestos, se han trazado las siguientes **tareas**:

- Realizar una búsqueda sobre los métodos de diseño de BD.
- Identificar el método de diseño más factible para el diseño de la BD del Sistema de Producción EPEPO.
- Estudiar el Modelo de Objeto.

- Realizar el Modelo Entidad-Relación (MER).
- Describir las entidades del MER.
- Identificar los métodos de prueba para la BD.

Definidas las situaciones particulares de nuestro objetivo general y las tareas que darán cumplimiento a los mismos, se puede dirigir la investigación, en esencia, hacia el diseño de la BD para el Sistema de Producción de la EPEPO, quedando así establecido el **campo de acción**.

A partir del problema anteriormente expuesto se plantea que con el adecuado diseño de la BD para el Sistema de Producción de la EPEPO, se logrará agilizar el proceso de almacenamiento y consulta de la información, resultando esta **la idea a defender** de la investigación.

Para facilitar la construcción de los modelos e hipótesis de investigación, se utilizaron **métodos científicos** tales como:

Métodos Teóricos:

- Análisis histórico - lógico para realizar el análisis histórico de los procesos de gestión de información en la EPEPO y las características de los principales servicios que brinda la producción de petróleo en el mundo.
- Análisis y síntesis para la confección del diseño teórico y metodológico de la investigación, análisis de las tendencias y tecnologías actuales de BD a nivel mundial.
- Modelación para la representación del modelo lógico y físico de la BD.

Métodos Empíricos:

- Observación para identificar las principales actividades que desarrollan los procesos de EPEPO.
- Entrevistas para recopilar información referente a los procesos que se desarrollan en EPEPO y con el fin de validar la propuesta que se presenta.

La población definida en la investigación está constituida por todos los trabajadores de EPEPO. El personal fue escogido de forma directa, pues se entrevistaron a los operadores, técnicos y tecnólogos

de cada unidad, constituyendo la mejor fuente de información y siendo esta la muestra que permitirá arribar a conclusiones; para ello se aplicó la técnica de muestreo no probabilística – intencional.

Como **posible resultado** de la investigación se espera: Diseño de la BD para el Sistema de Producción de la EPEPO.

La investigación ha quedado plasmada en tres capítulos, donde se plantean todo lo referente con la misma y con la propuesta final que ha dado como resultado. A continuación se muestra una panorámica de los temas que se abordan en cada uno de los capítulos.

Capítulo 1: Se brinda información sobre el origen, clasificaciones y desarrollo de las BD, se analizan las características del diseño de la arquitectura de las BD, se caracterizan los Sistemas Gestores de Bases de Datos (SGBD), se describen las herramientas a emplear para el diseño y confección de la BD del Sistema de Producción de EPEPO.

Capítulo 2: Se analiza el modelo de objeto del negocio, se describe la arquitectura y se realiza una breve descripción de cada tabla y los campos que las componen, así como el MER correspondiente a la BD en cuestión, teniendo en cuenta los requisitos funcionales y no funcionales trazados desde el negocio y la fase de requerimientos.

Capítulo 3: Se verá lo referente a la validación teórica del diseño, la integridad, así como la normalización y el análisis de la redundancia. También se describirán los tipos de pruebas identificados y se explicara la prueba realizada a la BD, además se analizara la trazabilidad de acciones.

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción

En el presente capítulo se brinda información sobre el origen, clasificaciones y desarrollo de las BD, se analizan las características del diseño de la arquitectura de las BD, se caracterizan los SGBD, se describen las herramientas a emplear para el diseño y confección de la BD del Sistema de Producción de EPEPO.

1.2 Surgimiento de las Base Datos.

Antes del surgimiento de las BD, la manera en que se almacenaba y trabajaba con la información era mediante los sistemas de ficheros. Un sistema de ficheros es un grupo de programas con los que trabajan los usuarios finales, donde cada programa controla sus propios datos, es decir, que no hay un lugar físico centralizado donde se almacenen los datos y al cual puedan acceder todos los que lo necesiten. **(Silvente, 2008)**

Esto implica separación y aislamiento de los datos, duplicación de datos, dependencia de datos, formatos de ficheros incompatibles, consultas fijas y proliferación de programas de aplicación. Luego aparece la necesidad de realizar un trabajo más efectivo con la información.

Inicialmente surgen los archivos secuenciales como almacenes de datos, que accedían rápido a la información pero de forma secuencial, y como alternativa a esta dificultad surgen los archivos indexados, que accedían directamente a la posición deseada del archivo. Estos métodos no cubrían en gran medida las necesidades de almacenamiento de la información por lo que se necesitaba un sistema de almacenamiento que facilitara el trabajo con operaciones complejas sin violar las restricciones, que garantizara la seguridad en el acceso de los usuarios y la integridad de la información. **(Calcines, y otros, 2008)**

Ante esta problemática surgieron las BD, las cuales son definidas por algunos autores como:

- Una colección de elementos de datos interrelacionados que pueden procesarse por una o más aplicaciones. **(W.Hansen, y otros, 2006)**

- Un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos. BD es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo. **(Almaguer, y otros, 2008)**

Los autores consultados no están menos alejados de la realidad y todos apoyan la idea de que el objetivo fundamental de las BD es precisamente almacenar un conjunto de datos que se encuentren relacionados entre sí y que puedan ser accedidos por una determinada aplicación. Esta investigación se acoge a las definiciones antes mencionadas al referirse a las BD.

1.3 Clasificaciones de las Bases de Datos.

Las BD pueden clasificarse de varias maneras, de acuerdo al criterio a elegir.

Según la variabilidad de los datos almacenados pueden clasificarse en: **(Silvente, 2008)**

- Bases de datos estáticas: Son BD de sólo lectura, utilizadas primordialmente para almacenar 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. Son aquellas BD en las que no se puede modificar la información que está guardada, es decir, solo se puede consultar.
- Bases de datos dinámicas: Son BD donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Son aquellas BD en las que la información que está almacenada está en constante cambio, ya que sobre ella se pueden realizar operaciones de inserción, actualización y eliminación así como las operaciones de consultas.

Según el contenido pueden clasificarse en:

- Bases de datos bibliográficas: Es un conjunto de referencias bibliográficas de publicaciones almacenadas informáticamente y que pueden ser recuperadas interactivamente gracias al lenguaje de consultas. **(Zurro, y otros, 2005)**
- Bases de datos de texto completo: Contienen el texto completo de la fuente de documentos que comprende la BD. Almacenan las fuentes primarias, por ejemplo, todo el contenido de las ediciones de una colección de revistas científicas. **(N.K.Malhotra, 2003)**
- Directorios: Proporcionan información sobre individuos, organizaciones y servicios, ejemplos nombres, direcciones entre otras cosas. Un ejemplo son las guías telefónicas en formato electrónico. **(Leyva, 2008)**

Teniendo en cuenta las facilidades que ofrecen las BD Dinámicas para la actualización y adicción de datos, así como para operaciones fundamentales de consulta, esta ha sido seleccionada para el desarrollo de la solución, además de la necesidad del sistema de interactuar de manera dinámica con la información que gestiona.

1.4 Fases del diseño de Base de Datos.

El diseño de BD se ha convertido en una actividad popular . Según ha avanzado la tecnología de BD, se han desarrollado metodologías y técnicas de diseño que han permitido que el diseño de la BD sea más fácil. Aunque el diseño de una BD es un proceso complejo. La complejidad se controla si se divide el problema en subproblemas y se resuelven independientemente. El diseño de una BD se descompone por lo general en 3 fases:

- Diseño conceptual.
- Diseño lógico. (Ver Anexo 2)
- Diseño físico. (Ver Anexo 3)

La primera fase, llamada diseño conceptual, produce una representación abstracta y de alto nivel de realidad creando un esquema conceptual de alto nivel, independiente del SGBD, partiendo de especificaciones de requerimientos que describan la realidad. La segunda fase, llamada diseño lógico, convierte esta representación en especificaciones que pueden implantarse en un sistema de cómputo y ser procesadas por él, además pueden desarrollarse independientemente de la elección del SGBD. La tercera fase, llamada diseño físico, es una traducción del esquema lógico al físico para el

SGBD escogido, determina la estructura de almacenamiento físico y los métodos de consulta requeridos para un acceso eficaz a los contenidos de una BD a partir de dispositivos de almacenamientos secundarios. **(B. C. Navate, 2008)**

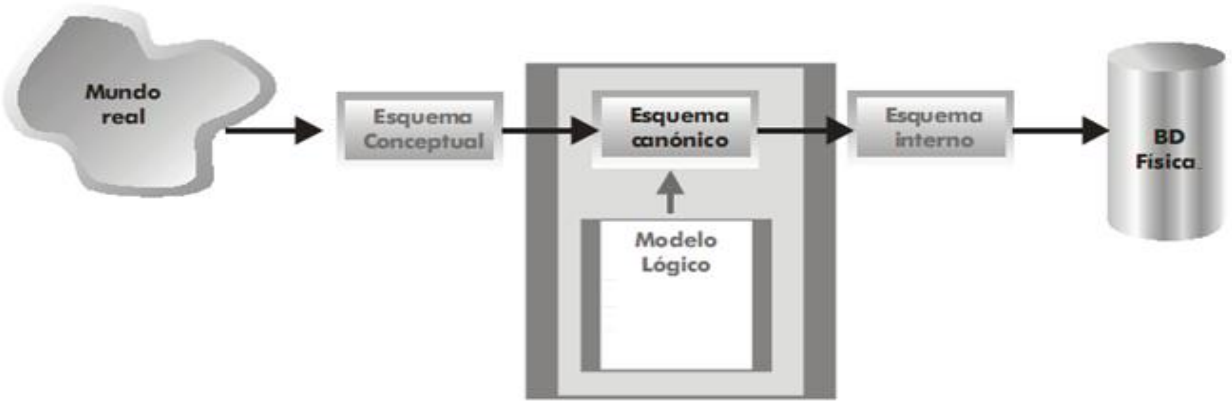


Figura 1: Enfoque orientado a los datos para el diseño de sistemas de información. **(Sánchez, 2004)**

1.5 Modelos de Base de Datos.

Los modelos de datos son vehículos para describir y representar la realidad. La calidad de los esquemas resultantes depende no sólo de la habilidad de los integrantes del equipo de desarrollo y en especial de los programadores y diseñadores de BD, sino también de las características del modelo de datos seleccionado.

A continuación se analizan el modelo jerárquico, modelo de red, modelo relacional y el modelo orientado a objeto.

1.5.1 Base de Datos Jerárquicas.

Son BD que almacenan su información en una estructura jerárquica y se organizan de forma similar a un árbol visto al revés. **(Leyva, 2008)**

Estas BD fueron desarrolladas para permitir la representación de aquellas situaciones de la vida real en las que predominan las relaciones de tipo 1: N, presentan un modelo muy rígido en el que las diferentes entidades de las que está compuesta una determinada situación se organizan en niveles

múltiples de acuerdo a una estricta relación PADRE/HIJO, de manera que un padre puede tener más de un hijo, todos ellos localizados en el mismo nivel, y un hijo únicamente puede tener un padre situado en el nivel inmediato superior al suyo. Esta estricta relación PADRE/HIJO implica que no puedan establecerse relaciones entre segmentos dentro de un mismo nivel siendo esta una de las limitaciones, además de representar eficientemente la redundancia de datos. Véase en la siguiente figura.

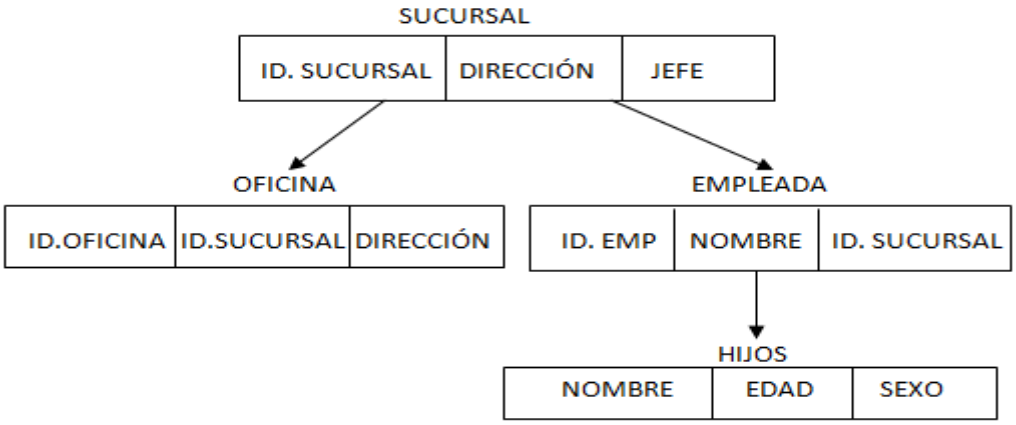


Figura 2: Ejemplo de estructura del Modelo Jerárquico. **(desarrolloweb.com, 2002)**

1.5.2 Base de Datos de Red.

El modelo de datos de red aparece a mediados de los 60 como respuesta a limitaciones del modelo jerárquico en cuanto a representación de relaciones más complejas. Es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo ya que el modelo de red permite que un nodo tenga varios nodos padres. Esto ofrece una solución eficiente al problema de la redundancia de datos. Es un modelo poco utilizado actualmente. **(Fábregas, y otros, 2007)** En la siguiente figura se puede apreciar la estructura de una BD de red:

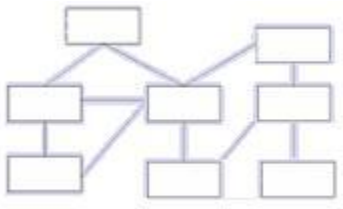


Figura 3: Ejemplo de estructura del Modelo de Red.

1.5.3 Base de Datos Relacionales.

La teoría de BD relacional es obra del investigador de IBM Edgar Codd en 1970, quien propuso además dos lenguajes de manipulación de datos para fortalecer el fundamento de este modelo, ellos son el algebra relacional y el cálculo relacional. El modelo de datos relacional organiza y representa los datos en forma de tablas o relaciones. **(W.Hansen, y otros, 2006)** Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. **(Leyva, 2008)**

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), lo cual tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la BD. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad para administrar la información.

Algunas de las ventajas del modelo relacional son: **(Pérez, y otros, 2008)**

- Define un álgebra, llamada álgebra relacional a partir de la cual se realizan todas las manipulaciones posibles sobre las relaciones, que se obtienen mediante el uso de operadores.
- Garantía de independencia de los datos.
- Conectividad garantizada con los lenguajes de programación estándar.
- Compatibilidad y estandarización.
- Favorece la normalización por ser más comprensible y aplicable.
- Garantiza la integridad referencial, así al eliminar un registro elimina todos los registros relacionados dependientes.
- Garantiza herramientas para evitar la duplicidad de registros, a través de campos claves o llaves.

A pesar de ser el modelo más usado también tiene algunos inconvenientes cuando se compara con modelos más actuales, algunos de ellos son: **(Márquez, 2002)**

- Imposibilidad de representar conocimiento en forma de reglas.

- Inexistencia de mecanismos de herencia de propiedades.
- Incompatibilidad entre los tipos de estructuras de datos que se transfieren ó desadaptación de impedancia.
- Dificultad para gestionar datos no atómicos.

1.5.4 Base de Datos Orientadas a Objetos.

Este modelo es bastante reciente y propio de los modelos informáticos orientados a objetos, trata de almacenar en la BD los objetos completos (estado y comportamiento), las cuales se les conoce como Base de Datos Orienta a Objetos (BDOO), donde los elementos de datos son objetos y las relaciones se mantienen por medio de inclusión lógica. Las entidades están representadas como clases, además contiene un método sistemático de representación de relación, y la interfaz uniforme de usuario es un sistema de mensajes que puede explorar los objetos y sus interconexiones. **(Benitez, y otros, 2002)**

Las BDOO incorporan todos los conceptos importantes del paradigma de objetos: **(Calcines, y otros, 2008)**

- Encapsulamiento: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

Presentan algunas características que le hacen reconocer se potencial, entre ellas encontramos: **(Pérez, y otros, 2008)**

- Combinación de los procedimientos de una entidad con sus datos, permite modelar tipos de datos complejos y definir operaciones entre ellos.
- Soporta relaciones de mucho a mucho.
- La cantidad de información que puede modelarse en una BDOO se incrementa, y es más fácil modelar esta información.

- Los sistemas de BDOO son capaces de tener mayores capacidades de modelado por medio de la extensibilidad.
- La reutilización de clases juega un rol vital en el desarrollo y mantenimiento más rápido de aplicaciones. Las clases genéricas son potentes, pero más importante es que ellas pueden ser usadas nuevamente.

Aunque en realidad son muy potentes hay que de decir que las BDOO son muy recientes por lo que la inmadurez del mercado lo constituye una posible fuente de problemas.

1.6 Selección del Modelo de Base de Datos a utilizar en la EPEPO.

De los modelos de BD estudiados en la investigación se selecciona el modelo relacional para el desarrollo de la BD de EPEPO, para ello se basa en determinados aspectos como:

- Alguno de los productores de software para la industria del petróleo utilizan dicho modelo y han obtenidos buenos resultados.
- Simplicidad conceptual mejorada: Ignora el almacenamiento de datos físicos, centra su atención en el panorama lógico de la BD, o sea, en la percepción humana sobre el almacenamiento. **(Silvente, 2008)**
- Diseño, administración y uso más fácil de las BD: Logra independencia de los datos e independencia estructural, y por ello, es más fácil diseñar la BD y administrar su contenido. **(Silvente, 2008)**

Por otra parte las limitaciones que presentan las BD Jerárquicas y de Red, las descartan como modelos candidatos para la solución; sus dificultades para administrar la información en una BD, complejizan cualquier propuesta de solución que se pueda presentar.

En tanto las BDOO pueden requerir más tiempo porque se pretende que promueva la reutilización futura, es el modelo más nuevo y por ende hace que surjan dudas sobre su viabilidad.

1.7 Diseño de la arquitectura de la Base de Datos

A partir de la necesidad de la EPEPO de automatizar todos sus procesos, y teniendo en cuenta su estructura organizativa, se proponen dos posibles diseños de la arquitectura.

1.7.1 Arquitectura centralizada

Se basa en la existencia de una máquina servidora que almacena los datos y las aplicaciones que los procesan. Los clientes se comportan como terminales y sólo sirven para introducir datos desde teclado.
Esta arquitectura cuenta con algunas ventajas que la distinguen como son:

- Gran nivel de seguridad de la BD.
- Fácil de administrar.
- Eliminación de los mecanismos de replicación.
- Menores costos de Software y Hardware.

Aunque tiene como inconveniente que la máquina servidora quedaría muy cargada.

1.7.2 Arquitectura descentralizada

La arquitectura descentralizada surge como una respuesta a la baja flexibilidad y al sobre cargue que existía en arquitectura centralizada. Está compuesta por un servidor central donde se encuentra la BD central a la cual se conectan varios servidores que contienen las BD locales. La interacción entre la BD central y las locales se realiza a través de la replicación, más adelante se analizará este tema.

Esta arquitectura presenta algunas ventajas que suprimen los inconvenientes de la arquitectura centralizada como son: **(Pérez, y otros, 2008)**

- Fiabilidad y disponibilidad: La falla de conectividad hacia una localidad no produce la desactivación del sistema local.
- Disminución de los niveles de procesamiento en la BD Central: Teniendo en cuenta que cada servidor contará con la información que trabajará.
- Funcionamiento offline de los sistemas locales: Si se produce el fallo de algún componente del servidor central, no demanda la paralización de los sitios de información locales.

Como resultado del estudio de las dos posibles soluciones para el diseño de la arquitectura de la BD, se propone hacer uso de la arquitectura descentralizada, esto se debe a la estructura de la EPEPO y a las condiciones de conectividad.

La EPEPO está formada por una entidad central ubicada en la dirección de la empresa y entidades afiliadas a la misma, las cuales reportan toda la información relacionada con la producción a la entidad central.

Se propone la creación de un centro de datos en la entidad central, un servidor en cada una de las entidades afiliadas. La información de cada una de las entidades afiliadas se replicará al centro de datos. Las ventajas que presenta esta arquitectura es que en caso de pérdida de conexión las afiliares seguirían trabajando con su servidor local y posteriormente de su establecimiento se replicaría la información, con esto se garantiza la no pérdida de datos, también aumentaría la seguridad en cada uno de los niveles.

1.8 Replicación de Datos.

La replicación de datos te permite copiar y distribuir idénticamente las tablas de una BD en múltiples bases de datos. Esto tiene la ventaja de que en todas las BD van a estar siempre disponible los datos correctos, además tiene algunas características que destacan el uso de la misma, como son: **(Fajardo, 2007)**

- Efectividad: Depende de la forma que los datos sean distribuidos y almacenados. A mayor efectividad, mayor será la disponibilidad de datos para ejecutar procesos paralelos.
- Alta Disponibilidad: Es la razón de tiempo prudente que un servicio puede ser accedido. En el mejor de los casos puede ser de un 100%, a pesar de los fallos que se puedan presentar en el servidor, debe existir un servidor adicional que posea alguna técnica de replicación que lo pueda suplantar en caso de ser necesario.
- Tolerancia a fallos: Garantiza un comportamiento correcto, donde efectivamente pueden existir un número finito de fallos y tipos de fallos.

- Coordinación: Cada una de las partes que conforman la BD distribuida, acuerda un consenso para realizar las invocaciones de los servicios a los objetos, que al final de la transacción debe realizarse tal y como fue solicitada, para lo cual debe utilizar algún tipo de ordenamiento.

1.8.1 Tipos de Réplica.

Existen dos tipos de replicación, Maestro-Esclavo y Maestro-Maestro.

- **Maestro-esclavo:** Sólo el maestro permite recibir consultas de lectura y escritura, mientras que el esclavo sólo de lectura. **(Pérez, y otros, 2008)** En la siguiente figura se puede apreciar la estructura de este tipo de réplica.

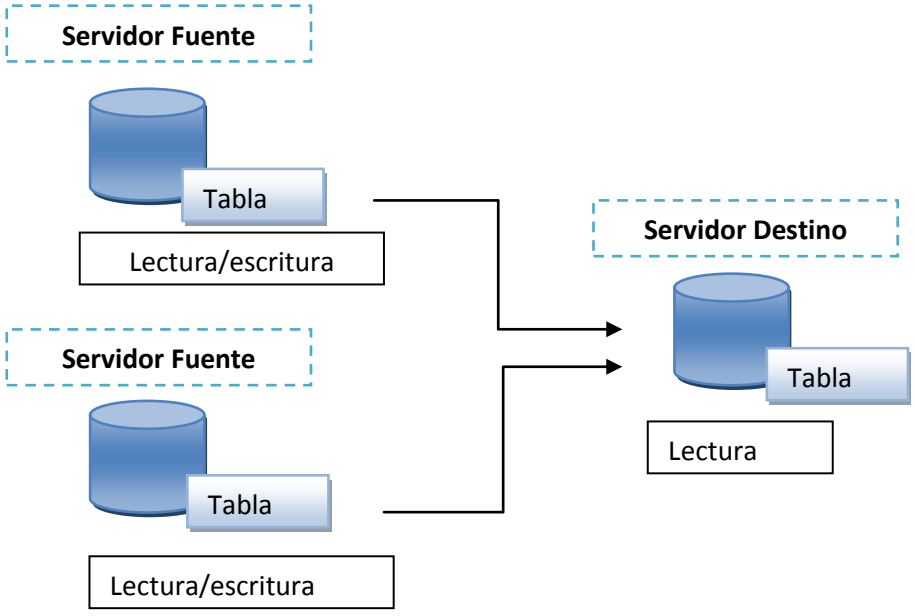


Figura 4: Replicación maestro-esclavo. **(Pérez, y otros, 2008)**

- **Maestro-Maestro:** Son varios sitios comunicándose entre sí y a la vez van actualizando sus datos, este es muy usado para equilibrar la carga de consultas, en sitios donde se conecten muchos usuarios. **(Pérez, y otros, 2008)** En la siguiente figura se puede apreciar la estructura de este tipo de réplica.

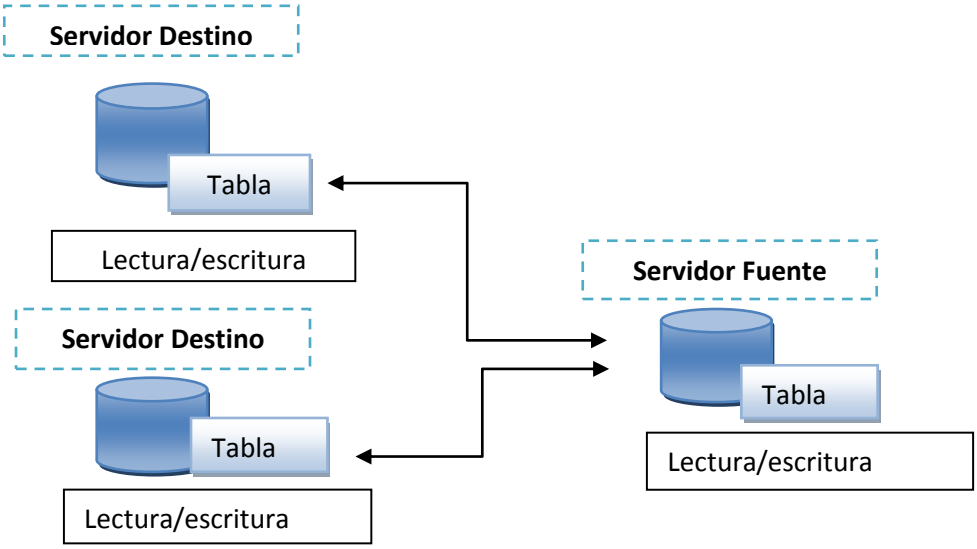


Figura 5: Replicación maestro-maestro. **(Pérez, y otros, 2008)**

1.8.2 Técnicas de replicación.

- Sincrónica (Confirmación en dos fases): Sólo se acepta la transacción cuando todos los sistemas implicados están conectados y listos para recibirla, si alguna falla, se anula el proceso.
- Asincrónica (Descarga y Recarga): Consiste en copiar los datos para un dispositivo de almacenamiento y luego esta salva a los demás servidores. Tiene la desventaja que el proceso se hace manual, y que se consultaban datos que ya estaban desactualizados.
- Instantánea: Consiste en hacer una instantánea a una tabla en un momento dado, y esto es lo que se va a replicar. Convenientemente para la réplica de datos pequeños. **(Fajardo, 2007)**

1.8.3 Por qué el uso de replicación.

Se propone el uso de replicaciones por varias razones, las cuales pueden ser categorizadas de la siguiente forma:

- Distribución de datos a otras ubicaciones: Involucra el movimiento de datos o subconjuntos de datos de una o más ubicaciones.
- Consolidación de datos desde otras ubicaciones: Una empresa puede tener sus datos en distintos sistemas distribuidos.
- Intercambio bidireccional de datos con otras ubicaciones: Los datos se pueden modificar en múltiples ubicaciones.

1.9 Análisis de los Sistemas Gestores de Base de Datos.

Un SGBD se define como el conjunto de programas que administran y gestionan la información contenida en una BD. Ayuda a realizar las siguientes acciones: **(Alvarez, 2007)**

- Definir una BD: Especificar tipos, estructuras y restricciones de datos.
- Construir la BD: Guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la BD: Realizar consultas, actualizarla, generar informes.
- Controlar la redundancia: La redundancia de datos tiene varios efectos negativos (duplicar el trabajo al actualizar, desperdicia espacio en disco, puede provocar inconsistencia de datos) aunque a veces es deseable por cuestiones de rendimiento.
- Restricción de los accesos no autorizados: Cada usuario ha de tener unos permisos de acceso y autorización.
- Cumplir las restricciones de integridad: El SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.

Un SGBD debe proporcionar una serie de lenguajes para la definición y manipulación de la BD. Estos lenguajes son los siguientes: **(Alvarez, 2007)**

- Lenguaje de definición de datos (DDL por sus siglas en ingles). Para definir los esquemas de la BD
- Lenguaje de manipulación de datos (DML por sus siglas en ingles). Para manipular los datos de la BD
- Lenguaje de control de datos (DCL por sus siglas en ingles). Para la administración de usuarios y seguridad en la BD.

1.9.1 Oracle.

Es un SGBD relacional, fabricado por Oracle Corporation, básicamente una herramienta cliente/servidor. Se considera a Oracle como uno de los SGBD más completos, destacando su soporte de transacciones, estabilidad y escalabilidad, además de ser multiplataforma. **(Leyva, 2008)**

Este SGBD tiene algunas características importantes que facilitan a los desarrolladores un mejor uso del mismo, estas son: **(Leyva, 2008)**

- Puede ser usado tanto para el manejo de información personal, como para gigantescas bibliotecas multimedia, y corre en laptops, computadoras personales (PC), microcomputadoras.
- Soporta la mayoría de los lenguajes de computación al igual que 26 idiomas diferentes.
- Corre automáticamente en más de 80 arquitecturas de hardware y software distinto sin tener la necesidad de cambiar una sola línea de código.
- Soporta datos alfanuméricos ubicados en las tradicionales "filas y columnas" de las BD, también soporta textos sin estructura, imágenes, audio y video.
- Incluye mejoras de rendimiento y de utilización de recursos.
- Está disponible en múltiples plataformas como Windows, Linux y todas las versiones de Unix.

Además este SGBD también cuenta con algunas problemáticas como son: (Leyva, 2008)

- Es un software propietario, además su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales, por norma general.
- Elevado costo de soporte técnico.

1.9.2 PostgreSQL.

El SGBD relacional orientado a objetos conocido como PostgreSQL está derivado del paquete Postgres escrito en Berkeley. Con más de una década de desarrollo tras él, PostgreSQL es el gestor de BD de código abierto más avanzado hoy día, ofreciendo control de concurrencia multiversión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación como pueden ser C, C++, Java, Perl, Python y otros. **(Alma, 2006)**

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada PGDG (PostgreSQL Global Development Group) por sus siglas en ingles.

Con el pasar del tiempo muchos desarrolladores entusiastas de los motores de BD se unieron al proyecto y entre todos comenzaron a incorporar muchas características al motor, destacándose entre ellas: **(Alvarez, 2008)**

- Cliente/Servidor: Usa una arquitectura proceso por usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- Alta concurrencia: Mediante un sistema de acceso concurrente multiversión denominado MVCC por sus siglas en inglés, PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Lenguaje Procedural: Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/PGSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL.
- Funciones: Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional.
- Herencia de Tablas: Una tabla puede heredar las funcionalidades de otra tabla especificada.
- Disparadores (Triggers): La ejecución de un procedimiento almacenado basado en una determinada acción (Insertar, Actualizar, Eliminar) sobre una tabla específica. Son además funciones enlazadas a operaciones sobre los datos.

PL/PGSQL

PL/PGSQL permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones, dando mucho más control automático que las sentencias SQL básicas. Es un lenguaje procedural cargable para el sistema de bases de datos PostgreSQL que tiene como principales objetivos: **(Pérez, y otros, 2008)**

- Crear funciones y procedimientos disparados por eventos (Triggers).
- Añadir estructuras de control al lenguaje SQL.
- Realizar cálculos complejos.
- Heredar todos los tipos definidos por el usuario, las funciones y los operadores.

- Ser fiable para el servidor.
- Fácil de usar.

Excepto en el caso de funciones de conversión de entrada/salida y de cálculo para tipos definidos, cualquier cosa que pueda definirse en funciones de lenguaje C puede ser hecho con PL/PGSQL. Es posible crear funciones complejas de cálculo y después usarlas para definir operadores o usarlas en índices funcionales. Las funciones que están escritas en PL/PGSQL aceptan parámetros y pueden devolver tipos de datos básicos o complejos como son vectores, registros, tablas y funcionan en cualquier plataforma donde PostgreSQL corra.

1.9.3 MySQL.

Es un SGBD relacional, fue creado por la empresa sueca MySQL AB, que mantiene el derecho de autor del código fuente del servidor SQL, así como también de la marca. Surgió alrededor de la década del 90, creado por Michael Windenis. Es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL. El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL). **(Alma, 2006)**

Inicialmente, MySQL carecía de algunos elementos esenciales en las BD relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre.

En las últimas versiones se pueden destacar las siguientes características principales: **(Alma, 2006)**

- El principal objetivo de MySQL es su velocidad y robustez.
- Soporta gran cantidad de tipos de datos para las columnas.
- Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.

- Cada BD cuenta con 3 archivos: Uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Flexible sistema de contraseñas y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintas lenguas.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.

MySQL tiene algunas desventajas que le imposibilitan realizar algunas tareas como estas son: **(Pérez, y otros, 2008)**

- No soporta transacciones ni subselects.
- No considera las claves ajenas. Ignora la integridad referencial, dejándola en manos del programador de la aplicación.
- Un gran porcentaje de las utilidades de MySQL no están documentadas. **(Alma, 2006)**

1.10 Selección del Sistema Gestor de Base de Datos.

Actualmente algunas de las empresas productoras de software para el petróleo utilizan SQL Server para el desarrollo de sus aplicaciones, esto tiene como inconveniente el uso de un software propietario y es por esta razón que no se utiliza SQL Server como SGBD a utilizar.

Teniendo en cuenta la necesidad de utilizar una herramienta libre, de alta tecnología, bajo costo y que se ajuste a las características de nuestro negocio se seleccionó PostgreSQL para el desarrollo de la BD del Sistema de Producción de EPEPO. Este gestor es multiplataforma, permitiendo su utilización en cualquier sistema operativo (SO) sin pérdida de información, es confiable, estable y seguro, lo que permite que la información almacenada en la BD sea controlada por las personas indicadas. Cuenta con gran escalabilidad, control de concurrencia y funcionalidades, facilitando el uso simultáneo de datos y realizaciones de cálculos, siendo esto una de las principales características del negocio, además cuenta con buenas características de almacenamiento como son:

- Máximo tamaño de BD ilimitado.

- Máximo tamaño de tabla 32 TB.
- Máximo tamaño de tupla 1.6 TB.
- Máximo tamaño de campo 1 GB.
- Máximo tuplas por tabla ilimitado.
- Máximo columnas por tabla 250 - 1600 dependiendo de los tipos de columnas.
- Máximo de índices por tabla ilimitado. **(Pérez, y otros, 2008)**

Permitiendo almacenar, actualizar e introducir datos históricos con que cuenta la EPEPO, facilitando el seguimiento de los mismos. Estas características destacan a PostgreSQL como SGBD a utilizar en el desarrollo del sistema.

1.11 Lenguaje Unificado de Modelado

Unified Modeling Lenguage (UML) es un lenguaje de modelado que indica cómo crear y leer los modelos, pero no dice cómo implementarlos. No es una guía para realizar el análisis y diseño orientado a objetos, es un lenguaje que permite modelar sistemas con tecnología orientada a objetos.
Las principales funciones de UML son: **(Orallo)**

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro sistema u otra persona lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Está compuesto por tres clases de bloques de construcción:

- Elementos: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
- Relaciones: Relacionan los elementos entre sí.
- Diagramas: Son colecciones de elementos con sus relaciones.

Hoy día, es necesario contar con un plan bien analizado. Un cliente tiene que comprender que es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades. A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros debe saber qué lugar toma su trabajo en la solución final. Por tal motivo se analiza como necesario el uso de UML como lenguaje de modelado.

1.12 Metodologías de desarrollo

La metodología es el conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de Sistemas de Información. **(Jacobson, y otros, 2000)** No existe una metodología de software universal, y aunque proporcionan guías para estimar costos, políticas y procedimientos para garantizar la calidad del software y las descripciones de los roles y responsabilidades de cada uno. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. Además la metodología en un proyecto de desarrollo de software define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo para cumplir un objetivo determinado. **(Hernández, y otros, 2001)**

En la actualidad las metodologías cuentan con dos clasificaciones, las metodologías ligeras o ágiles y las metodologías pesadas o robustas:

- Las metodologías ligeras o ágiles: Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores que las metodologías convencionales (ISO-9000, CMM, etc.) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración. **(Gallo, y otros, 2004)** Un ejemplo de esta es eXtreme Programing (XP).
- Las metodologías pesadas o robustas: Se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. Ejemplo es Rational Unified Process (RUP).

Elegir una metodología no es una cuestión simple, es algo que depende principalmente de dos factores, el tipo de proyecto que se desarrolle y el conocimiento que exista en la empresa.

1.12.1 Programación Extrema.

XP Forma parte del conjunto de metodologías ágiles que centran sus prioridades en las personas, no en los procesos, en la actualidad se proyecta a ser un modelo de desarrollo común, sencillo y adaptable a las características cambiantes y exigentes de empresas y clientes. La metodología 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 al éxito del proyecto.

Algunas de las características de XP son: **(Leyva, 2008)**

- Pruebas Unitarias: Se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelante en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- Re-fabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Esta metodología a pesar de tener algunos inconvenientes facilita que los cambios que se realicen en el proyecto sean más rápidos ya que cuentan con al menos un especialista de la materia, el cual domina el contenido relacionado con el producto que se esté desarrollando en ese momento por el equipo de desarrollo, esto facilita aclaración de dudas y aunque los desarrolladores deben ser los mejores, esto permite ganar en tiempo.

1.12.2 Proceso Unificado de Desarrollo.

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada RUP. **(Jacobson, y otros, 2000)**

El RUP es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas o software a desarrollar, para diferentes áreas de aplicación y

diferentes tipos de empresas. Pertenece a la familia de metodologías "pesadas", a causa de la gran cantidad de procesos y documentación que requiere. Se ha convertido en la metodología estándar para el desarrollo de proyectos en un entorno orientado a objeto. Es un proceso iterativo e incremental, el que facilita que sea un proceso planificado y gestionado:

- Se adapta a los cambios de los requerimientos con pocas alteraciones.
- Involucra al usuario/cliente durante el proceso.
- Permite detectar y gestionar los riesgos durante todo el ciclo de vida de proyecto.
- Se basa en la construcción de prototipos ejecutables.

Además de las características propias de los procesos iterativos se caracteriza por ser: **(Jacobson, y otros, 2000)**

- Dirigido por los Casos de Uso: Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso.
- Centrado en la Arquitectura: Relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.
- Iterativo e Incremental: Dividiéndose el proyecto en fases. Actualmente se suele hablar de ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido se denomina iteración en el proyecto en la que se realizan varios tipos de trabajo (denominados flujos). Además, cada iteración parte de la anterior incrementando o revisando la funcionalidad implementada.

RUP define por lo tanto el proceso mediante dos dimensiones: **(Frutos, y otros, 2007)**

- Dinámica o en el tiempo. Se expresa en términos de ciclos, fases, iteraciones y fechas límite o hitos.
- Estática. Se describe en términos de actividades, entregables, roles y flujos de trabajos.

En la siguiente figura se puede apreciar el comportamiento de cada flujo de trabajo en cada una de las fases.

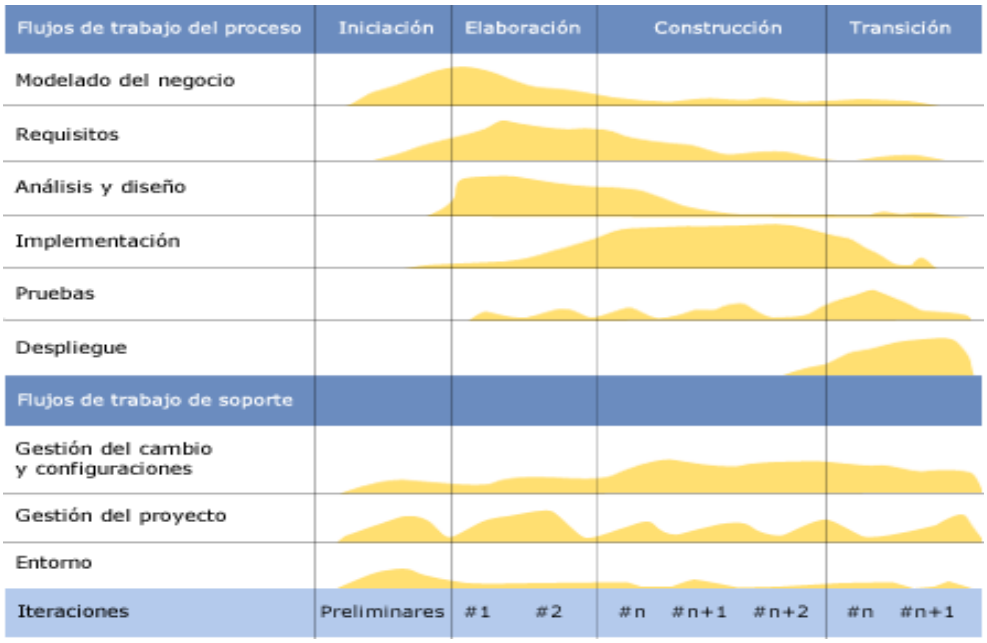


Figura 6: Flujos, fases e iteraciones de la Metodología RUP. **(Alma, 2006)**

Una vez analizadas algunas de las características de RUP y aunque pertenece a la familia de las metodologías pesadas se puede utilizar como si de una metodología ligera se tratase, para ello se siguen las siguientes prácticas:

- Utilizar un conjunto pequeño de actividades y entregables. Los entregables de RUP son opcionales, por lo cual se evita crear aquellos que no proporcionan valor. Se puede enfocar en programar de forma previa y no en documentar de forma previa.
- RUP es una metodología iterativa, en la cual los requerimientos y el diseño no se completan antes de la implementación. Éstos se van definiendo a medida que se avanza en las iteraciones.
- Aplicar modelos ágiles a UML.

Luego del análisis realizado anteriormente se puede definir que esta metodología permite desarrollar versiones cada vez más completas donde el usuario final puede percibir un avance progresivo, junto con la integración, documentación, diseño e implementación de nuevos requerimientos que con otros métodos consume demasiado tiempo, además una vez documentados todos los artefactos y actividades realizadas permite que nuevos desarrolladores puedan continuar con la realización de nuevas versiones.

1.13 Herramienta para el Modelado

Las Herramientas CASE permiten aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero pues mejora la comunicación entre los desarrolladores del proyecto y facilita un mejor intercambio de ideas con el cliente y entre los propios integrantes del equipo de desarrollo, permitiendo modelar los procesos de negocio de las empresas, aunque no son suficientes si no van acompañadas de las técnicas y metodologías adecuadas, no se implantan de la forma correcta o si no se complementan por otros aspectos relativos a la calidad. El entorno CASE debe estar integrado con otros aspectos que también persiguen la mejora del software como son las métricas, el modelo de proceso, evaluación de capacidad de desarrollo y calidad.

1.13.1 Visual Paradigm para UML

Visual Paradigm fue la herramienta CASE seleccionada en el proyecto para modelar los procesos de negocio y el desarrollo del sistema. Se utilizará para realizar el diseño del modelo de datos, es muy poderosa, fácil de usar, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, prueba y despliegue, además es multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones.

Permite representar todo tipo de diagramas UML para las distintas fases por la que transita un software en desarrollo como la captura de requisitos, análisis, diseño e implementación además de generar código desde diagramas y generar documentación, lo que facilita el trabajo de los desarrolladores.

Dentro de sus principales características se pueden citar: **(freedownloadmanager)**

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio, Actor de negocio, Documento.
- Ingeniería inversa.
- Generación de código.
- Editor de Detalles de Casos de Uso.
- Diagramas de Flujos de Datos.
- Soporte ORM (Generación de objetos Java desde la BD).

- Generación de BD (Transformación de diagramas de Entidad-Relación en tablas de BD).
- Generador de informes para generación de documentación.
- Incorpora soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

1.14 Herramienta de desarrollo EMS SQL Manager 2005 for PostgreSQL.

Para la administración del servidor de BD PostgreSQL se seleccionó la herramienta EMS SQL Manager 2005 for PostgreSQL. Esta aplicación trabaja con versiones de PostgreSQL superior a la 8.1 y soporta sus últimas características como espacios de tablas y nombres de argumentos en funciones. Tiene una interfaz gráfica muy amigable e incluye un modo guiado de trabajo muy claro. **(SQLMANAGER, 2008)**

Sus principales características son: (Almaguer, y otros, 2008)

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

1.15 Conclusiones

Con la investigación realizada en este capítulo, se pudo analizar y comprender de forma detallada algunas definiciones importantes para la comprensión del proceso de diseño de BD. Se definió PostgreSQL SGBD, se determinó el uso de RUP como metodología de desarrollo de software, Visual

Paradigm como herramienta CASE para el modelado y el uso del modelo relacional para diseñar la BD de la EPEPO, además se definió el diseño de una arquitectura descentralizada con el objetivo de utilizar la replicación de datos.

CAPÍTULO 2: Descripción y análisis de la solución propuesta.

2.1 Introducción

En el presente capítulo se analiza el modelo de objeto del negocio, las clases del diseño, además se describe la arquitectura y se realiza una breve descripción de cada tabla y los valores que almacenan las mismas, así como el MER correspondientes a la BD en cuestión teniendo en cuenta los requisitos funcionales y no funcionales trazados desde el negocio y la fase de requerimientos.

2.2 Modelo de objeto del negocio.

El modelo de objeto (MO) es uno de los artefactos que se genera en el flujo de trabajo Modelamiento del Negocio en la fase de Inicio, en el que se representan las relaciones entre los trabajadores y las entidades del negocio. Para identificar las clases persistentes que representan entidades a almacenar en la BD se consultó el modelo de objeto diseñado por los analistas. (Ver anexo 1)

2.3 Descripción de la arquitectura.

La arquitectura seleccionada para la realización del sistema de la EPEPO es en capa. Se va a estructurar en tres partes bien definidas; una capa para presentación en la que se utilizará el framework Swing para la aplicación de escritorio, una capa para la lógica de negocio en la que se utilizará el framework Spring, la Inversion of Control (IoC) y la programación orientada a aspectos (POA) y otra capa para el acceso a datos en la que se utilizará el framework Hibernate 3.2, este permite establecer relaciones entre clases Java y tablas de una BD. Para ello hace uso de ficheros xml en los que se describe dicha relación que indicará a Hibernate 3.2 como persistir las clases Java. En esta estructura una capa solo tiene relación con la capa siguiente como se puede apreciar en la siguiente figura.

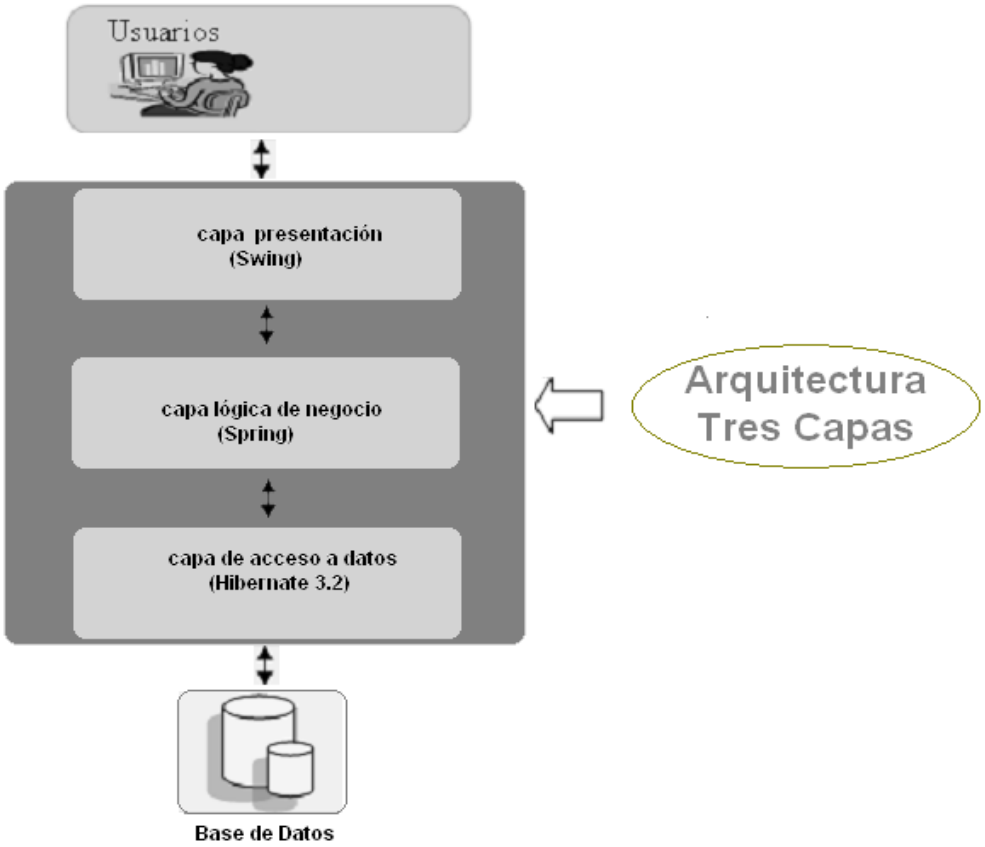


Figura 7: Arquitectura en capas.

2.3.1 Arquitectura de PostgreSQL.

PostgreSQL funciona con una arquitectura Cliente/Servidor conocida como "proceso por usuario", un proceso servidor (Postmaster) y una serie de aplicaciones cliente que realizan solicitudes de acciones a la BD. Por cada una de estas aplicaciones cliente, el proceso Postmaster crea un proceso postgres como se muestra en la siguiente figura.

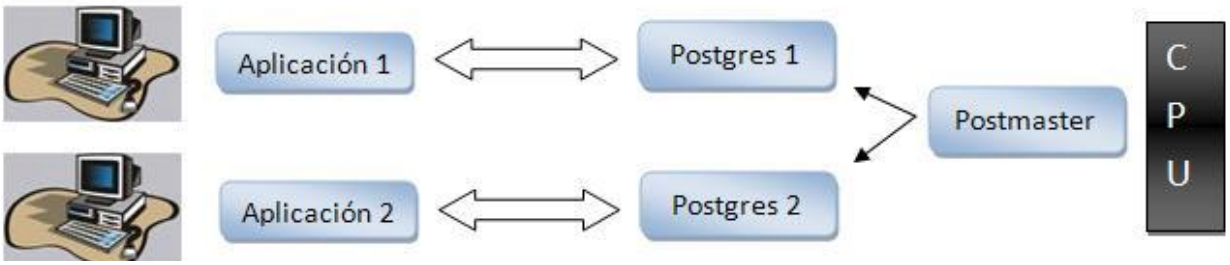


Figura 8: Arquitectura de PostgreSQL **(Pérez, y otros, 2008)**

Se ejecutan una serie de aplicaciones cliente llamadas (FrontEnd) y una serie de procesos en el servidor llamados (BackEnd), de modo que la interacción entre un programa que se ejecuta en el cliente, por ejemplo pgadmin3, una aplicación de PHP y el servidor de BD quedaría como se muestra en la siguiente figura. **(Pérez, y otros, 2008)**

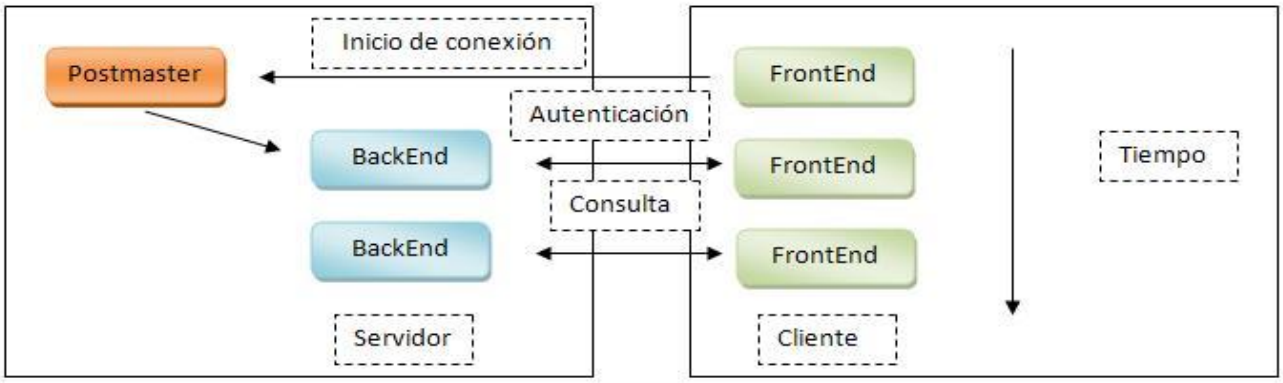


Figura 9: Interacción entre FrondEnd y BackEnd **(Pérez, y otros, 2008)**

El proceso Postmaster es el proceso inicial, que gestiona los accesos multiusuario y multiconexión y levanta la memoria compartida, está al tanto de solicitudes de nuevas conexiones, lanza procesos de atención de demanda, realizando las operaciones sobre la BD a solicitud de los clientes, además realiza el enlazado con los archivos de datos, gestionando estos ficheros, donde los ficheros pueden pertenecer a varias BD. La comunicación entre BackEnd/FrontEnd se realiza mediante TCP/IP, normalmente por el puerto 5432. **(Franco, 2008)**

2.4 Selección de los requisitos.

Para que un sistema de software funcione correctamente se debe lograr una comunicación efectiva entre los usuarios y el equipo de proyecto con el objetivo de llegar a un entendimiento de lo que hay que hacer, lo que constituye la clave del éxito en la producción de un software.

Aquí radica la importancia de identificar los requerimientos como parte del proceso de desarrollo del software. Los requisitos se clasifican en funcionales y no funcionales.

2.4.1 Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, los cuales constituyen un paso esencial para el diseño de las entidades de la BD propuesta.

RF1. Gestionar Pozo.

1.1 El sistema debe permitir insertar pozo.
1.2 El sistema debe permitir modificar pozo.
1.3 El sistema debe permitir eliminar pozo.

RF2. Gestionar Yacimiento.

2.1 El sistema debe permitir insertar yacimiento.
2.2 El sistema debe permitir modificar yacimiento.
2.3 El sistema debe permitir eliminar yacimiento.

RF3. Gestionar Método de Explotación.

3.1 El sistema debe permitir insertar métodos de explotación.
3.2 El sistema debe permitir modificar métodos de explotación.
3.3 El sistema debe permitir eliminar métodos de explotación.

RF4. Gestionar Capa.

4.1 El sistema debe permitir insertar capa.

4.2 El sistema debe permitir modificar capa.

4.3 El sistema debe permitir eliminar capa.

RF5. Gestionar Compañía.

5.1 El sistema debe permitir insertar compañía.

5.2 El sistema debe permitir modificar compañía.

5.3 El sistema debe permitir eliminar compañía.

RF6. Gestionar Tipo de Pozo.

6.1 El sistema debe permitir insertar tipo de pozo.

6.2 El sistema debe permitir modificar tipo de pozo.

6.3 El sistema debe permitir eliminar tipo de pozo.

RF7. Gestionar Instalación.

7.1 El sistema debe permitir insertar instalación.

7.2 El sistema debe permitir modificar instalación.

7.3 El sistema debe permitir eliminar instalación.

RF8. Gestionar Tipo de Instalación.

8.1 El sistema debe permitir insertar tipo de instalación.

8.2 El sistema debe permitir modificar tipo de instalación.

8.3 El sistema debe permitir eliminar tipo de instalación.

RF9. Gestionar Producción Fundamental.

9.1 El sistema debe permitir insertar producción fundamental.

9.2 El sistema debe permitir modificar producción fundamental.

9.3 El sistema debe permitir eliminar producción fundamental.

RF10. Gestionar Categoría de Pozo.

10.1 El sistema debe permitir insertar categoría de pozo.
10.2 El sistema debe permitir modificar categoría de pozo.
10.3 El sistema debe permitir eliminar categoría de pozo.

RF11. Gestionar Parámetros de Pozo del Centro Colector (CC).

11.1 El sistema debe permitir insertar parámetros de pozo del CC.
11.2 El sistema debe permitir modificar parámetros de pozo del CC.
11.3 El sistema debe permitir eliminar parámetros de pozo del CC.

RF12. Gestionar Parámetros del Separador de Entrada (PSE) del CC.

12.1 El sistema debe permitir insertar PSE del CC.
12.2 El sistema debe permitir modificar PSE del CC.
12.3 El sistema debe permitir eliminar PSE del CC.

RF13. Gestionar Parámetros del Separador de Medición (PSM) del CC.

13.1 El sistema debe permitir insertar PSM del CC.
13.2 El sistema debe permitir modificar PSM del CC.
13.3 El sistema debe permitir eliminar PSM del CC.

RF14. Gestionar Parámetros del F.W.K.O del CC.

14.1 El sistema debe permitir insertar parámetros del F.W.K.O del CC.
14.2 El sistema debe permitir modificar parámetros del F.W.K.O del CC.
14.3 El sistema debe permitir eliminar parámetros del F.W.K.O del CC.

RF15. Gestionar Dosificación de Productos Químicos (DPQ) del CC.

15.1 El sistema debe permitir insertar DPQ del CC.

15.2 El sistema debe permitir modificar DPQ del CC.

15.3 El sistema debe permitir eliminar DPQ del CC.

RF16. Gestionar Entidad.

16.1 El sistema debe permitir insertar entidad.

16.2 El sistema debe permitir modificar entidad.

16.3 El sistema debe permitir eliminar entidad.

RF17. Gestionar Tipo de Producto Químico (TPQ).

17.1 El sistema debe permitir insertar TPQ.

17.2 El sistema debe permitir modificar TPQ.

17.3 El sistema debe permitir eliminar TPQ.

RF18. Gestionar Trasiego.

18.1 El sistema debe permitir insertar trasiego.

18.2 El sistema debe permitir modificar trasiego.

18.3 El sistema debe permitir eliminar trasiego.

RF20. El sistema debe permitir generar el Parte de Producción del CC.

RF21. El sistema debe permitir graficar la información.

RF22. El sistema debe permitir autenticar usuarios.

RF23. Gestionar Tipo de Destino (TD).

23.1 El sistema debe permitir insertar TD.

23.2 El sistema debe permitir modificar TD.

23.3 El sistema debe permitir eliminar TD.

RF24. Gestionar Ciclos de Producción (CP).

24.1 El sistema debe permitir insertar CP.
24.2 El sistema debe permitir modificar CP.
24.3 El sistema debe permitir eliminar CP.

RF25. Gestionar Permiso.

25.1 El sistema debe permitir insertar permiso.
25.2 El sistema debe permitir modificar permiso.
25.3 El sistema debe permitir eliminar permiso.

RF26. Gestionar Rol.

26.1 El sistema debe permitir insertar rol.
26.2 El sistema debe permitir modificar rol.
26.3 El sistema debe permitir eliminar rol.

RF27. Exportar Información.

27.1 El sistema debe permitir exportar información a formato Excel.
27.2 El sistema debe permitir exportar información a formato PDF.

RF28. Gestionar Destino.

28.1 El sistema debe permitir insertar destino.
28.2 El sistema debe permitir modificar destino.
28.3 El sistema debe permitir eliminar destino.

RF29. Gestionar Tipo de Afectación (TA).

29.1 El sistema debe permitir insertar TA.
29.2 El sistema debe permitir modificar TA.

29.3 El sistema debe permitir eliminar TA.

RF30. Gestionar Usuario.

30.1 El sistema debe permitir insertar usuario.
30.2 El sistema debe permitir modificar usuario.
30.3 El sistema debe permitir eliminar usuario.

2.4.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable. **(Jacobson, y otros, 2000)**

Requerimientos de Hardware.

Estaciones de Trabajo:

- Periféricos Mouse y Teclado PS 2 o USB.
- 1Mb de cache L2, 256 Mb de memoria RAM.
- 5 GB de espacio libre en disco.
- CPU Pentium II a 800 MHz o superior.
- Impresora.

Servidor de Base de Datos:

- Periféricos: Mouse y Teclado PS 2 o USB.
- Tarjeta de Red.
- 1GB de RAM.
- Capacidad de Almacenamiento que garantice soporte para la BD (80 GB).
- Arquitectura hardware Intel
- Sistema operativo Windows 2000 o superior, GNU-Linux (recomendado), Unix.

Requerimientos de Software.

Estaciones de Trabajo:

- Sistema Operativo: Windows 2000 o superior, GNU-Linux, Unix.
- Java Runtime Environment versión 1.6

Servidor de Base de Datos:

- Sistema Operativo: Windows 98 o superior, GNU-Linux (recomendado), Unix.
- Gestor de Base de Datos: PostgreSQL. v8.2

Redes:

- La red existente en las instalaciones debe de soportar la transacción de paquetes de información de al menos unos 10 a razón de 2 ó 3 Mb/s.
- Para hacer más fiable la aplicación debe de estar protegida contra fallos de corriente y de conectividad, para lo que se deberán parametrizar los tiempos para realizar copias de seguridad. Primero replicarán los Centros Colectores, luego las Baterías y por último el Despacho de Producción, el Despacho Central trabajará directamente con la BD general. Todo esto se realizará teniendo en cuenta el tiempo de mejor tráfico en la red, o sea, a partir de las 12 am.

Seguridad:

- Se utilizarán las reglas de la “programación segura”, se deberá hacer un fuerte tratamiento de excepciones desde la capa de Presentación hasta la capa de Acceso a Datos incluyendo el trabajo con la Base de Datos. Para reforzar este aspecto los desarrolladores se apoyarán en los framework seleccionados
- Los usuarios de la BD solamente tendrán acceso a las tablas de las Bases de Datos a las cuales se les designe según el rol que desempeñe, no se utilizarán usuarios con privilegios administrativos para realizar las conexiones entre servidores.

- Se registrarán y auditarán las trazas dejadas por los usuarios al realizar las diferentes operaciones en el sistema. La programación de las auditorías será programada según corresponda.
- La asignación de usuarios y sus opciones sobre el sistema se garantizará desde el subsistema de Administración.

Portabilidad, escalabilidad, reusabilidad:

- El sistema deberá poder ser utilizado desde cualquier plataforma de software (Sistema Operativo).
- El sistema deberá hacer un uso racional de los recursos de hardware de la máquina, sobre todo en las PC clientes.
- La aplicación se construirá utilizando estándares y patrones, para facilitar su integración futura con componentes desarrollados por cualquier empresa y garantizar la posibilidad de un mantenimiento ágil.
- De acuerdo a las condiciones económicas del país, el sistema deberá minimizar la cantidad de gastos de despliegue.
- Se desarrollará cada pieza del sistema en forma de componentes (subsistemas) con el fin de reutilizarlos para futuras versiones.
- La documentación de la arquitectura deberá ser reutilizable para poder documentarla como una familia de productos.

2. 5 Diagrama Entidad-Relación.

Desde hace mucho tiempo los desarrolladores han buscado un sistema que les permita visualizar una BD, sus tablas, el contenido de éstas, las claves y las interrelaciones con otras tablas. El MER permite efectuar esta representación de forma sencilla, mediante entidades con sus respectivos atributos e interrelaciones. Esto resulta bastante interesante porque permite tratar mediante el MER problemas concretos de la realidad, aunque a la hora de construir modelos sobre realidades concretas es cuando surgen los problemas. Las entidades tienen muchos atributos diferentes, de los cuales se debe aprender a elegir los que se necesitan. De igual forma sucede con las relaciones entre las entidades.
El MER consta de 32 entidades interrelacionadas, cada una cuenta con sus respectivos atributos y estos a su vez tienen sus dominios. (Ver anexo 3)

2.5.1 Descripción de las tablas.

Tabla 1: registro

Nombre de la tabla: registro			
Descripción: Almacena un histórico de todas las acciones que realizan los usuarios.			
Atributo	Tipo	Llave	Descripción
id_registro	integer	Llave Primaria (PK)	Identificador del registro.
fecha	timestamp		Fecha en que se registra el registro.
accion	char		Acción que realiza el usuario.
descripcion	varchar		Descripción del registro.
ip	varchar		IP desde el cual se realiza la acción.
tabla	varchar		Tabla a la cual se le realiza la acción.
usuario	varchar	Llave Foránea (FK)	Usuario que realiza la acción.

Tabla 2: funcionalidad

Nombre de la tabla: funcionalidad			
Descripción: Almacena las funcionalidades que se pueden realizar en el sistema.			
Atributo	Tipo	Llave	Descripción
id_funcionalidad	integer	PK	Identificador de la funcionalidad.

nombre	varchar		Nombre de la funcionalidad.

Tabla 3: funcionalidad_rol

Nombre de la tabla: funcionalidad_rol			
Descripción: Almacena las funcionalidades que pueden realizar los roles en el sistema.			
Atributo	Tipo	Llave	Descripción
id_rol	integer	PK , FK	Identificador del rol.
id_funcionalidad	integer	PK , FK	Identificador de la funcionalidad.

Tabla 4: rol

Nombre de la tabla: rol			
Descripción: Almacena los roles del sistema.			
Atributo	Tipo	Llave	Descripción
id_rol	integer	PK	Identificador del rol.
nombre	varchar		Nombre del rol.
descripcion	varchar		Descripción del rol.

Tabla 5: usuario

Nombre de la tabla: usuario			
Descripción: Almacena los usuarios que interactúan con la aplicación.			
Atributo	Tipo	Llave	Descripción

usuario	varchar	PK	Almacena el usuario.
nombre	varchar		Almacena el nombre correspondiente al usuario.
contrasenna	varchar		Contraseña del usuario.
estado	bit		Especifica si el usuario está conectado.
entidad	integer	FK	Identificador de la entidad a la que pertenece el usuario.
id_rol	integer	FK	Identificador del rol al que pertenece el usuario.
id_inst	varchar	FK	Identificador de la instalación a la que pertenece el usuario.

Tabla 6: cod_capas

Nombre de la tabla: cod_capas			
Descripción: Almacena las capas en la que puede producir un pozo.			
Atributo	Tipo	Llave	Descripción
cod_capas	integer	PK	Identificador de la capa.
descripcion	varchar		Nombre de la capa.

Tabla 7: capas_pozos

Nombre de la tabla: capas_pozos			
Descripción: Almacena el histórico de las capas en la que ha producido un pozo.			
Atributo	Tipo	Llave	Descripción

cod_capas	integer	PK , FK	Identificador de la capa.
nombre	varchar	PK , FK	Nombre del pozo.
entidad	integer	PK , FK	Almacena la entidad a la que pertenece el pozo.
fecha_prod	date		Fecha en que comenzó la producción del pozo en la capa seleccionada.
activa	bit		Si está activa la capa o no.

Tabla 8: cod_prod_fund

Nombre de la tabla: cod_prod_fund			
Descripción: Almacena la producción fundamental del pozo.			
Atributo	Tipo	Llave	Descripción
cod_prod	char	PK	Identificador de la producción fundamental del pozo.
descripcion	varchar		Descripción de la producción fundamental.

Tabla 9: cod_ciclo

Nombre de la tabla: cod_ciclo			
Descripción: Almacena los ciclos de los pozos.			
Atributo	Tipo	Llave	Descripción
cod_ciclo	char	PK	Identificador del ciclo en que se encuentra un pozo.
descripcion	varchar		Descripción del ciclo.

Tabla 10: cod_tipo_pozo

Nombre de la tabla: cod_tipo_pozo			
Descripción: Almacena la etapa en que se encuentra el pozo.			
Atributo	Tipo	Llave	Descripción
cod_tipo	char	PK	Identificador de la etapa en que se encuentra el pozo.
descripcion	varchar		Descripción del codificador de tipo de pozo, es decir el nombre de la etapa en que se encuentra el pozo.

Tabla 11: cod_metodos

Nombre de la tabla: cod_metodos			
Descripción: Almacena el método que se emplea para extraer el fluido del pozo.			
Atributo	Tipo	Llave	Descripción
cod_metodos	char	PK	Identificador del método que se utiliza para extraer el fluido del pozo.
descripcion	varchar		Descripción del codificador del método de extracción.

Tabla 12: cod_categ_prod

Nombre de la tabla: cod_categ_prod			
Descripción: Almacena las categorías de los pozos.			
Atributo	Tipo	Llave	Descripción
cod_categoria	char	PK	Identificador de la categoría del pozo.
descripcion	varchar		Nombre de la categoría.

Tabla 13: cod_pozos

Nombre de la tabla: cod_pozos			
Descripción: Almacena la información de los pozos.			
Atributo	Tipo	Llave	Descripción
nombre	varchar	PK	Nombre del pozo.
entidad	integer	PK , FK	Almacena la entidad a la que pertenece el pozo.
coef_a	bit		Almacena si se aplica o no el coeficiente de ajuste al pozo.
coef_p	bit		Almacena si se aplica o no el coeficiente de prorrateo al pozo.
descripcion	varchar		Descripción del pozo.
cod_categoria	char	FK	Identificador de la categoría del pozo.
cod_ciclo	char	FK	Identificador del ciclo en que se encuentra un pozo.
cod_metodos	char	FK	Identificador del método que se utiliza para extraer el fluido del pozo.
cod_prod	char	FK	Identificador de la producción fundamental del pozo.
cod_tipo	char	FK	Identificador de la etapa en que se encuentra el pozo.
id_propietario	integer	FK	Identificador del propietario del pozo.
id_operario	integer	FK	Identificador del operador del pozo.
cod_yac	varchar	FK	Identificador del yacimiento al que pertenece el pozo.
id_inst	varchar	FK	Identificador de la instalación a la que pertenece el pozo.

Tabla 14: cod_tipo_instalacion

Nombre de la tabla: cod_tipo_instalacion			
Descripción: Almacena el nombre de los distintos tipos de instalaciones que puede tener una entidad.			
Atributo	Tipo	Llave	Descripción
cod_tipo	char	PK	Identificador del tipo de instalación.
descripcion	varchar		Nombre del tipo de instalación.

Tabla 15: cod_instalaciones

Nombre de la tabla: cod_instalaciones			
Descripción: Almacena las instalaciones con que cuenta una entidad.			
Atributo	Tipo	Llave	Descripción
id_inst	varchar	PK	Identificador de la instalación.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación.
descripcion	varchar		Descripción de la instalación.
caracteristicas	varchar		Características de la instalación.
cod_tipo	char	FK	Identificador del tipo de instalación.

Tabla 16: cod_entidades

Nombre de la tabla: cod_entidades
Descripción: Almacena las entidades que pueden trabajar con el sistema.

Atributo	Tipo	Llave	Descripción
entidad	integer	PK	Almacena el identificador de la entidad.
siglas	varchar		Almacena las siglas de la entidad.
descripcion	varchar		Almacena la descripción de la entidad.

Tabla 17: cod_yacimientos

Nombre de la tabla: cod_yacimientos			
Descripción: Almacena los yacimientos de las entidades.			
Atributo	Tipo	Llave	Descripción
cod_yac	varchar	PK	Almacena el identificador del yacimiento.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece el yacimiento.
nombre	varchar		Almacena el nombre del yacimiento.
descripcion	varchar		Almacena la descripción del yacimiento.

Tabla 18: cod_companias

Nombre de la tabla: cod_companias			
Descripción: Almacena las compañías que operan o son propietaria de los pozos.			
Atributo	Tipo	Llave	Descripción
id_comp	integer	PK	Identificador de la compañía.
nombre	varchar		Nombre de la compañía.

Tabla 19: planes_prod

Nombre de la tabla: planes_prod			
Descripción: Almacena los planes de producción de la empresa.			
Atributo	Tipo	Llave	Descripción
anno	integer	PK	Año en que se planifica el plan de producción de una entidad.
indicador	varchar	PK , FK	Identificador del indicador del plan.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece el plan.

Tabla 20: cod_indicadores

Nombre de la tabla: cod_indicadores			
Descripción: Almacena los indicadores con que se establecen los planes de producción, es decir, los distintos tipos de planes que influyen en la formación de un plan de producción.			
Atributo	Tipo	Llave	Descripción
indicador	varchar	PK	Identificador del indicador.
descripcion	varchar		Descripción del indicador.
tipo	char		Tipo de indicador.

Tabla 21: cod_tipo_destinos

Nombre de la tabla: cod_tipo_destinos			
Descripción: Almacena los tipos de movimiento del fluido.			
Atributo	Tipo	Llave	Descripción

cod_tipo	char	PK	Identificador del tipo de movimiento.
descripcion	varchar		Descripción del tipo de movimiento del fluido.

Tabla 22: cod_destinos

Nombre de la tabla: cod_destinos			
Descripción: Almacena todos los lugares a donde se trasiega el fluido de la empresa.			
Atributo	Tipo	Llave	Descripción
id_destino	integer	PK	Identificador del destino.
cod_tipo	char	FK	Identificador del tipo de movimiento del fluido.
descripcion	varchar		Almacena la descripción del destino.

Tabla 23: cod_meses

Nombre de la tabla: cod_meses			
Descripción: Almacena los meses que conforman un plan de producción.			
Atributo	Tipo	Llave	Descripción
id_mes	integer	PK	Identificador del mes.
nombre	varchar		Nombre del mes.
planificado	integer		Cantidad de petróleo planificado para ese mes.
anno	integer	FK	Año en que se planifica el plan de producción.
indicador	varchar	FK	Identificador del indicador del plan.
entidad	integer	FK	Identificador de la entidad a la que pertenece el plan.

Tabla 24: cod_tipo_prod

Nombre de la tabla: cod_tipo_prod			
Descripción: Almacena los tipos de productos químicos.			
Atributo	Tipo	Llave	Descripción
cod_q	integer	PK	Identificador del tipo de producto químico.
descripcion	varchar		Descripción del tipo de producto químico.

Tabla 25: cod_afect_generacion

Nombre de la tabla: cod_afect_generacion			
Descripción: Almacena los tipos de afectaciones.			
Atributo	Tipo	Llave	Descripción
cod_afect	integer	PK	Identificador del tipo de afectación.
descripcion	varchar		Descripción del tipo de afectación.

Tabla 26: afect_unidades_generales

Nombre de la tabla: afect_unidades_generales			
Descripción: Almacena las afectaciones de las instalaciones.			
Atributo	Tipo	Llave	Descripción
fecha	date	PK	Fecha y hora en que se registra la afectación.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación afectada.

id_inst	varchar	PK , FK	Identificador de la instalación que se afecta.
horas_afectadas	integer		Cantidad de horas afectadas.
cod_afect	integer	PK , FK	Identificador del tipo de afectación.

Tabla 27: dosif_prod_q

Nombre de la tabla: dosif_prod_q			
Descripción: Almacena los productos químicos que se aplican en una instalación.			
Atributo	Tipo	Llave	Descripción
id_dosif	integer	PK	Identificador de la dosificación de productos químicos.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación.
id_inst	varchar	PK , FK	Identificador de la instalación que aplica la química.
producto	varchar		Producto que se aplica.
entrada_quimica	integer		Almacena la entrada de química a la instalación.
entrada_cons	integer		Almacena la entrada de consumo de química.
consumo	integer		Almacena el consumo real de química a la instalación.
fecha_aplicado	date		Almacena la fecha en que fue aplicada la química.
cod_q	integer	FK	Identificador del tipo de producto químico.

Tabla 28: param_medicion

Nombre de la tabla: param_medicion

Descripción: Almacena los parámetros de las mediciones realizadas a los pozos.			
Atributo	Tipo	Llave	Descripción
fecha_medicion	date	PK	Fecha en que se realiza la medición.
nombre	varchar	PK , FK	Nombre del pozo en medición.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece el pozo en medición.
bsw	float		Porciento de agua.
p_separador	float		Presión del separador.
q_liq	float		Caudal de líquido.
q_pet	float		Caudal de petróleo.
agua	float		Agua que contiene el fluido.
rgp	float		Relación Gas-Petróleo.
q_gas	float		Caudal de gas.
observaciones	varchar		Observaciones que se le realicen a la medición.

Tabla 29: param_sep_ent

Nombre de la tabla: param_sep_ent			
Descripción: Almacena los parámetros del separador de entrada.			
Atributo	Tipo	Llave	Descripción
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación.
id_inst	varchar	PK , FK	Identificador de la instalación que controla los parámetros.

fecha	date	PK	Almacena la fecha que se recogen los parámetros.
p_entrada	float		Presión de entrada.
p_sep	float		Presión del separador.
flujo_gas	float		Flujo de gas.
nivel_separ	float		Nivel del separador.
p_gaseod	float		Presión del gaseoducto
p_linea_iny_agua	float		Presión en línea de inyección de agua.

Tabla 30: param_fwko

Nombre de la tabla: param_fwko			
Descripción: Almacena los parámetros del FWKO (Separador trifásico).			
Atributo	Tipo	Llave	Descripción
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación.
id_inst	varchar	PK , FK	Identificador de la instalación.
fecha	date	PK	Fecha en que se registran los parámetros.
temp_ent_fluido	integer		Temperatura de entrada de fluido.
temp_sal_fluido	integer		Temperatura de salida de fluido.
temp_sal_uno	integer		Temperatura de salida de gases Q1.
temp_sal_dos	integer		Temperatura de salida de gases Q2.
bsw_ent	float		BSW de entrada.
bsw_sal	float		BSW de salida.

nivel_agua	float		Nivel de agua del fwko.
flujo_gas	integer		Flujo de gas.
desarenado	float		Desarenado del fluido.
flujom_agua_fwko	integer		Lectura del flujómetro de agua de fwko.
flujom_agua_iny	integer		Lectura del flujómetro de agua de inyección.
horas_trab	integer		Horas trabajadas del fwko.

Tabla 31: param_pozo_cc

Nombre de la tabla: param_pozo_cc			
Descripción: Almacena los parámetros del pozo que controla el centro colector.			
Atributo	Tipo	Llave	Descripción
fecha_ param	date	PK	Fecha en que se controlan los parámetros.
nombre	varchar	PK , FK	Nombre del pozo.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece el pozo.
pt	integer		Presión del tubing.
pc	integer		Presión del casing.
choque	integer		Almacena los choques.
torque	float		Almacena los torques.
horas_trabajadas	integer		Almacena las horas trabajadas del pozo.
rpm	integer		Revoluciones por minutos.

fecha_medicion	date	FK	Fecha en que se realiza la medición del pozo.

Tabla 32: trasiego

Nombre de la tabla: trasiego			
Descripción: Almacena los trasiegos realizados por las instalaciones.			
Atributo	Tipo	Llave	Descripción
id_trasiego	integer	PK	Identificador del trasiego.
entidad	integer	PK , FK	Identificador de la entidad a la que pertenece la instalación.
id_inst	varchar	PK , FK	Identificador de la instalación que realiza el trasiego.
paila	varchar		Chapa de la paila.
conduce	integer		Número del conduce.
cantidad	integer		Cantidad que se va a trasegar.
cunna	varchar		Cuña en la que se va a realizar el trasiego.
observaciones	varchar		Observaciones del trasiego.
id_destino	integer	FK	Identificador del destino.
fecha	date		Fecha en que se realiza el trasiego.

2.7 Conclusiones

En este capítulo se realizó un estudio del modelo de objeto realizado por los analistas, se analizaron los requisitos funcionales y no funcionales, se realizó y se describió el modelo físico de datos que solucionará y cumplirá con las expectativas del cliente, se definieron 32 tablas que representan los datos relevantes a almacenar en la BD.

CAPÍTULO 3: Validación del diseño realizado

3.1 Introducción

En este capítulo se realizará la validación teórica del diseño teniendo en cuenta algunos aspectos como son: la integridad, la normalización y el análisis de la redundancia. También se describirán los tipos de pruebas identificados y se explicará la prueba realizada a la BD, además se analizará la trazabilidad de acciones.

3.2 Integridad de datos

La implementación de la integridad relacional en una BD consiste en establecer las reglas de consistencia correspondientes a los datos requeridos de las tablas, el chequeo de los valores válidos y únicos, así como la integridad de las llaves primarias y foráneas.

El modelo de datos relacional incluye varios tipos de restricciones de integridad en BD Relacional como son:

- Datos Requeridos: Establece que una columna tenga un valor no nulo. Se define efectuando la declaración de una columna como NOT NULL cuando la tabla que contiene la columna se crea por primera vez, como parte de la sentencia CREATE TABLE. **(Caballero, y otros, 2007)**
- Integridad de entidad: Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla, sino la BD 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 fallara**. (Santana, y otros, 2007)**
- Integridad de Transacciones: Define los estados por los que una tupla puede pasar válidamente, como son introducido, pendiente, seleccionado, enviado, cancelado y terminado. Está encargada de asegurar que el estado de una determinada tupla, no pase de un estado inicial a uno final, sin haber pasado por los estados intermedios. En el caso de la BD de la EPEPO no se encuentran restricciones de este tipo.
- Integridad referencial: Otra de las reglas, es la llamada restricción de integridad referencial que se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de las dos

relaciones. Establece que una tupla en una relación que haga referencia a otra, deberá referirse a un valor existente en esa relación. Un ejemplo de aplicación de este concepto en el modelo de datos es el siguiente: se asegura que los atributos cod_ciclo y cod_prod de la tabla cod_pozos sean llaves foráneas, debido a que provienen y son llaves primarias de las entidades cod_ciclo y cod_prod_fund respectivamente. Además cumplen con las restricciones siguientes:

1. Los datos cod_ciclo y cod_prod fundamental de la tabla cod_pozos tienen los mismos dominios que en sus entidades de origen.
2. Al introducir los datos del cod_ciclo y el cod_prod en la tabla cod_pozos, éstos tienen que encontrarse en su entidad de origen, es decir, la tabla cod_pozos no puede tener un cod_ciclo que no exista en la tabla cod_ciclo y su cod_prod debe encontrarse de igual forma en la tabla cod_prod_fund.

Por lo planteado anteriormente se comprueba que en el modelo de datos se cumple con la regla de integridad referencial.

3.3 Normalización y análisis de redundancia.

La normalización es un proceso primordial en el diseño de una BD, contribuye a eliminar la redundancia de la información y evitar problemas de inserción o actualización de los datos, y con esto, optimizar la gestión de la información almacenada.

Para normalizar una BD se deben realizar varias fases en orden, siendo entre ellas dependientes, lo que indica que para que una BD se encuentre en una determinada Forma Normal (FN), primeramente tiene que cumplir con las reglas de los niveles inferiores de normalización. Existen 6 FN que garantizan un buen diseño de la BD.

- Primera Forma Normal (1FN).
- Segunda Forma Normal (2FN).
- Tercera Forma Normal (3FN).
- Forma Normal Boyce-Codd (BCNF).
- Cuarta Forma Normal.
- Quinta Forma Normal o Forma Normal de Proyección-Unión.

Llegar a una forma BCNF puede llegar a ser complicado, debido a esto en muchas ocasiones es suficiente con alcanzar la 3NF para lograr un buen diseño.

Con la tercera forma normal los procesos de refinamiento sucesivo a realizar en la normalización de las relaciones, nos llevan a descomponer la tabla primaria hasta obtener tablas sencillas donde la redundancia y repeticiones sean mínimas. **(Márquez., 2008)**

Desarrollado el proceso de normalización, se puede comprobar que satisface la 1FN, se demuestra que no hay presencia de atributos multivaluados y sólo se permiten valores atómicos, esto implica que cada tabla contenga para cada atributo un sólo dominio y que tiene un valor único para cada fila.

Para la 2FN se tuvo en cuenta primeramente que estuviera en 1FN y que existieran dependencias totales, es decir, que todos los atributos no llaves dependan de todos los atributos que componen la llave y no de solo una parte de ellos.

El análisis del diseño propuesto demostró que el mismo se encuentra 2FN, debido a que todos los atributos no llaves tienen una dependencia total de los atributos llaves, ejemplo de esto en la tabla *cod_entidades:*

cod_entidades (entidad, siglas, descripcion_entidad, id_inst, descripcion_inst, caracteristicas_inst, cod_tipo, descripcion_tipo)

Como se puede apreciar esta tabla no se encuentra en 2FN porque existen atributos que no dependen de la llave primaria, llevando a 2FN nos quedaría:

cod_entidades (entidad, siglas, descripcion)

cod_instalaciones (entidad, id_inst, descripcion_inst, caracteristicas_inst, cod_tipo, descripcion_tipo)

Para la 3FN se analizó primeramente que estuviera en 2FN y que no aparecieran dependencias transitivas entre los atributos no llaves, es decir, ningún atributo debe depender de otro que no sea la llave primaria, por ejemplo, la siguiente tabla no está en 3FN, es notable una relación transitiva entre el *cod_tipo* y *descripcion_tipo* como se muestra a continuación.

cod_instalaciones (entidad, id_inst, descripcion_inst, caracteristicas_inst, cod_tipo, descripcion_tipo)

Dando solución a lo planteado anteriormente quedaría:

cod_instalaciones (entidad, id_inst, descripcion, caracteristicas, cod_tipo)

cod_tipo_instalacion (cod_tipo, descripcion)

La BD de la EPEPO se encuentra en 3FN, de esta manera se resuelven los problemas de inconsistencia y en gran medida la redundancia en sus tablas, en algunas tablas existe redundancia de datos que son necesarios como por ejemplo las llaves primarias que se reflejan en otras tablas a través de las relaciones teniendo en cuenta las caracteristicas del negocio.

3.4 Descripción de la prueba realizada.

Para garantizar que la BD cumpla con los requisitos de los usuarios y funcione sin ningún problema se necesitan buenos procedimientos de pruebas. Las pruebas de validación funcional normalmente se centran en las operaciones siguientes:

- La carga de la BD se realiza sin violar la integridad de los datos.
- La correcta interfaz de las aplicaciones con la BD.
- El rendimiento del sistema satisface las necesidades para las que se adquirió el SGBD.

Para cumplir con las operaciones antes mencionadas se pueden realizar pruebas de stress, de carga y de volumen. La prueba de stress lleva al sistema al máximo punto, para poder medir sus capacidades y las condiciones en las cuales trabaja realizando una cantidad definida de peticiones y procesos. La prueba de carga tiene como objetivo simular una carga de producción real, conexiones del usuario y observar cómo se comportan la aplicación y la BD bajo cargas extremas, esto permite identificar los problemas potenciales de rendimiento, antes que se ponga en producción la BD. Por otra parte la prueba de volumen sostiene al elemento de prueba a grandes cantidades de datos, para determinar si se alcanzan los límites que hacen fallar al software. También identifica la carga máxima o continua que el elemento de prueba puede manejar por un período dado.

En el caso de la BD de la EPEPO se realizó la prueba de volumen, permitiendo un llenado voluminoso e inteligente de la BD, sin violar la integridad de la misma, se utilizó EMS PostgreSQL Data Generator 2005. Esta herramienta es muy poderosa en la generación de datos de pruebas para BD construidas en PostgreSQL. Permite seleccionar tablas y columnas para generar datos, definir rangos de valores, así como la cantidad de tuplas que se desea generar. Una de las ventajas de esta herramienta es que

genera los datos que provienen de otras tablas para evitar errores de integridad referencial. El empleo de esta herramienta posibilitó chequear la integridad de la BD, obteniéndose muy buenos resultados.

En la siguiente tabla se muestran la cantidad de datos insertados en la BD (10454) de forma eficiente apreciando en un tiempo relativamente pequeño (45 segundos) en relación con la cantidad de datos insertados.

Tabla 33: Datos de prueba a la BD.

Cantidad de datos insertados	Tiempo de inserción (seg)
775	0:00:06
1401	0:00:06
765	0:00:06
1419	0:00:06
1547	0:00:06
4547	0:00:15

3.5 Trazabilidad de las acciones

La trazabilidad permite dejar rastro de las acciones que el usuario realiza con los datos. Por ejemplo, si un usuario realiza cualquier acción (eliminar, insertar, seleccionar, mostrar) sobre un registro de alguna tabla, queda constancia de dicho suceso, es decir, desde donde, quien y cuando realizó la acción. De esta forma, puede hacerse un seguimiento de todas las acciones que se han realizado sobre un registro concreto o sobre una tabla. O todas las acciones que ha realizado un usuario concreto. Pudiendo conformar un histórico de las acciones sobre los datos. La trazabilidad puede realizarse sobre tablas o sobre procesos. Incluso pueden crearse trazabilidades sobre errores o conflictos de la aplicación. **(GALIANO, 2006)**

Los SGBD cuentan con un excelente mecanismo que le permite guardar logs para seguir las trazas en la BD. Estos logs son usados para registrar datos o información sobre (quién, qué, cuándo, dónde) interactuó con la BD convirtiéndose en un registro oficial de eventos.

En el SGBD PostgreSQL los logs se encuentran en archivos de texto en la carpeta pg_log en ellos se registra la fecha, hora y acción realizada en la BD como se muestra en la siguiente figura.

```
2009-04-07 08:00:12 LOG:  database system was interrupted at 2009-04-06 22:20:59 Hora estándar de Sudamérica O.
2009-04-07 08:00:12 LOG:  checkpoint record is at 0/13E1D10
2009-04-07 08:00:12 LOG:  redo record is at 0/1BE1D10; undo record is at 0/0; shutdown TRUE
2009-04-07 08:00:12 LOG:  next transaction ID: 0/60164; next OID: 124211
2009-04-07 08:00:12 LOG:  next MultiXactId: 1; next MultiXactOffset: 0
2009-04-07 08:00:12 LOG:  database system was not properly shut down; automatic recovery in progress
2009-04-07 08:00:12 LOG:  record with zero length at 0/1BE1D60
2009-04-07 08:00:12 LOG:  redo is not required
2009-04-07 08:00:12 LOG:  database system is ready
2009-04-07 14:31:44 WARNING:  there is no transaction in progress
2009-04-07 14:32:58 WARNING:  there is no transaction in progress
2009-04-07 14:36:56 WARNING:  there is no transaction in progress
2009-04-07 14:37:38 WARNING:  there is no transaction in progress
2009-04-07 14:39:17 WARNING:  there is no transaction in progress
```

Figura 10: Logs que se encuentran en archivos de texto en la carpeta pg_log.

3.6 Conclusiones

En este capítulo se han analizado aspectos a tener en cuenta para realizar un buen diseño de BD, como son: la integridad de los datos, la normalización y análisis de redundancia, la prueba realizada y la trazabilidad de las acciones. Se normalizó la BD hasta 3FN para eliminar redundancias e inconsistencias de datos. Se describieron algunos tipos de pruebas como son las pruebas de stress, de carga y de volumen; realizándose la prueba de volumen. Se utilizaron mecanismos para almacenar la trazabilidad, permitiendo tener un control de las acciones que realiza cada usuario en la BD.

Conclusiones Generales

Una vez culminada la investigación se arriban a las siguientes conclusiones:

- Como resultado de este trabajo se obtuvo el diseño de una BD relacional, implementada en el SGBD PostgreSQL, que aporta una alta escalabilidad, y facilidades para el manejo de la información de la producción de petróleo en la EPEPO.
- El diseño de la BD propuesta, se validó de forma teórica, analizando su integridad y seguridad, además de llevarla hasta la 3FN, a partir de lo que se puede plantear que se cuenta con un diseño robusto, libre de redundancias innecesarias.
- Mediante el uso de la herramienta EMS PostgreSQL Data Generator 2005 se realizó la prueba de volumen, a través de la cual se demostró la resistencia del diseño ante un volumen elevado de información.
- Se realizó la replicación de datos la cual facilitó el intercambio de información de una o más ubicaciones, utilizando la herramienta Dew Replicador sobre JEE.

Recomendaciones

Debido a los resultados de todo el proceso de investigación realizado y basado en la experiencia adquirida se proponen las siguientes recomendaciones:

- Mantener sobre la BD un estricto cumplimiento del proceso de mantenimiento y copias de seguridad periódicas, logrando así que se mantenga la fiabilidad y funcionamiento óptimo de la BD.
- Realizar la implementación de la BD.

Bibliografía Citada

Agency, Internacional Energy. 2004. Key World Energy Statistics. . 2004.

Alma, Enríquez Toledo. 2006. MySQL. [Online] 2006. [Cited: marzo 08, 2009.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf..

Almaguer, Yaquelin Cintra and Núñez, Yandy León. 2008. Aplicación de Técnicas de Base de Datos para el soporte de indicadores dinámicos en el Sistema Automatizado para el Control de Gestión de Indicadores de Refinación (SACGIR). Ciudad de la Habana : s.n., 2008.

Alvarez, Brenda Tipsy. 2008. [Online] octubre 05, 2008. [Cited: marzo 20, 2009.] http://bren-tipsy.blogspot.com/2008/10/2-departamental-metodologa-de.html.

Alvarez, Sara. 2007. Equipo DesarrolloWeb.com. [Online] julio 31, 2007. [Cited: febrero 26, 2009.] http://www.desarrolloweb.com..

B. C. Navate. 2008. Diseño conceptual de las bases de datos. Un enfoque de entidad-relación. [Online] 2008. [Cited: febrero 02, 2009.] http://books.google.com/books?hl=es&lr=&id=-DP-0puz338C&oi=fnd&pg=PR18&dq=clasificaci%C3%B3n+de+las+base+de+datos&ots=_PgK-Drs13&sig=iZzh5672..

Benitez, R and Roa, I. 2002. Bases de Datos de Orientadas a Objetos. 2002.

Caballero, William Bravo and Trujillo, Rodolfo Ávila. 2007. Diseño de la base de datos para el módulo de IPC de la Oficina Nacional de Estadísticas. Ciudad de La Habana : s.n., 2007.

Calcines, Anabel Vega and Ruiz, Alfredo Rodriguez. 2008. LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del Modulo Liberación Analitica. Ciudad de la Habana : s.n., 2008.

desarrolloweb.com. 2002. Modelos de bases de datos. [Online] 2002. [Cited: marzo 03, 2009.] http://www.desarrolloweb.com..

Fábregas, Yuniesky and Fernández, Daniel. 2007. Administración, configuración y optimización de un Sistema de BD Descentralizado en Oracle Database 10g release 2. Ciudad de la Habana : s.n., 2007.

Fajardo, Norge. 2007. Sistema de replicación para base de datos distribuida en PostgreSQL. Ciudad de la Habana : s.n., 2007.

Franco, Emanuel Calvo. 2008. Charla sobre Postgresql para las Jornadas Regionales de Software Libre . Argentina : s.n., 2008.

freedownloadmanager. Paradigma Visual para UML. Free Download Manager. [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/..

Frutos, Evaristo de and Guerra, Ignacio. 2007. Metodología. Metodología de Desarrollo de Software. [Online] Diciembre 12 , 2007. [Cited: marzo 26, 2009.] https://eduforge.org/docman/view.php/230/3180/SUMA_Metodologia_v0.1.pdf..

GALIANO, J. L. M. 2006. Generadores de código. 2006.

Gallo, Eliza and Vergara, Mikel. 2004. European Software Institute. [Online] 2004. [Cited: marzo 25, 2009.] www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc..

Hernández, E, Hernández, J and C, Lizandra. 2001. C++ Estandar . ITP Paraninf. 2001.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000. biblioteca.uci.cu. El Proceso Unificado de Desarrollo de Software. [Online] 2000. [Cited: marzo 22, 2009.] http://biblioteca.uci.cu/titdigitales.htm#igs.

Leyva, Arnoldo Domínguez. 2008. Modelo Lógico y Físico de la Base de Datos correspondiente a los Módulos de Investigación Criminalística y Estadística del Proyecto CICPC. Ciudad de la Habana : s.n., 2008.

Márquez, Merche. 2002. Base de datos Orientado a Objetos. Diseño de Sistemas de Base de Datos. [Online] abril 12, 2002. [Cited: marzo 16, 2009.] http://www3.uji.es/~mmarques/e16/teoria/cap2.pdf..

Márquez., Mercedes. 2008. Sistemas de bases de datos. [Online] 2008. [Cited: febrero 26, 2009.] http://www3.uji.es/~mmarques/f47/apun/node4.html.

N.K.Malhotra. 2003. Investigaciones de Mercado. Un enfoque practico. Cuarta edición. 2003.

Orallo, E. H. El Lenguaje Unificado de Modelado (UML).

Pérez, Erich Mario Gómez and Gálvez, Ariel Torres. 2008. Administración y optimización de un Sistema de Base de Datos Descentralizado, en PostgreSQL. Ciudad de la Habana : s.n., 2008.

Sagastume, Mara Luz Polanco. 2007. Características de la Producción y el Comercio Mundial del Petróleo. Universidad de San Carlos de Guatemala. : s.n., 2007.

Sánchez, Jorge. 2004. Principios sobre Bases de Datos Relacionales. [Online] 2004. [Cited: febrero 05, 2009.] http://creativecommons.org/licenses/by-nc-sa/2.0/.

Santana, Leonel de Jesús Castillo and Sánchez, Yaniel Alvarez. 2007. Base de Datos para la Residencia de la UCI. Ciudad de la Habana : s.n., 2007.

Silvente, Serguéi Frómeta. 2008. Modelo lógico y físico de la base de datos del módulo de Investigaciones Forenses del proyecto CICPC. Ciudad de la Habana : s.n., 2008.

SQLMANAGER. 2008. EMS Manager for PostgreSQL Overview. [Online] Diciembre 4, 2008. [Cited: marzo 28, 2009.] http://www.sqlmanager.net/products/postgresql/manager.

W.Hansen, Gary and Hansen, James V. 2006. Diseño y Administración de Base de Datos. 2da Edición. 2006.

Zurro, A. Martin and C, J. 2005. Atencion Primaria. Conceptos, organización y práctica clínica. Quinta edición. 2005.

Bibliografía Consultada

Agency, Internacional Energy. 2004. Key World Energy Statistics. . 2004.

Alma, Enríquez Toledo. 2006. MySQL. [Online] 2006. [Cited: marzo 08, 2009.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf..

Almaguer, Yaquelin Cintra and Núñez, Yandy León. 2008. Aplicación de Técnicas de Base de Datos para el soporte de indicadores dinámicos en el Sistema Automatizado para el Control de Gestión de Indicadores de Refinación (SACGIR). Ciudad de la Habana : s.n., 2008.

Alvarez, Brenda Tipsy. 2008. [Online] octubre 05, 2008. [Cited: marzo 20, 2009.] http://bren-tipsy.blogspot.com/2008/10/2-departamental-metodologa-de.html.

Alvarez, Sara. 2007. Equipo DesarrolloWeb.com. [Online] julio 31, 2007. [Cited: febrero 26, 2009.] http://www.desarrolloweb.com..

B. C. Navate. 2008. Diseño conceptual de las bases de datos. Un enfoque de entidad-relación. [Online] 2008. [Cited: febrero 02, 2009.] http://books.google.com/books?hl=es&lr=&id=-DP-0puz338C&oi=fnd&pg=PR18&dq=clasificaci%C3%B3n+de+las+base+de+datos&ots=_PgK-Drs13&sig=iZzh5672..

Benitez, R and Roa, I. 2002. Bases de Datos de Orientadas a Objetos. 2002.

Caballero, William Bravo and Trujillo, Rodolfo Ávila. 2007. Diseño de la base de datos para el módulo de IPC de la Oficina Nacional de Estadísticas. Ciudad de La Habana : s.n., 2007.

Calcines, Anabel Vega and Ruiz, Alfredo Rodriguez. 2008. LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del Modulo Liberación Analitica. Ciudad de la Habana : s.n., 2008.

desarrolloweb.com. 2002. Modelos de bases de datos. [Online] 2002. [Cited: marzo 03, 2009.] http://www.desarrolloweb.com..

Fábregas, Yuniesky and Fernández, Daniel. 2007. Administración, configuración y optimización de un Sistema de BD Descentralizado en Oracle Database 10g release 2. Ciudad de la Habana : s.n., 2007.

Fajardo, Norge. 2007. Sistema de replicación para base de datos distribuida en PostgreSQL. Ciudad de la Habana : s.n., 2007.

Franco, Emanuel Calvo. 2008. Charla sobre Postgresql para las Jornadas Regionales de Software Libre . Argentina : s.n., 2008.

Free Download Manager. Visual Paradigma para UML. [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/..

freedownloadmanager. Paradigma Visual para UML. Free Download Manager. [Online] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/..

Frutos, Evaristo de and Guerra, Ignacio. 2007. Metodología. Metodología de Desarrollo de Software. [Online] Diciembre 12 , 2007. [Cited: marzo 26, 2009.] https://eduforge.org/docman/view.php/230/3180/SUMA_Metodologia_v0.1.pdf..

GALIANO, J. L. M. 2006. Generadores de código. 2006.

Gallo, Eliza and Vergara, Mikel. 2004. European Software Institute. [Online] 2004. [Cited: marzo 25, 2009.] www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc..

Hernández, E, Hernández, J and C, Lizandra. 2001. C++ Estandar . ITP Paraninf. 2001.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000. biblioteca.uci.cu. El Proceso Unificado de Desarrollo de Software. [Online] 2000. [Cited: marzo 22, 2009.] http://biblioteca.uci.cu/titdigitales.htm#igs.

Leyva, Arnoldo Domínguez. 2008. Modelo Lógico y Físico de la Base de Datos correspondiente a los Módulos de Investigación Criminalística y Estadística del Proyecto CICPC. Ciudad de la Habana : s.n., 2008.

ManualesGratis. 2006. [Online] Septiembre 23, 2006. http://www.manualesgratis.org/manuales/bases-de-datos/postgresql/postgresql--conceptos-de-arquitectura-01836.html..

Márquez, Merche. 2002. Base de datos Orientado a Objetos. Diseño de Sistemas de Base de Datos. [Online] abril 12, 2002. [Cited: marzo 16, 2009.] http://www3.uji.es/~mmarques/e16/teoria/cap2.pdf..

Márquez., Mercedes. 2008. Sistemas de bases de datos. [Online] 2008. [Cited: febrero 26, 2009.] http://www3.uji.es/~mmarques/f47/apun/node4.html.

N.K.Malhotra. 2003. Investigaciones de Mercado. Un enfoque practico. Cuarta edición. 2003.

Orallo, E. H. El Lenguaje Unificado de Modelado (UML).

Pérez, Erich Mario Gómez and Gálvez, Ariel Torres. 2008. Administración y optimización de un Sistema de Base de Datos Descentralizado, en PostgreSQL. Ciudad de la Habana : s.n., 2008.

Sagastume, Mara Luz Polanco. 2007. Características de la Producción y el Comercio Mundial del Petróleo. Universidad de San Carlos de Guatemala. : s.n., 2007.

Sánchez, Jorge. 2004. Principios sobre Bases de Datos Relacionales. [Online] 2004. [Cited: febrero 05, 2009.] http://creativecommons.org/licenses/by-nc-sa/2.0/.

Santana, Leonel de Jesús Castillo and Sánchez, Yaniel Alvarez. 2007. Base de Datos para la Residencia de la UCI. Ciudad de la Habana : s.n., 2007.

Silvente, Serguéi Frómeta. 2008. Modelo lógico y físico de la base de datos del módulo de Investigaciones Forenses del proyecto CICPC. Ciudad de la Habana : s.n., 2008.

SQLMANAGER. 2008. EMS Manager for PostgreSQL Overview. [Online] Diciembre 4, 2008. [Cited: marzo 28, 2009.] http://www.sqlmanager.net/products/postgresql/manager.

Tasé, Licet Cardero and Zayas, Daliagna Bicet. 2007. Diseño e implementación de la Base de Datos del Sistema Informático de los Tribunales Militares de Región. Ciudad de la Habana : s.n., 2007.

W.Hansen, Gary and Hansen, James V. 2006. Diseño y Administración de Base de Datos. 2da Edición. 2006.

Zurro, A. Martin and C, J. 2005. Atencion Primaria. Conceptos, organización y práctica clínica. Quinta edición. 2005.

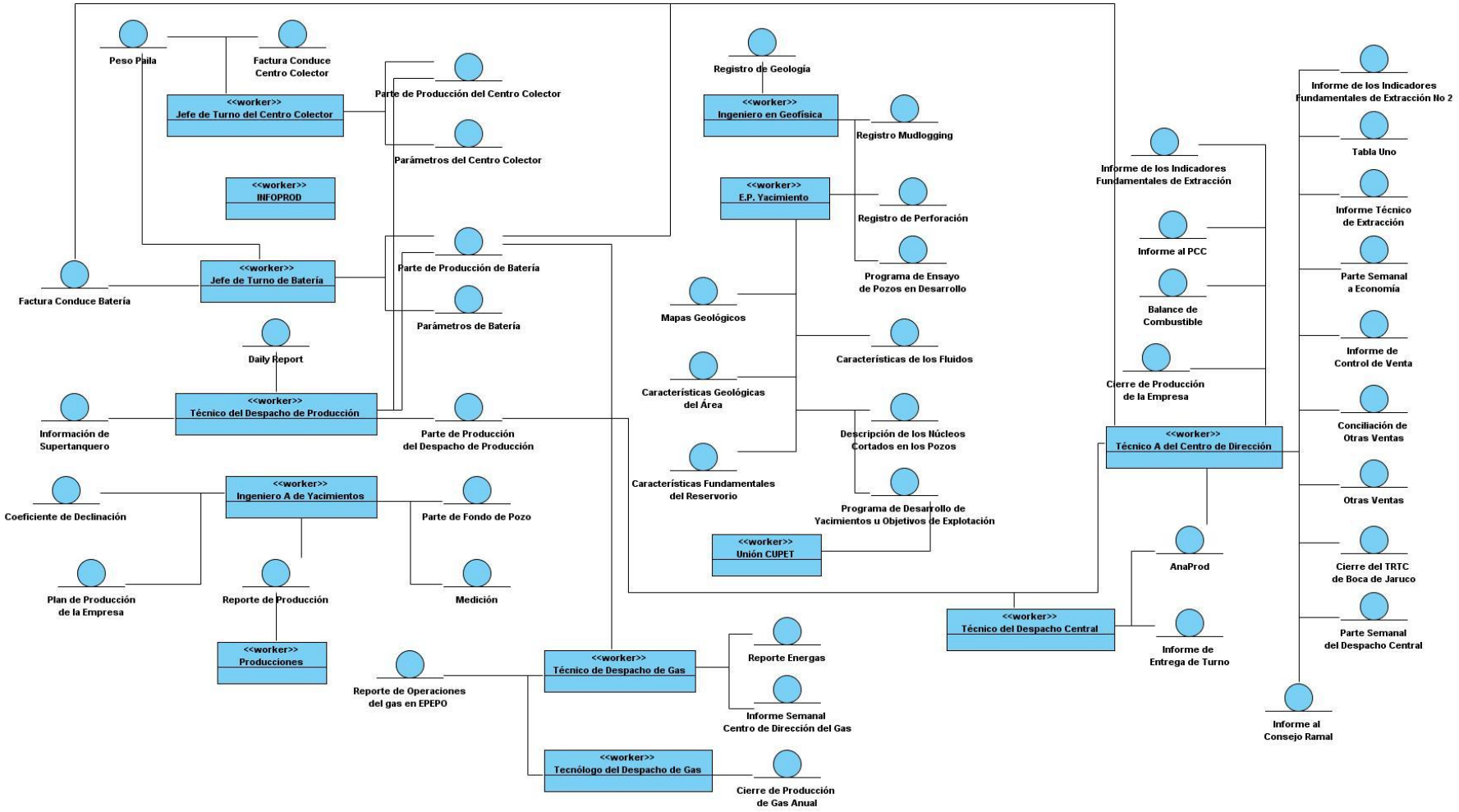
Glosario de Términos

- BSW: Porciento de agua.
- Framework: Es una estructura de soporte, utilizada en el desarrollo de software, sobra la cual pueden organizarse y desarrollarse proyectos. Representa una arquitectura de software y provee una estructura y una metodología de trabajo.
- FWKO: (Free Water Knock Out): Separador trifásico usado para darle tratamiento al crudo.
- Herramientas CASE: (Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas que pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas.
- Logs: Es un registro oficial de eventos durante un periodo de tiempo en particular.
- Java: Es un lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros.
- Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos.
- Normalización: Proceso de reducción sobre una estructura de datos que procura aumentar la integridad, disminuir la redundancia y las dependencias funcionales de esa estructura.
- Replicación: proceso de copia y mantenimiento de bases de datos múltiples que hacen un sistema distribuido. Consiste en dos servidores: uno maestro y uno esclavo. El maestro registra todas las consultas que provocan cambios en la base de datos y las almacena en un fichero de registro binario. El servidor esclavo se conecta al maestro, lee las condiciones en el fichero binario y las ejecuta contra su copia local.
- Trazabilidad: Aptitud de reconstruir la historia, la utilización o la localización de un producto por medio de identificaciones registradas.
- XML (Extensive Markup Language): El XML es considerado como un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas. Se trata de un estándar del Wold Wide Web Consortium (W3C) cuyo objetivo es crear unas reglas

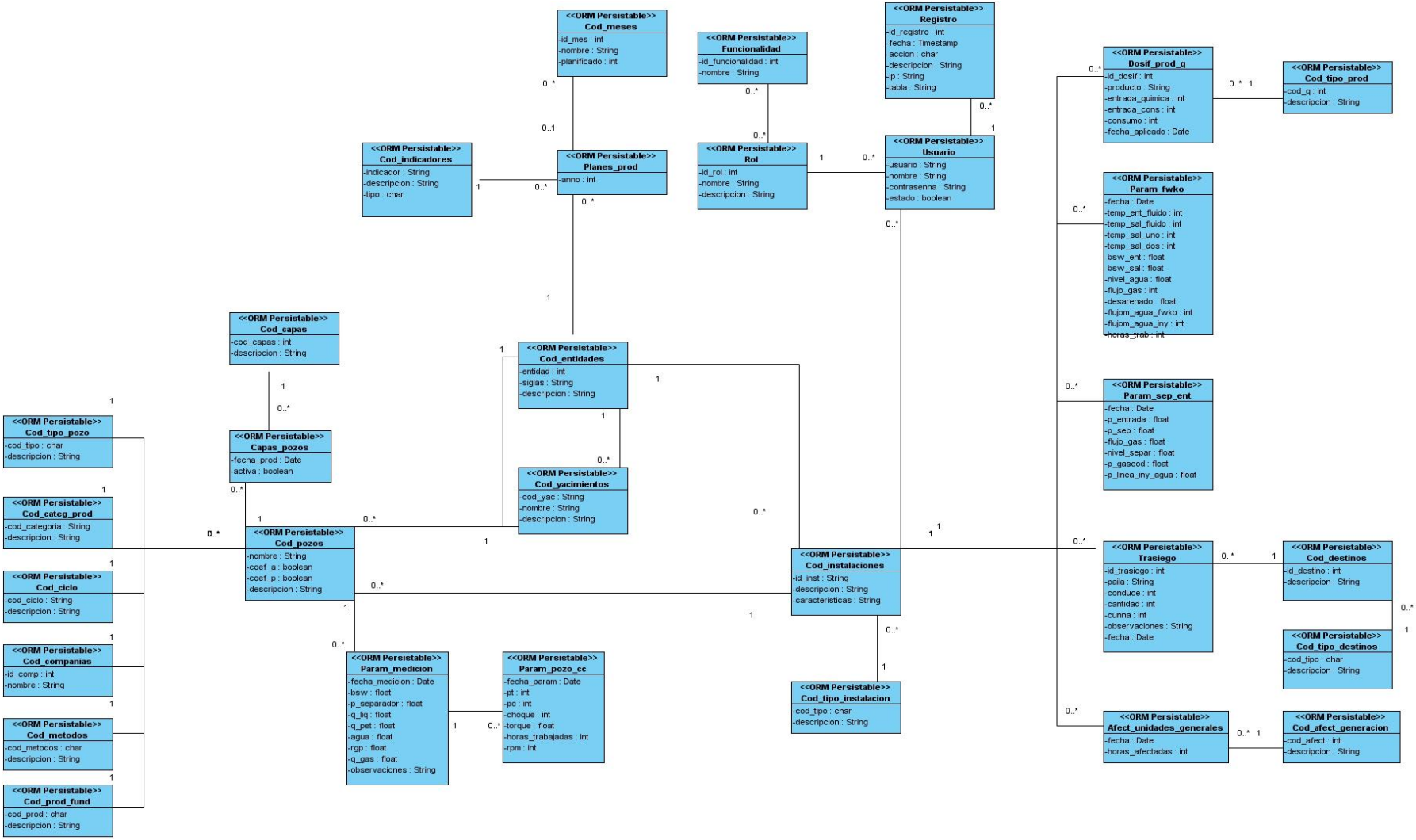
básicas para permitir el intercambio de información estructurada entre aplicaciones, y en particular, entre aplicaciones web.

Anexos

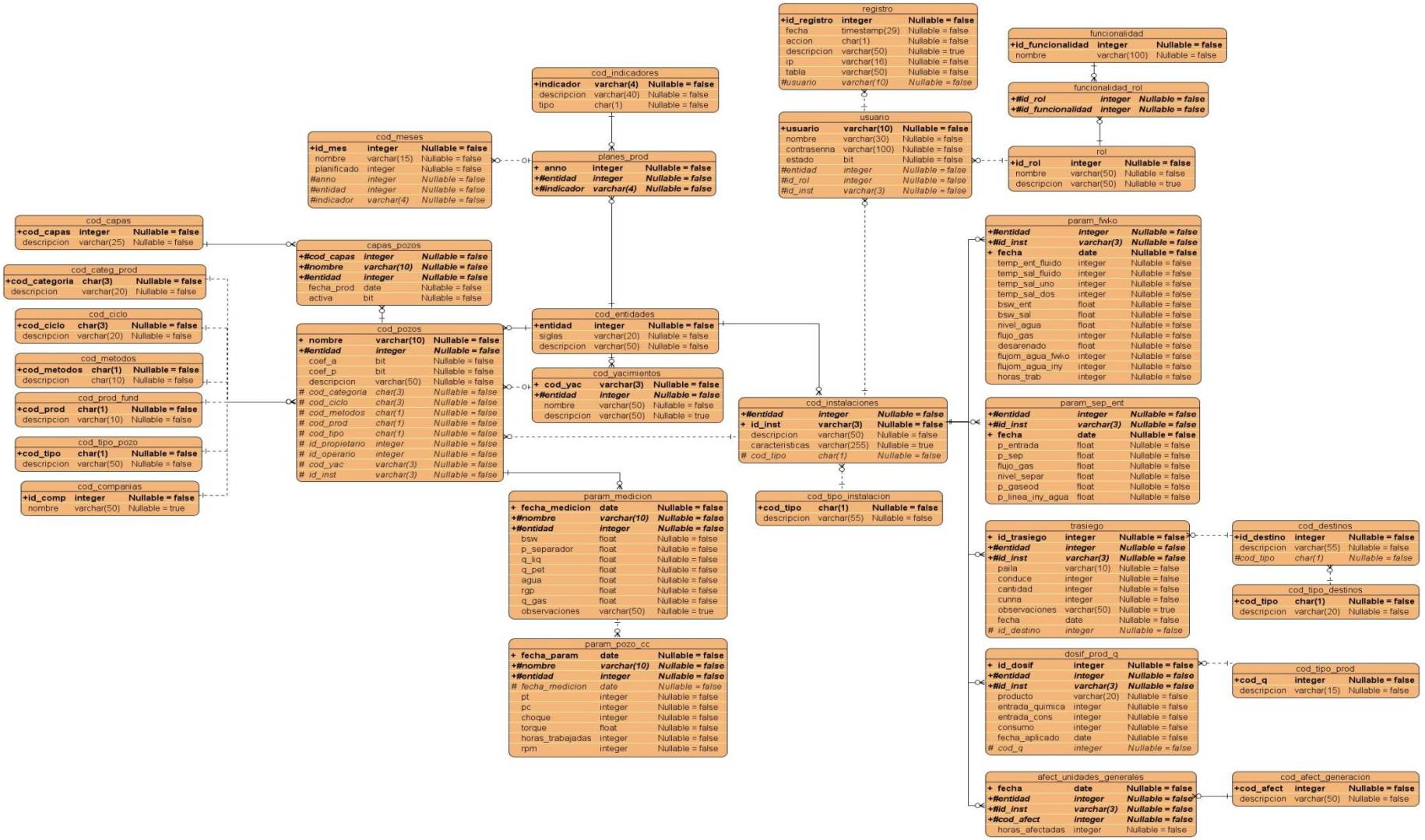
Anexo 1. Modelo de Objeto.



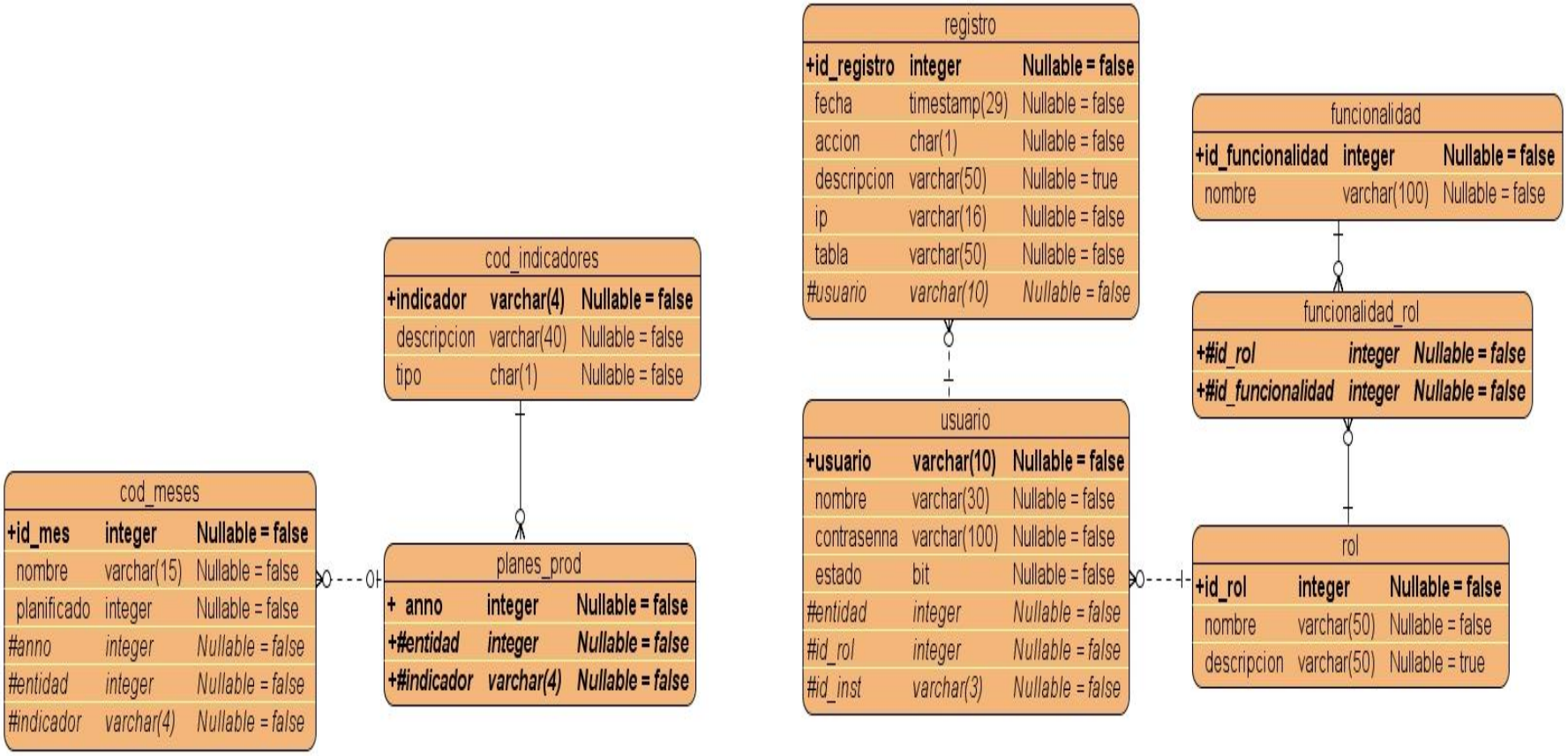
Anexo 2. Diagrama de Clases Persistentes.



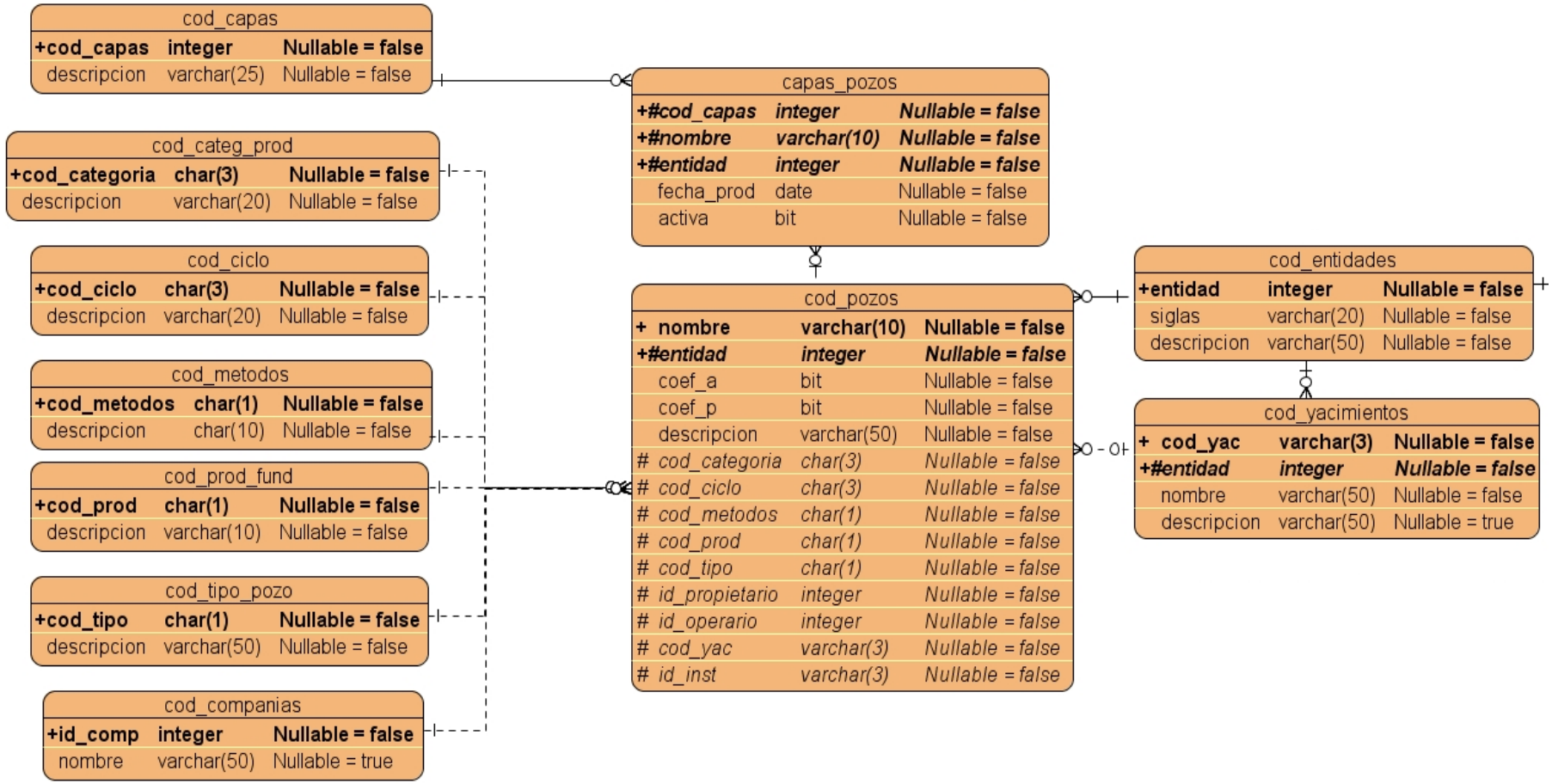
Anexo 3. Modelo Entidad-Relación.



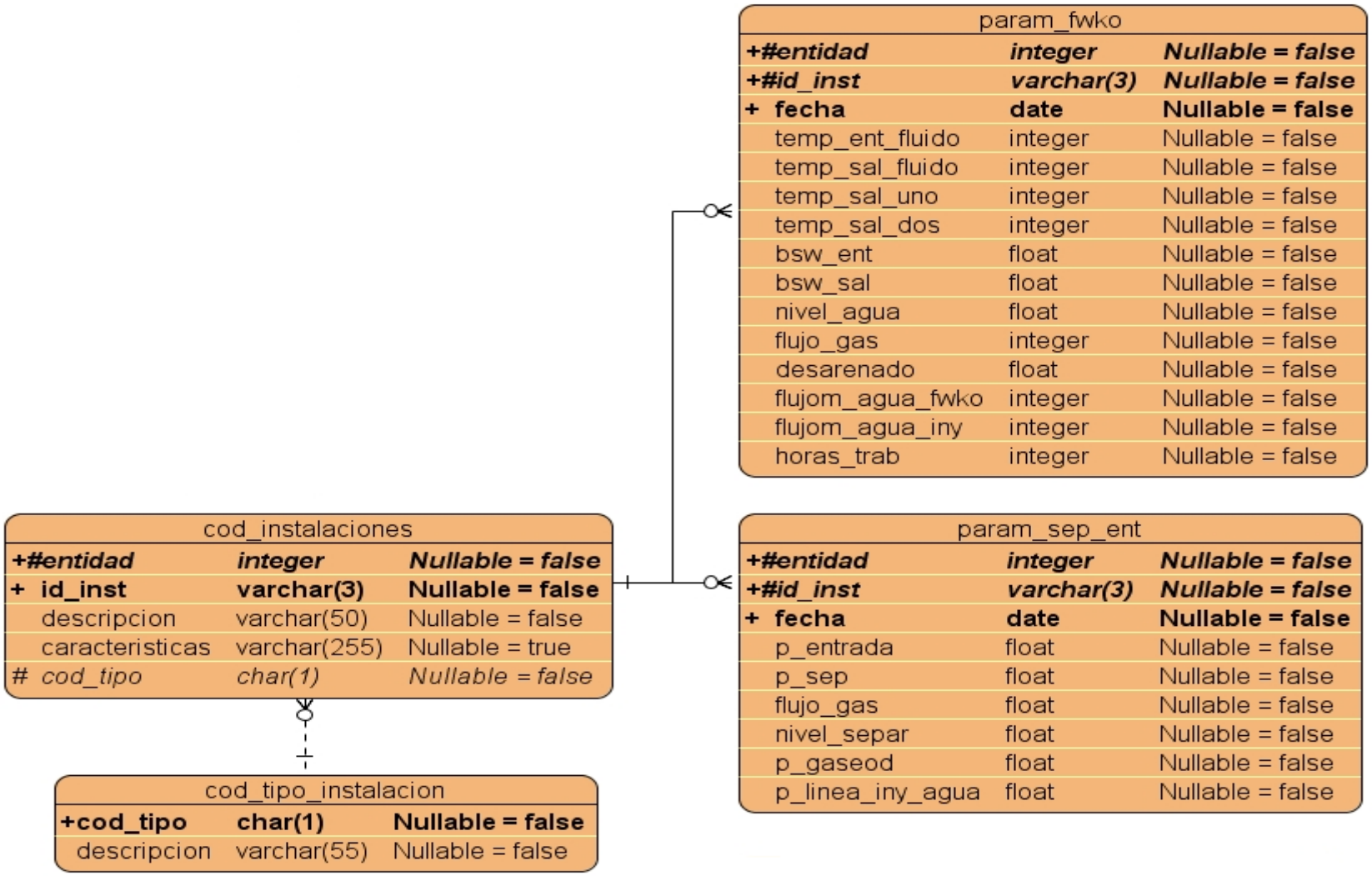
Anexo 3.1. Modelo Entidad-Relación parte 1.



Anexo 3.2. Modelo Entidad-Relación parte 2.



Anexo 3.3. Modelo Entidad-Relación parte 3.



Anexo 3.4. Modelo Entidad-Relación parte 4.